NAME

intro — introduction to general commands (tools and utilities)

DESCRIPTION

Section one of the manual contains most of the commands which comprise the BSD user environment. Some of the commands included in section one are text editors, command shell interpreters, searching and sorting tools, file manipulation commands, system status commands, remote file copy commands, mail commands, compilers and compiler tools, formatted output tools, and line printer commands.

All commands set a status value upon exit which may be tested to see if the command completed normally. The exit values and their meanings are explained in the individual manuals. Traditionally, the value 0 signifies successful completion of the command.

SEE ALSO

man(1), intro(2), intro(3), intro(4), intro(5), intro(6), intro(7), intro(8), intro(9)

Tutorials in the *UNIX User's Manual Supplementary Documents*.

HISTORY

An **intro** manual appeared in Version 6 AT&T UNIX.

NAME

addr2line − convert addresses into file names and line numbers.

SYNOPSIS

addr2line [**−b** *bfdname*|**−−target=***bfdname*]
[**−C**|**−−demangle**[=*style*]]
[**−e** *filename*|**−−exe=***filename*]
[**−f**|**−−functions**] [**−s**|**−−basename**]
[**−H**|**−−help**] [**−V**|**−−version**]
[addr addr ...]

DESCRIPTION

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

The executable to use is specified with the **−e** option. The default is the file *a.out*.

addr2line has two modes of operation.

In the first, hexadecimal addresses are specified on the command line, and **addr2line** displays the file name and line number for each address.

In the second, **addr2line** reads hexadecimal addresses from standard input, and prints the file name and line number for each address on standard output. In this mode, **addr2line** may be used in a pipe to convert dynamically chosen addresses.

The format of the output is **FILENAME:LINENO**. The file name and line number for each address is printed on a separate line. If the **−f** option is used, then each **FILENAME:LINENO** line is preceded by a **FUNCTIONNAME** line which is the name of the function containing the address.

If the file name or function name can not be determined, **addr2line** will print two question marks in their place. If the line number can not be determined, **addr2line** will print 0.

OPTIONS

The long and short forms of options, shown here as alternatives, are equivalent.

−b *bfdname*
−−target=*bfdname*
Specify that the object-code format for the object files is *bfdname*.

−C
−−demangle[=*style***]**
Decode (*demangle*) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

−e *filename*
−−exe=*filename*
Specify the name of the executable for which addresses should be translated. The default file is *a.out*.

−f
−−functions
Display function names as well as file and line number information.

−s
−−basenames
Display only the base of each file name.

SEE ALSO

Info entries for *binutils*.

COPYRIGHT

NAME

afslog — obtain AFS tokens

SYNOPSIS

afslog [**-h** | **--help**][**--no-v4**][**--no-v5**][**-u** | **--unlog**][**-v** | **--verbose**] [**--version**][**-c** *cell* | **--cell=***cell*][**-k** *realm* | **--realm=***realm*][**-P** *principal* | **--principal=***principal*][**-p** *path* | **--file=***path*] [*cell* | *path ...*]

DESCRIPTION

afslog obtains AFS tokens for a number of cells. What cells to get tokens for can either be specified as an explicit list, as file paths to get tokens for, or be left unspecified, in which case **afslog** will use whatever magic `krb_afslog`(3) decides upon.

Supported options:

--no-v4
: This makes **afslog** not try using Kerberos 4.

--no-v5
: This makes **afslog** not try using Kerberos 5.

-P *principal*, **--principal** *principal*
: select what Kerberos 5 principal to use.

--cache *cache*
: select what Kerberos 5 credential cache to use. **--principal** overrides this option.

-u, **--unlog**
: Destroy tokens instead of obtaining new. If this is specified, all other options are ignored (except for **--help** and **--version**).

-v, **--verbose**
: Adds more verbosity for what is actually going on.

-c *cell,* **--cell=***cell*
: This specified one or more cell names to get tokens for.

-k *realm*, **--realm=***realm*
: This is the Kerberos realm the AFS servers live in, this should normally not be specified.

-p *path*, **--file=***path*
: This specified one or more file paths for which tokens should be obtained.

Instead of using **-c** and **-p**, you may also pass a list of cells and file paths after any other options. These arguments are considered files if they are either the strings "." or ".." or they contain a slash, or if there exists a file by that name.

EXAMPLES

Assuming that there is no file called "openafs.org" in the current directory, and that `/afs/openafs.org` points to that cell, the follwing should be identical:

```
$ afslog -c openafs.org
$ afslog openafs.org
$ afslog /afs/openafs.org/some/file
```

SEE ALSO

`krb_afslog`(3)

NAME

altqstat — show altq status

SYNOPSIS

altqstat [**-enrs**] [**-c** *count*] [**-w** *wait*] [**-i** *interface*] [**-I** *input_interface*]

DESCRIPTION

The **altqstat** command displays the status of a queueing discipline. The contents displayed by **altqstat** is specific to each queueing discipline.

The options are as follows:

-e Echo communication with altqd(8) to standard output. This option is for debugging.

-n Disable communication with altqd(8). The interface should be explicitly specified.

-r Enter the raw console mode to talk to altqd(8). This option is for debugging queue information exchange between **altqstat** and altqd(8).

-s List all interfaces, classes and filters currently installed.

-c *count* **altqstat** exits after displaying *count* times. If no repeat *count* is specified, the default is infinity.

-w *wait* Pause *wait* seconds between each display. If no repeat *wait* interval is specified, the default is 5 seconds.

-i *interface*
Show information about the specified interface. If no *interface* is specified, the default interface is the first interface returned from altqd(8).

-I *input_interface*
Show information about the specified input interface. This option is used to specify *traffic conditioner* at an input interface.

FILES

/var/run/altq_quip Unix domain socket for communicating with altqd(8)

SEE ALSO

altq.conf(5), altqd(8), altq(9)

NAME

ansi2knr − convert ANSI C to Kernighan & Ritchie C

SYNOPSIS

ansi2knr [--varargs] input_file [output_file]

DESCRIPTION

If no output_file is supplied, output goes to stdout.
There are no error messages.

ansi2knr recognizes function definitions by seeing a non-keyword identifier at the left margin, followed by a left parenthesis, with a right parenthesis as the last character on the line, and with a left brace as the first token on the following line (ignoring possible intervening comments). It will recognize a multi-line header provided that no intervening line ends with a left or right brace or a semicolon. These algorithms ignore whitespace and comments, except that the function name must be the first thing on the line.

The following constructs will confuse it:

- Any other construct that starts at the left margin and follows the above syntax (such as a macro or function call).
- Some macros that tinker with the syntax of the function header.

The --varargs switch is obsolete, and is recognized only for backwards compatibility. The present version of *ansi2knr* will always attempt to convert a ... argument to va_alist and va_dcl.

AUTHOR

L. Peter Deutsch <ghost@aladdin.com> wrote the original ansi2knr and continues to maintain the current version; most of the code in the current version is his work. ansi2knr also includes contributions by Francois Pinard <pinard@iro.umontreal.ca> and Jim Avera <jima@netcom.com>.

NAME

ansi2knr − convert ANSI C to Kernighan & Ritchie C

SYNOPSIS

ansi2knr [--varargs] input_file [output_file]

DESCRIPTION

If no output_file is supplied, output goes to stdout.
There are no error messages.

ansi2knr recognizes function definitions by seeing a non-keyword identifier at the left margin, followed by a left parenthesis, with a right parenthesis as the last character on the line, and with a left brace as the first token on the following line (ignoring possible intervening comments). It will recognize a multi-line header provided that no intervening line ends with a left or right brace or a semicolon. These algorithms ignore whitespace and comments, except that the function name must be the first thing on the line.

The following constructs will confuse it:

- Any other construct that starts at the left margin and follows the above syntax (such as a macro or function call).
- Some macros that tinker with the syntax of the function header.

The --varargs switch is obsolete, and is recognized only for backwards compatibility. The present version of *ansi2knr* will always attempt to convert a ... argument to va_alist and va_dcl.

AUTHOR

L. Peter Deutsch <ghost@aladdin.com> wrote the original ansi2knr and continues to maintain the current version; most of the code in the current version is his work. ansi2knr also includes contributions by Francois Pinard <pinard@iro.umontreal.ca> and Jim Avera <jima@netcom.com>.

NAME

aout2hux — convert a.out/ELF executable to Human68k .x format

SYNOPSIS

aout2hux [**-o** *output_file*] *aout1 loadaddr1 aout2 loadaddr2*

DESCRIPTION

aout2hux reads two a.out(5) or ELF format executables with different load addresses and generate Human68k '.x' format executable.

If the input files are a.out, they must be static OMAGIC / NMAGIC m68k executables. If the input files are ELF, they must be static m68k executables. Two input executables must be created from the same objects, and have different loading addresses. Each of the load address is specified by a hexadecimal number. Load address shall be multiple of 4 for as(1) and ld(1) of NetBSD/m68k.

If the output file is not specified by **-o** option, the name out.x is used.

FILES

out.x default output file.

EXAMPLES

The following command sequence creates a Human68k executable foo.x from object files a.o and b.o:

```
cc -N -nostdlib -static -Wl,-Text,0        -o aout1 a.o b.o
cc -N -nostdlib -static -Wl,-Text,10203040 -o aout2 a.o b.o
aout2hux -o foo.x aout1 0 aout2 10203040
```

This example uses 0x0 and 0x10203040 as the load addresses.

SEE ALSO

as(1), cc(1), ld(1), a.out(5)

HISTORY

The **aout2hux** utility first appeared in NetBSD 1.4.

BUGS

Symbol and debugging information is not converted.

The generated executable is not so effective as that of Human68k native compiler.

NAME

apply — apply a command to a set of arguments

SYNOPSIS

apply [**-a***c*] [**-***#*] *command argument ...*

DESCRIPTION

apply runs the named *command* on each argument *argument* in turn.

Character sequences of the form "%d" in *command*, where "d" is a digit from 1 to 9, are replaced by the d´th following unused *argument*. In this case, the largest digit number of arguments are discarded for each execution of *command*.

The options are as follows:

-*#* Normally arguments are taken singly; the optional number **-#** specifies the number of arguments to be passed to *command*. If the number is zero, *command* is run, without arguments, once for each *argument*.

If any sequences of "%d" occur in command, the **-#** option is ignored.

-a*c*
The use of the character "%" as a magic character may be changed with the **-a** option.

ENVIRONMENT

The following environment variable affects the execution of **apply**:

SHELL Pathname of shell to use. If this variable is not defined, the Bourne shell is used.

FILES

/bin/sh Default shell

EXAMPLES

```
apply echo *
```
is similar to ls(1);
```
apply -2 cmp a1 b1 a2 b2 a3 b3
```
compares the 'a' files to the 'b' files;
```
apply -0 who 1 2 3 4 5
```
runs who(1) 5 times; and
```
apply 'ln %1 /usr/joe' *
```
links all files in the current directory to the directory /usr/joe.

HISTORY

The **apply** command appeared in 4.2BSD.

AUTHORS

Rob Pike

BUGS

Shell metacharacters in *command* may have bizarre effects; it is best to enclose complicated commands in single quotes ('').

NAME

apropos — locate commands by keyword lookup

SYNOPSIS

apropos [**-C** *path*] [**-M** *path*] [**-m** *path*] *keyword ...*

DESCRIPTION

apropos shows which manual pages contain instances of any of the given *keyword(s)* in their title line. Each word is considered separately and case of letters is ignored. Words which are part of other words are considered; when looking for "compile", **apropos** will also list all instances of "compiler".

If the line output by **apropos** starts "`name(section) ...`" you can enter "`man section name`" to get its documentation.

The options are as follows:

-C Use different man(1) configuration file than the default, `/etc/man.conf`.

-M Override the list of standard directories **apropos** searches for a database named `whatis.db`. The supplied *path* must be a colon ":" separated list of directories. This search path may also be set using the environment variable `MANPATH`.

-m Augment the list of standard directories **apropos** searches for its database. The supplied *path* must be a colon ":" separated list of directories. These directories will be searched before the standard directories, or the directories supplied with the **-M** option or the `MANPATH` environment variable.

ENVIRONMENT

`MANPATH` The standard search path used by man(1) may be overridden by specifying a path in the `MANPATH` environment variable. The format of the path is a colon ":" separated list of directories.

FILES

`whatis.db`	name of the apropos database
`/etc/man.conf`	man(1) configuration file, used to get location of whatis database if `MANPATH` is not set.

SEE ALSO

man(1), whatis(1), whereis(1), man.conf(5), makewhatis(8)

HISTORY

The **apropos** command appeared in 3.0 BSD.

NAME

ar − create, modify, and extract from archives

SYNOPSIS

ar [**−X32_64**] [**−**]*p*[*mod* [*relpos*] [*count*]] *archive* [*member*...]

DESCRIPTION

The GNU **ar** program creates, modifies, and extracts from archives. An *archive* is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called *members* of the archive).

The original files' contents, mode (permissions), timestamp, owner, and group are preserved in the archive, and can be restored on extraction.

GNU **ar** can maintain archives whose members have names of any length; however, depending on how **ar** is configured on your system, a limit on member-name length may be imposed for compatibility with archive formats maintained with other tools. If it exists, the limit is often 15 characters (typical of formats related to a.out) or 16 characters (typical of formats related to coff).

ar is considered a binary utility because archives of this sort are most often used as *libraries* holding commonly needed subroutines.

ar creates an index to the symbols defined in relocatable object modules in the archive when you specify the modifier **s**. Once created, this index is updated in the archive whenever **ar** makes a change to its contents (save for the **q** update operation). An archive with such an index speeds up linking to the library, and allows routines in the library to call each other without regard to their placement in the archive.

You may use **nm −s** or **nm −−print−armap** to list this index table. If an archive lacks the table, another form of **ar** called **ranlib** can be used to add just the table.

GNU **ar** is designed to be compatible with two different facilities. You can control its activity using command-line options, like the different varieties of **ar** on Unix systems; or, if you specify the single command-line option **−M**, you can control it with a script supplied via standard input, like the MRI ''librarian'' program.

OPTIONS

GNU **ar** allows you to mix the operation code *p* and modifier flags *mod* in any order, within the first command-line argument.

If you wish, you may begin the first command-line argument with a dash.

The *p* keyletter specifies what operation to execute; it may be any of the following, but you must specify only one of them:

d *Delete* modules from the archive. Specify the names of modules to be deleted as *member*...; the archive is untouched if you specify no files to delete.

If you specify the **v** modifier, **ar** lists each module as it is deleted.

m Use this operation to *move* members in an archive.

The ordering of members in an archive can make a difference in how programs are linked using the library, if a symbol is defined in more than one member.

If no modifiers are used with `m`, any members you name in the *member* arguments are moved to the *end* of the archive; you can use the **a**, **b**, or **i** modifiers to move them to a specified place instead.

p *Print* the specified members of the archive, to the standard output file. If the **v** modifier is specified, show the member name before copying its contents to standard output.

If you specify no *member* arguments, all the files in the archive are printed.

q *Quick append*; Historically, add the files *member*... to the end of *archive*, without checking for replacement.

The modifiers **a**, **b**, and **i** do *not* affect this operation; new members are always placed at the end of the

archive.

The modifier **v** makes **ar** list each file as it is appended.

Since the point of this operation is speed, the archive's symbol table index is not updated, even if it already existed; you can use **ar s** or **ranlib** explicitly to update the symbol table index.

However, too many different systems assume quick append rebuilds the index, so GNU **ar** implements **q** as a synonym for **r**.

r Insert the files *member*... into *archive* (with *replacement*). This operation differs from **q** in that any previously existing members are deleted if their names match those being added.

If one of the files named in *member*... does not exist, **ar** displays an error message, and leaves undisturbed any existing members of the archive matching that name.

By default, new members are added at the end of the file; but you may use one of the modifiers **a**, **b**, or **i** to request placement relative to some existing member.

The modifier **v** used with this operation elicits a line of output for each file inserted, along with one of the letters **a** or **r** to indicate whether the file was appended (no old member deleted) or replaced.

t Display a *table* listing the contents of *archive*, or those of the files listed in *member*... that are present in the archive. Normally only the member name is shown; if you also want to see the modes (permissions), timestamp, owner, group, and size, you can request that by also specifying the **v** modifier.

If you do not specify a *member*, all files in the archive are listed.

If there is more than one file with the same name (say, **fie**) in an archive (say **b.a**), **ar t b.a fie** lists only the first instance; to see them all, you must ask for a complete listing−−−in our example, **ar t b.a**.

x *Extract* members (named *member*) from the archive. You can use the **v** modifier with this operation, to request that **ar** list each name as it extracts it.

If you do not specify a *member*, all files in the archive are extracted.

A number of modifiers (*mod*) may immediately follow the *p* keyletter, to specify variations on an operation's behavior:

a Add new files *after* an existing member of the archive. If you use the modifier **a**, the name of an existing archive member must be present as the *relpos* argument, before the *archive* specification.

b Add new files *before* an existing member of the archive. If you use the modifier **b**, the name of an existing archive member must be present as the *relpos* argument, before the *archive* specification. (same as **i**).

c *Create* the archive. The specified *archive* is always created if it did not exist, when you request an update. But a warning is issued unless you specify in advance that you expect to create it, by using this modifier.

f Truncate names in the archive. GNU **ar** will normally permit file names of any length. This will cause it to create archives which are not compatible with the native **ar** program on some systems. If this is a concern, the **f** modifier may be used to truncate file names when putting them in the archive.

i Insert new files *before* an existing member of the archive. If you use the modifier **i**, the name of an existing archive member must be present as the *relpos* argument, before the *archive* specification. (same as **b**).

l This modifier is accepted but not used.

N Uses the *count* parameter. This is used if there are multiple entries in the archive with the same name. Extract or delete instance *count* of the given name from the archive.

o Preserve the *original* dates of members when extracting them. If you do not specify this modifier, files extracted from the archive are stamped with the time of extraction.

P Use the full path name when matching names in the archive. GNU **ar** can not create an archive with a full path name (such archives are not POSIX complaint), but other archive creators can. This option will cause GNU **ar** to match file names using a complete path name, which can be convenient when extracting a single file from an archive created by another tool.

s Write an object-file index into the archive, or update an existing one, even if no other change is made to the archive. You may use this modifier flag either with any operation, or alone. Running **ar s** on an archive is equivalent to running **ranlib** on it.

S Do not generate an archive symbol table. This can speed up building a large library in several steps. The resulting archive can not be used with the linker. In order to build a symbol table, you must omit the **S** modifier on the last execution of **ar**, or you must run **ranlib** on the archive.

u Normally, **ar r**... inserts all files listed into the archive. If you would like to insert *only* those of the files you list that are newer than existing members of the same names, use this modifier. The **u** modifier is allowed only for the operation **r** (replace). In particular, the combination **qu** is not allowed, since checking the timestamps would lose any speed advantage from the operation **q**.

v This modifier requests the *verbose* version of an operation. Many operations display additional information, such as filenames processed, when the modifier **v** is appended.

V This modifier shows the version number of **ar**.

ar ignores an initial option spelt **−X32_64**, for compatibility with AIX. The behaviour produced by this option is the default for GNU **ar**. **ar** does not support any of the other **−X** options; in particular, it does not support **−X32** which is the default for AIX **ar**.

SEE ALSO

nm (1), *ranlib* (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

AS − the portable GNU assembler.

SYNOPSIS

as [**−a**[**cdhlns**][=*file*]] [**−−alternate**] [**−D**]
[**−−defsym** *sym=val*] [**−f**] [**−g**] [**−−gstabs**] [**−−gstabs+**]
[**−−gdwarf−2**] [**−−help**] [**−I** *dir*] [**−J**] [**−K**] [**−L**]
[**−−listing−lhs−width**=*NUM*] [**−−listing−lhs−width2**=*NUM*]
[**−−listing−rhs−width**=*NUM*] [**−−listing−cont−lines**=*NUM*]
[**−−keep−locals**] [**−o** *objfile*] [**−R**] [**−−statistics**] [**−v**]
[**−version**] [**−−version**] [**−W**] [**−−warn**] [**−−fatal−warnings**]
[**−w**] [**−x**] [**−Z**] [**−−target−help**] [*target-options*]
[**−−**|*files* ...]

Target Alpha options:
[**−m***cpu*]
[**−mdebug** | **−no−mdebug**]
[**−relax**] [**−g**] [**−G***size*]
[**−F**] [**−32addr**]

Target ARC options:
[**−marc[5|6|7|8]**]
[**−EB**|**−EL**]

Target ARM options:
[**−mcpu**=*processor*[+*extension*...]]
[**−march**=*architecture*[+*extension*...]]
[**−mfpu**=*floating-point-format*]
[**−mfloat−abi**=*abi*]
[**−meabi**=*ver*]
[**−mthumb**]
[**−EB**|**−EL**]
[**−mapcs−32**|**−mapcs−26**|**−mapcs−float**|
−mapcs−reentrant]
[**−mthumb−interwork**] [**−k**]

Target CRIS options:
[**−−underscore** | **−−no−underscore**]
[**−−pic**] [**−N**]
[**−−emulation=criself** | **−−emulation=crisaout**]
[**−−march=v0_v10** | **−−march=v10** | **−−march=v32** | **−−march=common_v10_v32**]

Target D10V options:
[**−O**]

Target D30V options:
[**−O**|**−n**|**−N**]

Target i386 options:
[**−−32**|**−−64**] [**−n**]

Target i960 options:
[**−ACA**|**−ACA_A**|**−ACB**|**−ACC**|**−AKA**|**−AKB**|
−AKC|**−AMC**]
[**−b**] [**−no−relax**]

Target IA−64 options:
[**−mconstant−gp**|**−mauto−pic**]
[**−milp32**|**−milp64**|**−mlp64**|**−mp64**]
[**−mle**|**mbe**]

[**−munwind−check=warning**|**−munwind−check=error**]
[**−mhint.b=ok**|**−mhint.b=warning**|**−mhint.b=error**]
[**−x**|**−xexplicit**] [**−xauto**] [**−xdebug**]

Target IP2K options:
[**−mip2022**|**−mip2022ext**]

Target M32R options:
[**−−m32rx**|**−−[no−]warn−explicit−parallel−conflicts**|
−−W[n]p]

Target M680X0 options:
[**−l**] [**−m68000**|**−m68010**|**−m68020**|...]

Target M68HC11 options:
[**−m68hc11**|**−m68hc12**|**−m68hcs12**]
[**−mshort**|**−mlong**]
[**−mshort−double**|**−mlong−double**]
[**−−force−long−branches**] [**−−short−branches**]
[**−−strict−direct−mode**] [**−−print−insn−syntax**]
[**−−print−opcodes**] [**−−generate−example**]

Target MCORE options:
[**−jsri2bsr**] [**−sifilter**] [**−relax**]
[**−mcpu=[210**|**340]**]

Target MIPS options:
[**−nocpp**] [**−EL**] [**−EB**] [**−O**[*optimization level*]]
[**−g**[*debug level*]] [**−G** *num*] [**−KPIC**] [**−call_shared**]
[**−non_shared**] [**−xgot**]
[**−mabi**=*ABI*] [**−32**] [**−n32**] [**−64**] [**−mfp32**] [**−mgp32**]
[**−march**=*CPU*] [**−mtune**=*CPU*] [**−mips1**] [**−mips2**]
[**−mips3**] [**−mips4**] [**−mips5**] [**−mips32**] [**−mips32r2**]
[**−mips64**] [**−mips64r2**]
[**−construct−floats**] [**−no−construct−floats**]
[**−trap**] [**−no−break**] [**−break**] [**−no−trap**]
[**−mfix7000**] [**−mno−fix7000**]
[**−mips16**] [**−no−mips16**]
[**−mips3d**] [**−no−mips3d**]
[**−mdmx**] [**−no−mdmx**]
[**−mdebug**] [**−no−mdebug**]
[**−mpdr**] [**−mno−pdr**]

Target MMIX options:
[**−−fixed−special−register−names**] [**−−globalize−symbols**]
[**−−gnu−syntax**] [**−−relax**] [**−−no−predefined−symbols**]
[**−−no−expand**] [**−−no−merge−gregs**] [**−x**]
[**−−linker−allocated−gregs**]

Target PDP11 options:
[**−mpic**|**−mno−pic**] [**−mall**] [**−mno−extensions**]
[**−m***extension*|**−mno−***extension*]
[**−m***cpu*] [**−m***machine*]

Target picoJava options:
[**−mb**|**−me**]

Target PowerPC options:
[**−mpwrx**|**−mpwr2**|**−mpwr**|**−m601**|**−mppc**|**−mppc32**|**−m603**|**−m604**|
−m403|**−m405**|**−mppc64**|**−m620**|**−mppc64bridge**|**−mbooke**|

−mbooke32|**−mbooke64**]
[**−mcom**|**−many**|**−maltivec**] [**−memb**]
[**−mregnames**|**−mno−regnames**]
[**−mrelocatable**|**−mrelocatable−lib**]
[**−mlittle**|**−mlittle−endian**|**−mbig**|**−mbig−endian**]
[**−msolaris**|**−mno−solaris**]

Target SPARC options:
[**−Av6**|**−Av7**|**−Av8**|**−Asparclet**|**−Asparclite**
−Av8plus|**−Av8plusa**|**−Av9**|**−Av9a**]
[**−xarch=v8plus**|**−xarch=v8plusa**] [**−bump**]
[**−32**|**−64**]

Target TIC54X options:
[**−mcpu=54[123589]**|**−mcpu=54[56]lp**] [**−mfar−mode**|**−mf**]
[**−merrors−to−file** <*filename*>|**−me** <*filename*>]

Target Xtensa options:
[**−−[no−]text−section−literals**] [**−−[no−]absolute−literals**]
[**−−[no−]target−align**] [**−−[no−]longcalls**]
[**−−[no−]transform**]
[**−−rename−section** *oldname=newname*]

DESCRIPTION

GNU **as** is really a family of assemblers. If you use (or have used) the GNU assembler on one architecture, you should find a fairly similar environment when you use it on another architecture. Each version has much in common with the others, including object file formats, most assembler directives (often called *pseudo-ops*) and assembler syntax.

as is primarily intended to assemble the output of the GNU C compiler `gcc` for use by the linker `ld`. Nevertheless, we've tried to make **as** assemble correctly everything that other assemblers for the same machine would assemble. Any exceptions are documented explicitly. This doesn't mean **as** always uses the same syntax as another assembler for the same architecture; for example, we know of several incompatible versions of 680x0 assembly language syntax.

Each time you run **as** it assembles exactly one source program. The source program is made up of one or more files. (The standard input is also a file.)

You give **as** a command line that has zero or more input file names. The input files are read (from left file name to right). A command line argument (in any position) that has no special meaning is taken to be an input file name.

If you give **as** no file names it attempts to read one input file from the **as** standard input, which is normally your terminal. You may have to type **ctl-D** to tell **as** there is no more program to assemble.

Use `−−` if you need to explicitly name the standard input file in your command line.

If the source is empty, **as** produces a small, empty object file.

as may write warnings and error messages to the standard error file (usually your terminal). This should not happen when a compiler runs **as** automatically. Warnings report an assumption made so that **as** could keep assembling a flawed program; errors report a grave problem that stops the assembly.

If you are invoking **as** via the GNU C compiler, you can use the **−Wa** option to pass arguments through to the assembler. The assembler arguments must be separated from each other (and the **−Wa**) by commas. For example:

```
gcc −c −g −O −Wa,−alh,−L file.c
```

This passes two options to the assembler: **−alh** (emit a listing to standard output with high-level and assembly source) and **−L** (retain local symbols in the symbol table).

Usually you do not need to use this **−Wa** mechanism, since many compiler command-line options are

automatically passed to the assembler by the compiler. (You can call the GNU compiler driver with the **−v** option to see precisely what options it passes to each compilation pass, including the assembler.)

OPTIONS

−a[cdhlmns]
Turn on listings, in any of a variety of ways:

−ac
omit false conditionals

−ad
omit debugging directives

−ah
include high-level source

−al include assembly

−am
include macro expansions

−an
omit forms processing

−as include symbols

=file
set the name of the listing file

You may combine these options; for example, use **−aln** for assembly listing without forms processing. The **=file** option, if used, must be the last one. By itself, **−a** defaults to **−ahls**.

−−alternate
Begin in alternate macro mode, see @ref{Altmacro,,.altmacro}.

−D Ignored. This option is accepted for script compatibility with calls to other assemblers.

−−defsym *sym=value*
Define the symbol *sym* to be *value* before assembling the input file. *value* must be an integer constant. As in C, a leading **0x** indicates a hexadecimal value, and a leading **0** indicates an octal value.

−f ''fast''−−−skip whitespace and comment preprocessing (assume source is compiler output).

−g
−−gen−debug
Generate debugging information for each assembler source line using whichever debug format is preferred by the target. This currently means either STABS, ECOFF or DWARF2.

−−gstabs
Generate stabs debugging information for each assembler line. This may help debugging assembler code, if the debugger can handle it.

−−gstabs+
Generate stabs debugging information for each assembler line, with GNU extensions that probably only gdb can handle, and that could make other debuggers crash or refuse to read your program. This may help debugging assembler code. Currently the only GNU extension is the location of the current working directory at assembling time.

−−gdwarf−2
Generate DWARF2 debugging information for each assembler line. This may help debugging assembler code, if the debugger can handle it. Note−−−this option is only supported by some targets, not all of them.

−−help
Print a summary of the command line options and exit.

--target-help
Print a summary of all target specific options and exit.

-I *dir*
Add directory *dir* to the search list for `.include` directives.

-J Don't warn about signed overflow.

-K Issue warnings when difference tables altered for long displacements.

-L
--keep-locals
Keep (in the symbol table) local symbols. On traditional a.out systems these start with **L**, but different systems have different local label prefixes.

--listing-lhs-width=*number*
Set the maximum width, in words, of the output data column for an assembler listing to *number*.

--listing-lhs-width2=*number*
Set the maximum width, in words, of the output data column for continuation lines in an assembler listing to *number*.

--listing-rhs-width=*number*
Set the maximum width of an input source line, as displayed in a listing, to *number* bytes.

--listing-cont-lines=*number*
Set the maximum number of lines printed in a listing for a single line of input to *number* + 1.

-o *objfile*
Name the object-file output from **as** *objfile*.

-R Fold the data section into the text section.

--statistics
Print the maximum space (in bytes) and total time (in seconds) used by assembly.

--strip-local-absolute
Remove local absolute symbols from the outgoing symbol table.

-v
-version
Print the **as** version.

--version
Print the **as** version and exit.

-W
--no-warn
Suppress warning messages.

--fatal-warnings
Treat warnings as errors.

--warn
Don't suppress warning messages or treat them as errors.

-w Ignored.

-x Ignored.

-Z Generate an object file even after errors.

-- | *files* **...**
Standard input, or source files to assemble.

The following options are available when as is configured for an ARC processor.

−marc[5|6|7|8]
This option selects the core processor variant.

−EB | −EL
Select either big-endian (−EB) or little-endian (−EL) output.

The following options are available when as is configured for the ARM processor family.

−mcpu=*processor***[+***extension***...]**
Specify which ARM processor variant is the target.

−march=*architecture***[+***extension***...]**
Specify which ARM architecture variant is used by the target.

−mfpu=*floating-point-format*
Select which Floating Point architecture is the target.

−mfloat−abi=*abi*
Select which floating point ABI is in use.

−mthumb
Enable Thumb only instruction decoding.

−mapcs−32 | −mapcs−26 | −mapcs−float | −mapcs−reentrant
Select which procedure calling convention is in use.

−EB | −EL
Select either big-endian (−EB) or little-endian (−EL) output.

−mthumb−interwork
Specify that the code has been generated with interworking between Thumb and ARM code in mind.

−k Specify that PIC code has been generated.

See the info pages for documentation of the CRIS-specific options.

The following options are available when as is configured for a D10V processor.

−O Optimize output by parallelizing instructions.

The following options are available when as is configured for a D30V processor.

−O Optimize output by parallelizing instructions.

−n Warn when nops are generated.

−N Warn when a nop after a 32−bit multiply instruction is generated.

The following options are available when as is configured for the Intel 80960 processor.

−ACA | −ACA_A | −ACB | −ACC | −AKA | −AKB | −AKC | −AMC
Specify which variant of the 960 architecture is the target.

−b Add code to collect statistics about branches taken.

−no−relax
Do not alter compare-and-branch instructions for long displacements; error if necessary.

The following options are available when as is configured for the Ubicom IP2K series.

−mip2022ext
Specifies that the extended IP2022 instructions are allowed.

−mip2022
Restores the default behaviour, which restricts the permitted instructions to just the basic IP2022 ones.

The following options are available when as is configured for the Renesas M32R (formerly Mitsubishi M32R) series.

--m32rx
Specify which processor in the M32R family is the target. The default is normally the M32R, but this option changes it to the M32RX.

--warn-explicit-parallel-conflicts or --Wp
Produce warning messages when questionable parallel constructs are encountered.

--no-warn-explicit-parallel-conflicts or --Wnp
Do not produce warning messages when questionable parallel constructs are encountered.

The following options are available when as is configured for the Motorola 68000 series.

-l Shorten references to undefined symbols, to one word instead of two.

-m68000 | -m68008 | -m68010 | -m68020 | -m68030
| -m68040 | -m68060 | -m68302 | -m68331 | -m68332
| -m68333 | -m68340 | -mcpu32 | -m5200
Specify what processor in the 68000 family is the target. The default is normally the 68020, but this can be changed at configuration time.

-m68881 | -m68882 | -mno-68881 | -mno-68882
The target machine does (or does not) have a floating-point coprocessor. The default is to assume a coprocessor for 68020, 68030, and cpu32. Although the basic 68000 is not compatible with the 68881, a combination of the two can be specified, since it's possible to do emulation of the coprocessor instructions with the main processor.

-m68851 | -mno-68851
The target machine does (or does not) have a memory-management unit coprocessor. The default is to assume an MMU for 68020 and up.

For details about the PDP-11 machine dependent features options, see @ref{PDP-11-Options}.

-mpic | -mno-pic
Generate position-independent (or position-dependent) code. The default is **-mpic**.

-mall
-mall-extensions
Enable all instruction set extensions. This is the default.

-mno-extensions
Disable all instruction set extensions.

-m*extension* | **-mno-***extension*
Enable (or disable) a particular instruction set extension.

-m*cpu*
Enable the instruction set extensions supported by a particular CPU, and disable all other extensions.

-m*machine*
Enable the instruction set extensions supported by a particular machine model, and disable all other extensions.

The following options are available when as is configured for a picoJava processor.

-mb
Generate ''big endian'' format output.

-ml
Generate ''little endian'' format output.

The following options are available when as is configured for the Motorola 68HC11 or 68HC12 series.

-m68hc11 | -m68hc12 | -m68hcs12
Specify what processor is the target. The default is defined by the configuration option when building the assembler.

−mshort
Specify to use the 16−bit integer ABI.

−mlong
Specify to use the 32−bit integer ABI.

−mshort−double
Specify to use the 32−bit double ABI.

−mlong−double
Specify to use the 64−bit double ABI.

−−force−long−branchs
Relative branches are turned into absolute ones. This concerns conditional branches, unconditional branches and branches to a sub routine.

−S | −−short−branchs
Do not turn relative branchs into absolute ones when the offset is out of range.

−−strict−direct−mode
Do not turn the direct addressing mode into extended addressing mode when the instruction does not support direct addressing mode.

−−print−insn−syntax
Print the syntax of instruction in case of error.

−−print−opcodes
print the list of instructions with syntax and then exit.

−−generate−example
print an example of instruction for each possible instruction and then exit. This option is only useful for testing **as**.

The following options are available when **as** is configured for the SPARC architecture:

−Av6 | −Av7 | −Av8 | −Asparclet | −Asparclite
−Av8plus | −Av8plusa | −Av9 | −Av9a
Explicitly select a variant of the SPARC architecture.

−Av8plus and **−Av8plusa** select a 32 bit environment. **−Av9** and **−Av9a** select a 64 bit environment.

−Av8plusa and **−Av9a** enable the SPARC V9 instruction set with UltraSPARC extensions.

−xarch=v8plus | −xarch=v8plusa
For compatibility with the Solaris v9 assembler. These options are equivalent to −Av8plus and −Av8plusa, respectively.

−bump
Warn when the assembler switches to another architecture.

The following options are available when as is configured for the 'c54x architecture.

−mfar−mode
Enable extended addressing mode. All addresses and relocations will assume extended addressing (usually 23 bits).

−mcpu=*CPU_VERSION*
Sets the CPU version being compiled for.

−merrors−to−file *FILENAME*
Redirect error output to a file, for broken systems which don't support such behaviour in the shell.

The following options are available when as is configured for a MIPS processor.

−G *num*
This option sets the largest size of an object that can be referenced implicitly with the gp register. It is only accepted for targets that use ECOFF format, such as a DECstation running Ultrix. The default

value is 8.

−EB
Generate ‘‘big endian’’ format output.

−EL
Generate ‘‘little endian’’ format output.

−mips1
−mips2
−mips3
−mips4
−mips5
−mips32
−mips32r2
−mips64
−mips64r2
Generate code for a particular MIPS Instruction Set Architecture level. **−mips1** is an alias for **−march=r3000**, **−mips2** is an alias for **−march=r6000**, **−mips3** is an alias for **−march=r4000** and **−mips4** is an alias for **−march=r8000**. **−mips5**, **−mips32**, **−mips32r2**, **−mips64**, and **−mips64r2** correspond to generic **MIPS V**, **MIPS32**, **MIPS32 Release 2**, **MIPS64**, and **MIPS64 Release 2** ISA processors, respectively.

−march=*CPU*
Generate code for a particular MIPS cpu.

−mtune=*cpu*
Schedule and tune for a particular MIPS cpu.

−mfix7000
−mno−fix7000
Cause nops to be inserted if the read of the destination register of an mfhi or mflo instruction occurs in the following two instructions.

−mdebug
−no−mdebug
Cause stabs-style debugging output to go into an ECOFF-style .mdebug section instead of the standard ELF .stabs sections.

−mpdr
−mno−pdr
Control generation of `.pdr` sections.

−mgp32
−mfp32
The register sizes are normally inferred from the ISA and ABI, but these flags force a certain group of registers to be treated as 32 bits wide at all times. **−mgp32** controls the size of general-purpose registers and **−mfp32** controls the size of floating-point registers.

−mips16
−no−mips16
Generate code for the MIPS 16 processor. This is equivalent to putting `.set mips16` at the start of the assembly file. **−no−mips16** turns off this option.

−mips3d
−no−mips3d
Generate code for the MIPS−3D Application Specific Extension. This tells the assembler to accept MIPS−3D instructions. **−no−mips3d** turns off this option.

−mdmx

−no−mdmx
Generate code for the MDMX Application Specific Extension. This tells the assembler to accept MDMX instructions. **−no−mdmx** turns off this option.

−−construct−floats
−−no−construct−floats
The **−−no−construct−floats** option disables the construction of double width floating point constants by loading the two halves of the value into the two single width floating point registers that make up the double width register. By default **−−construct−floats** is selected, allowing construction of these floating point constants.

−−emulation=*name*
This option causes **as** to emulate **as** configured for some other target, in all respects, including output format (choosing between ELF and ECOFF only), handling of pseudo-opcodes which may generate debugging information or store symbol table information, and default endianness. The available configuration names are: **mipsecoff**, **mipself**, **mipslecoff**, **mipsbecoff**, **mipslelf**, **mipsbelf**. The first two do not alter the default endianness from that of the primary target for which the assembler was configured; the others change the default to little− or big-endian as indicated by the **b** or **l** in the name. Using **−EB** or **−EL** will override the endianness selection in any case.

This option is currently supported only when the primary target **as** is configured for is a MIPS ELF or ECOFF target. Furthermore, the primary target or others specified with **−−enable−targets=...** at configuration time must include support for the other format, if both are to be available. For example, the Irix 5 configuration includes support for both.

Eventually, this option will support more configurations, with more fine-grained control over the assembler's behavior, and will be supported for more processors.

−nocpp
as ignores this option. It is accepted for compatibility with the native tools.

−−trap
−−no−trap
−−break
−−no−break
Control how to deal with multiplication overflow and division by zero. **−−trap** or **−−no−break** (which are synonyms) take a trap exception (and only work for Instruction Set Architecture level 2 and higher); **−−break** or **−−no−trap** (also synonyms, and the default) take a break exception.

−n When this option is used, **as** will issue a warning every time it generates a nop instruction from a macro.

The following options are available when as is configured for an MCore processor.

−jsri2bsr
−nojsri2bsr
Enable or disable the JSRI to BSR transformation. By default this is enabled. The command line option **−nojsri2bsr** can be used to disable it.

−sifilter
−nosifilter
Enable or disable the silicon filter behaviour. By default this is disabled. The default can be overridden by the **−sifilter** command line option.

−relax
Alter jump instructions for long displacements.

−mcpu=[210|340]
Select the cpu type on the target hardware. This controls which instructions can be assembled.

−EB
Assemble for a big endian target.

−EL
Assemble for a little endian target.

See the info pages for documentation of the MMIX-specific options.

The following options are available when as is configured for an Xtensa processor.

−−text−section−literals | −−no−text−section−literals
With **−−text−section−literals**, literal pools are interspersed in the text section. The default is **−−no−text−section−literals**, which places literals in a separate section in the output file. These options only affect literals referenced via PC-relative `L32R` instructions; literals for absolute mode `L32R` instructions are handled separately.

−−absolute−literals | −−no−absolute−literals
Indicate to the assembler whether `L32R` instructions use absolute or PC-relative addressing. The default is to assume absolute addressing if the Xtensa processor includes the absolute `L32R` addressing option. Otherwise, only the PC-relative `L32R` mode can be used.

−−target−align | −−no−target−align
Enable or disable automatic alignment to reduce branch penalties at the expense of some code density. The default is **−−target−align**.

−−longcalls | −−no−longcalls
Enable or disable transformation of call instructions to allow calls across a greater range of addresses. The default is **−−no−longcalls**.

−−transform | −−no−transform
Enable or disable all assembler transformations of Xtensa instructions. The default is **−−transform**; **−−no−transform** should be used only in the rare cases when the instructions must be exactly as specified in the assembly source.

SEE ALSO

gcc (1), *ld* (1), and the Info entries for *binutils* and *ld*.

COPYRIGHT

NAME

asa — interpret carriage-control characters

SYNOPSIS

asa [*file ...*]

DESCRIPTION

The **asa** utility reads files sequentially, mapping FORTRAN carriage-control characters to line-printer control sequences, and writes them to the standard output.

The first character of each line is interpreted as a carriage-control character. The following characters are interpreted as follows:

<space>	Output the rest of the line without change.
0	Output a <newline> character before printing the rest of the line.
1	Output a <formfeed> character before printing the rest of the line.
+	The trailing <newline> of the previous line is replaced by a <carriage-return> before printing the rest of the line.

Lines beginning with characters other than the above are treated as if they begin with <space>.

EXIT STATUS

The **asa** utility exit 0 on success, and >0 if an error occurs.

EXAMPLES

To view a file containing the output of a FORTRAN program:

```
asa file
```

To format the output of a FORTRAN program and redirect it to a line-printer:

```
a.out | asa | lpr
```

SEE ALSO

f77(1)

STANDARDS

The **asa** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”) and to X/Open Commands and Utilities Issue 5 (“XCU5”).

AUTHORS

J.T. Conklin, Winning Strategies, Inc.

NAME

at, **batch**, **atq**, **atrm** — queue, examine or delete jobs for later execution

SYNOPSIS

at [**-bdlmrVv**] [**-f** *file*] [**-q** *queue*] **-t** *[[CC]YY]MMDDhhmm[.SS]*
at [**-bdlmrVv**] [**-f** *file*] [**-q** *queue*] *time*
at [**-V**] **-c** *job* [*job ...*]
atq [**-Vv**] [**-q** *queue*]
atrm [**-V**] *job* [*job ...*]
batch [**-mVv**] [**-f** *file*] [**-q** *queue*] [**-t** *[[CC]YY]MMDDhhmm[.SS]*]
batch [**-mVv**] [**-f** *file*] [**-q** *queue*] [*time*]

DESCRIPTION

at and **batch** read commands from standard input or a specified file which are to be executed at a later time, using sh(1).

at Executes commands at a specified time.

atq Lists the user's pending jobs, unless the user is the superuser. In that case, everybody's jobs are listed.

atrm Deletes jobs.

batch Executes commands when system load levels permit. In other words, when the load average drops below 1.5, or the value specified in the invocation of atrun(8).

at allows some moderately complex *time* specifications. It accepts times of the form *HHMM* or *HH:MM* to run a job at a specific time of day. (If that time is already past, the next day is assumed.) You may also specify 'midnight', 'noon', or 'teatime' (4pm) and you can have a time-of-day suffixed with 'AM' or 'PM' for running in the morning or the evening. You can also say what day the job will be run, by giving a date in the form *month-name day* with an optional *year*, or giving a date of the form *MMDDYY* or *MM/DD/YY* or *DD.MM.YY*. The specification of a date must follow the specification of the time of day. You can also give times like [**now**] or [**now**] '+ *count time-units*', where the time-units can be 'minutes', 'hours', 'days', 'weeks,' 'months,' or 'years' and you can tell **at** to run the job today by suffixing the time with 'today' and to run the job tomorrow by suffixing the time with 'tomorrow'.

For example, to run a job at 4pm three days from now, you would do

```
at 4pm + 3 days,
```

to run a job at 10:00am on July 31, you would do

```
at 10am Jul 31
```

and to run a job at 1am tomorrow, you would do

```
at 1am tomorrow.
```

Alternatively the time may be specified in a language-neutral fashion by using the **-t** options.

For both **at** and **batch**, commands are read from standard input or the file specified with the **-f** option and executed. The working directory, the environment (except for the variables TERM, TERMCAP, DISPLAY and _) and the *umask* are retained from the time of invocation. An **at** or **batch** command invoked from a su(1) shell will retain the current userid. The user will be mailed standard error and standard output from his commands, if any. Mail will be sent using the command sendmail(8). If **at** is executed from a su(1) shell, the owner of the login shell will receive the mail.

The superuser may use these commands in any case. For other users, permission to use at is determined by the files /var/at/at.allow and /var/at/at.deny.

If the file `/var/at/at.allow` exists, only usernames mentioned in it are allowed to use **at**.

If `/var/at/at.allow` does not exist, `/var/at/at.deny` is checked, every username not mentioned in it is then allowed to use **at**.

If neither exists, only the superuser is allowed use of **at**.

An empty `/var/at/at.deny` means that every user is allowed use these commands. This is the default configuration.

OPTIONS

-b Is an alias for **batch**.

-c Cats the jobs listed on the command line to standard output.

-d Is an alias for **atrm**.

-f *file* Reads the job from *file* rather than standard input.

-l Is an alias for **atq**.

-m Send mail to the user when the job has completed even if there was no output.

-q *queue* Uses the specified queue. A queue designation consists of a single letter. Valid queue designations range from 'a' to 'z' and 'A' to 'Z'. The 'c' queue is the default for **at** and the 'E' queue for **batch**. Queues with higher letters run with increased niceness. If a job is submitted to a queue designated with an uppercase letter, it is treated as if it had been submitted to batch at that time. If **atq** is given a specific queue, it will only show jobs pending in that queue.

-r Is an alias for **atrm**.

-t For both **at** and **batch**, the time may be specified in a language-neutral format consisting of:

- *CC* The first two digits of the year (the century).
- *YY* The second two digits of the year. If *YY* is specified, but *CC* is not, a value for *YY* between 69 and 99 results in a *CC* value of 19. Otherwise, a *CC* value of 20 is used.
- *MM* The month of the year, from 01 to 12.
- *DD* The day of the month, from 01 to 31.
- *hh* The hour of the day, from 00 to 23.
- *mm* The minute of the hour, from 00 to 59.
- *SS* The second of the minute, from 00 to 61.

-V Prints the version number to standard error.

-v For **atq**, shows completed but not yet deleted jobs in the queue. Otherwise shows the time the job will be executed.

FILES

`/var/at/jobs`	Directory containing job files
`/var/at/spool`	Directory containing output spool files
`/var/run/utmp`	Login records
`/var/at/at.allow`	Allow permission control
`/var/at/at.deny`	Deny permission control
`/var/at/.lockfile`	Job-creation lock file.

SEE ALSO

nice(1), sh(1), umask(2), atrun(8), cron(8), sendmail(8)

STANDARDS

The **at** and **batch** utilities conform to IEEE Std 1003.2-1992 (“POSIX.2”).

AUTHORS

At was mostly written by Thomas Koenig <ig25@rz.uni-karlsruhe.de>. The time parsing routines are by David Parsons <orc@pell.chi.il.us>.

BUGS

If the file /var/run/utmp is not available or corrupted, or if the user is not logged on at the time **at** is invoked, the mail is sent to the userid found in the environment variable LOGNAME. If that is undefined or empty, the current userid is assumed.

at and **batch** as presently implemented are not suitable when users are competing for resources. If this is the case for your site, you might want to consider another batch system, such as **nqs**.

NAME

atf-cleanup — safe removal of directory hierarchies

SYNOPSIS

atf-cleanup *path1* [*.. pathN*]
atf-cleanup -h

DESCRIPTION

atf-cleanup is a utility that removes files and directories recursively, but doing it in a safe manner to avoid crossing mount point boundaries. Before the removal of a given tree, **atf-cleanup** will look for file systems mounted anywhere inside it and try to unmount them. If all mount points were properly unmounted, it later proceeds to do the real removal. This tool is used internally by test programs to clean up the test cases' work directories, as they may have left other file systems mounted inside them.

In the first synopsis form, **atf-cleanup** proceeds to scan and remove all the given files or directories.

In the second synopsis form, **atf-cleanup** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

SEE ALSO

atf(7)

NAME

atf-compile — generates POSIX shell test programs

SYNOPSIS

atf-compile -o *out-file* *src1* [*.. srcN*]
atf-compile -h

DESCRIPTION

atf-compile generates a POSIX shell program based on a set of input files that implement test cases in this language. All the given source files are concatenated in order, and the tool prepends and appends to the results the necessary boilerplate to implement a test program directly runnable by the user.

In the first synopsis form, **atf-compile** will generate the *out-file* test program by adding the source files to it as well as all the necessary helper code to turn it into an executable.

In the second synopsis form, **atf-compile** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-o *out-file* Specifies the file to which the results will be written.

SEE ALSO

atf(7)

NAME

atf-config — queries static configuration information of ATF

SYNOPSIS

atf-config [**-t**] [*var1* [*.. varN*]]
atf-config -h

DESCRIPTION

atf-config is a utility that queries static configuration information of ATF. Static configuration refers to all those values for settings that were built into the ATF binaries at build time.

In the first synopsis form, **atf-config** will print variable-value pairs for all built-in static variables if no variable names are provided as arguments. If any is provided, it will only print the variable-value pairs for those variables. The output of the utility does not use the '=' symbol to separate the variable name from its corresponding value in an attempt to avoid sourcing the output in shell scripts or Makefiles. If you need to do that, the **-t** flag allows you to query the value of individual variables without any surrounding text.

In the second synopsis form, **atf-config** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-t Changes the output of the utility to show the variable values, one per line, without the variable names.

Static configuration variables

The following list describes all the variables that are part of ATF's static configuration:

atf_arch The architecture name detected by ATF. This is derived from *atf_machine* because it is a subset of it. Given that this name might be misdetected, it is provided to the user as a configuration variable so that he can fix its value temporarily until a real fix is incorporated into mainstream sources.

atf_confdir The path to the directory that contains the system-wide configuration files for ATF.

atf_libexecdir The path to the directory that contains the auxiliary utilities of ATF, used internally by the public tools.

atf_machine The machine type name detected by ATF. This should not be tunable but is provided for symmetry with *atf_arch*.

atf_pkgdatadir The path to the directory that contains the files that form the ATF's shell-scripting library.

atf_shell The path to the shell interpreter that will be used by ATF.

atf_workdir The path to the temporary directory that the utilities and the test programs will use to store temporary files in.

ENVIRONMENT

Every variable that is part of the static configuration can be overridden at run-time by defining an environment variable. This environment variable has the exact same name as the one shown by **atf-config** except that the name is all composed of uppercase letters.

The recognized environment variables are:

`ATF_ARCH`	Overrides the built-in value of *atf_arch*.
`ATF_CONFDIR`	Overrides the built-in value of *atf_confdir*.
`ATF_LIBEXECDIR`	Overrides the built-in value of *atf_libexecdir*.
`ATF_MACHINE`	Overrides the built-in value of *atf_machine*.
`ATF_PKGDATADIR`	Overrides the built-in value of *atf_pkgdatadir*.
`ATF_SHELL`	Overrides the built-in value of *atf_shell*.
`ATF_WORKDIR`	Overrides the built-in value of *atf_workdir*.

SEE ALSO

`atf`(7)

NAME

atf-exec — executes a command after modifying settings in its container process

SYNOPSIS

atf-exec -g *command*
atf-exec -h

DESCRIPTION

atf-exec is a wrapper that executes the given command after doing some additional tasks. These tasks involve modifying the properties of the process that will run the command that cannot be changed with regular system tools.

In the first synopsis form, **atf-exec** will execute the provided command after applying the changes requested by the options to its environment. If no additional options are given, the command is executed as if **atf-exec** had not been used.

In the second synopsis form, **atf-exec** will print information about all supported options and their purpose.

The following options are available:

-g Configures the process to become a new process group leader.

-h Shows a short summary of all available options and their purpose.

SEE ALSO

atf-killpg(1), atf(7)

NAME

atf-format — formats a text paragraph to fit nicely on screen

SYNOPSIS

atf-format [*str1* [*.. strN*]]
atf-format **-t** *tag* [**-l** *length*] [**-r**] [*str1* [*.. strN*]]
atf-format **-h**

DESCRIPTION

atf-format formats text messages to not overflow the terminal's width, and optionally adds a prefix string to them. Messages can be fed through the standard input or through multiple arguments; in the latter case, all of them are concatenated as if they were separated by a single space. Different lines (those separated by a new-line character) are treated as different paragraphs and thus formatted accordingly.

In the first synopsis form, **atf-format** formats the message to not overflow the terminal's width. The message is supposed to start at column 0.

In the second synopsis form, **atf-format** also formats the message to not overflow the terminal's width, but it appends a tag to the beginning of the string. This tag may or may be not repeated on each line of the text; if it is not repeated, the text is simply indented. This synopsis form is useful to easily format two-column tables, being the first one much shorter than the second one. The message is supposed to start at column 0.

In the third synopsis form, **atf-format** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-l *length* Specifies the length in characters of te tag. Useful if the tag is shorter than the desired length, which happens when formatting two-column tables.

-r Repeat the tag on each line. Otherwise it is only shown on the first one, and all others are indented appropriately using whitespace.

-t *tag* Specifies the tag to use

SEE ALSO

atf(7)

NAME

atf-killpg — sends a signal to a process group

SYNOPSIS

atf-killpg [**-s** *signal*] *pgid*
atf-killpg -h

DESCRIPTION

atf-killpg sends a signal to a process group.

In the first synopsis form, **atf-killpg** will send the specified signal to the given process group. If the signal is not provided through the **-s** flag, 'SIGTERM' will be send instead.

In the second synopsis form, **atf-killpg** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-s *signal* The numerical signal to send; defaults to 'SIGTERM'.

SEE ALSO

atf-exec(1), atf(7)

NAME

atf-report — transforms the output of atf-run to different formats

SYNOPSIS

atf-report [**-o** *fmt1:path1* [.. **-o** *fmtN:pathN*]]
atf-report -h

DESCRIPTION

atf-report reads the output of **atf-run** and transforms it to different formats. Some of these are user-friendly and others are machine-parseable, which opens a wide range of possibilities to analyze the results of a test suite's execution. See **Output formats** below for more details on which these formats are.

In the first synopsis form, **atf-report** reads the output of **atf-run** through its standard input and, if no **-o** options are given, prints a user-friendly report on its standard output using the 'ticker' format. If **-o** options are provided (more than one are allowed), they specify the complete list of reports to generate. They are all generated simulatneously, and for obvious reasons, two reports cannot be written to the same file. Note that the default output is suppressed when **-o** is provided.

In the second synopsis form, **atf-report** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-o *fmt:path* Adds a new output format. *fmt* is one of the formats described later on in **Output formats**. *path* specifies where the report will be written to. Depending on the chosen format, this may refer to a single file or to a directory. For those formats that write to a single file, specifying a '-' as the path will redirect the report to the standard output.

Output formats

The following output formats are allowed:

csv A machine-parseable Comma-Separated Values (CSV) file. This file contains the results for all test cases and test programs. Test cases are logged using the following syntax:

```
tc, test-program, test-case, result[, reason]
```

The 'result' field for test cases is always one of 'passed', 'skipped' or 'failed'. The last two are always followed by a reason.

Test programs are logged with the following syntax:

```
tp, test-program, result[, reason]
```

In this case, the 'result' can be one of: 'passed', which denotes test programs that ran without any failure; 'failed', which refers to test programs in which one or more test cases failed; or 'bogus', which mentions those test programs that failed to execute by some reason. The reason file is only available in the last case.

You should not rely on the order of the entries in the resulting output.

ticker A user-friendly report that shows the progress of the test suite's execution as it operates. This type of report should always be redirected to a virtual terminal, not a file, as it may use control sequences that will make the output unreadable in regular files.

xml A report contained in a single XML file. Ideal for later processing with `xsltproc`(1) to generate nice HTML reports.

EXAMPLES

The most simple way of running a test suite is to pipe the output of **atf-run** through **atf-report** without any additional flags. This will use the default output format, which is suitable to most users:

```
atf-run | atf-report
```

In some situations, it may be interesting to get a machine-parseable file aside from the standard report. This can be done as follows:

```
atf-run | atf-report -o csv:testsuite.csv -o ticker:-
```

Or if the standard report is not desired, thus achieving completely silent operation: atf-run | atf-report -o csv:testsuite.csv

SEE ALSO

atf-run(1), atf(7)

NAME

atf-run — executes a collection of test programs

SYNOPSIS

atf-run [**-v** *var1=value1* [.. **-v** *varN=valueN*]]
[*test_program1* [*.. test_programN*]]
atf-run -h

DESCRIPTION

atf-run executes a collection of test programs or, in other words, a complete test suite. The results of each test program are collected by the tool, and are then multiplexed into a single machine-parseable report; see atf-formats(5) for more details. This report can later be transformed into many different and saner formats using the **atf-report** tool.

The list of test programs to execute is read from an Atffile present in the current directory. This file describes the test suite stored in the directory it lives in, which aside from the list of test programs also includes meta-data and configuration variables.

atf-run is also in charge of reading the configuration files that tune the behavior of each test program and passing down the necessary variables to them. More details on how this is done are given in the **Configuration** section.

In the first synopsis form, **atf-run** parses the Atffile in the current directory and runs all the test programs specified in it. If any test program names are given as part of the command line, those are the ones executed instead of the complete list.

In the second synopsis form, **atf-run** will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-v -var=value Sets the configuration variable *var* to the given value *value*.

Configuration

atf-run reads configuration data from multiple places. After all of these places have been analyzed, a list of variable-value pairs are passed to the test programs to be run.

The following locations are scanned for configuration data, in order. Items down the list override values defined above them:

1. Configuration variables defined in the Atffile.
2. Configuration variables defined in the system-wide configuration file shared among all test suites. This lives in ${ATF_CONFDIR}/common.conf.
3. Configuration variables defined in the system-wide test-suite-specific configuration file. This lives in ${ATF_CONFDIR}/<test-suite>.conf.
4. Configuration variables defined in the user-specific configuration file shared among all test suites. This lives in ${HOME}/.atf/common.conf.
5. Configuration variables defined in the user-specific test-suite-specific configuration file. This lives in ${HOME}/.atf/<test-suite>.conf.
6. Configuration variables provided as part of the command line through the **-v** option.

The value of *ATF_CONFIG* in the above list determined as detailed in atf-config(1).

Hooks

atf-run's internal behavior can be customized by the system administrator and the user by means of hooks. These hooks are written in the shell script language for simplicity and are stored in the following files, which are read in the order provided below:

1. ${ATF_CONFDIR}/atf-run.hooks
2. ${HOME}/.atf/atf-run.hooks

The following hooks are supported:

info_start_hook — Called before **atf-run** executes any test program. The purpose of this hook is to write additional 'info' stanzas to the top of the output report; these are defined by the 'application/X-atf-tps format' described in atf-formats(1). Always use the 'atf_tps_writer_info' function to print these.

This takes no parameters.

info_end_hook — Similar to 'info_start_hook' but executed after all test programs have been run so that additional 'info' stanzas can be added to the bottom of the output report.

This takes no parameters.

All hooks are accompanied by a function named 'default_<hook_name>' that can be executed by them to invoke the default behavior built into **atf-run**. For example, in order to extend the default 'info_start_hook' hook, we could write the following function:

```
info_start_hook()
{
    default_info_start_hook "${@}"

    atf_tps_writer_info "uptime" "$(uptime)"
}
```

SEE ALSO

atf-report(1), atf-test-program(1), atf(7)

NAME

atf-test-program — common interface to ATF test programs

SYNOPSIS

atf-test-program [**-r** *fd*] [**-s** *srcdir*] [**-v** *var1=value1* [.. **-v** *varN=valueN*]]
[*test_case1* [*.. test_caseN*]]
atf-test-program -l *test_case1* [*test_case1* [*.. test_caseN*]]
atf-test-program -h

DESCRIPTION

Test programs written using the ATF libraries all share a common user interface, which is what this manual page describes.

In the first synopsis form, the test program will execute the listed test cases, or all of them if none are provided as arguments. The test cases' names can be given as glob patterns.

In the second synopsis form, the test program will list all available test cases alongside a brief description of what they do. If any arguments are specified, the list will be restricted to the matching test cases. The test cases' names can be given as glob patterns.

In the third synopsis form, the test program will print information about all supported options and their purpose.

The following options are available:

-h Shows a short summary of all available options and their purpose.

-l Lists available test cases alongside a brief description for each of them.

-r *fd* Specifies the file descriptor to which the test case will redirect its results. These results follow a machine-parseable standardized format, and are not meant for direct user consumption. See `atf-formats`(5) for more details.

-s *srcdir* The path to the directory where the test program is located. This is needed in all cases, except when the test program is being executed from the current directory. The test program will use this path to locate any helper data files or utilities.

-v *var=value* Sets the configuration variable *var* to the value *value*.

SEE ALSO

`atf-run`(1), `atf`(7)

NAME

atf-version — shows information about the installed ATF version

SYNOPSIS

atf-version
atf-version −h

DESCRIPTION

atf-version is a utility that prints information about the version of ATF currently installed in the system.

In the first synopsis form, **atf-version** shows the release version of the ATF package installed in the system (the installation that corresponds to the instance of **atf-version** being executed) and also shows information related to the repository revision used to build that package.

In the second synopsis form, **atf-version** will print information about all supported options and their purpose.

SEE ALSO

atf(7)

NAME

audioctl — control audio device

SYNOPSIS

audioctl [**-d** *device*] [**-n**] **-a**
audioctl [**-d** *device*] [**-n**] *name ...*
audioctl [**-d** *device*] [**-n**] **-w** *name=value ...*

DESCRIPTION

The **audioctl** command displays or sets various audio system driver variables. If a list of variables is present on the command line, then **audioctl** prints the current value of those variables for the specified device. If the **-a** flag is specified, all variables for the device are printed. If the **-w** flag is specified **audioctl** attempts to set the specified variables to the given values.

The **-d** flag can be used to give an alternative audio control device, the default is `/dev/audioctl0`.

The **-n** flag suppresses printing of the variable name.

ENVIRONMENT

`AUDIOCTLDEVICE` the audio control device to use.

FILES

`/dev/audio0`	audio I/O device (resets on open)
`/dev/audioctl0`	audio control device
`/dev/sound0`	audio I/O device (does not reset on open)

EXAMPLES

To set the playing sampling rate to 11025, you can use

```
audioctl -w play.sample_rate=11025
```

To set all of the play parameters for CD-quality audio, you can use

```
audioctl -w play=44100,2,16,slinear_le
```

Note that many of the variables that can be inspected and changed with **audioctl** are reset when `/dev/audio0` is opened. This can be circumvented by using `/dev/sound0` instead.

SEE ALSO

`audioplay`(1), `audiorecord`(1), `mixerctl`(1), `audio`(4), `sysctl`(8)

HISTORY

The **audioctl** command first appeared in NetBSD 1.3.

COMPATIBILITY

The old **-f** flag is still supported. This support will be removed eventually.

NAME

audioplay — play audio files

SYNOPSIS

audioplay [**-iqVh**] [**-v** *volume*] [**-b** *balance*] [**-p** *port*] [**-d** *device*][**-f** [**-c** *channels*] [**-e** *encoding*] [**-P** *precision*] [**-s** *sample-rate*]] [files ...]

DESCRIPTION

The **audioplay** program copies the named audio files, or the standard input if no files are named, to the audio device. The special name "-" is assumed to mean the standard input. The input files must contain a valid audio header, and the encoding must be understood by the underlying driver.

OPTIONS

The following options are available:

-i If the audio device cannot be opened, exit now rather than wait for it.

-q Be quiet.

-V Be verbose.

-h Print a help message.

-v Set the volume (gain) to *volume*. This value must be between 0 and 255.

-b Set the balance to *balance*. This value must be between 0 and 63.

-p Set the output port to *port*. The valid values of *port* are "speaker", "headphone" and "line".

-d Set the audio device to be *device*. The default is /dev/sound.

-f Force playing, even if the format is unknown. The **-f** flag can be used in addition with the following flags to change the number of channels, encoding, precision and sample rate.

-c when combined with the **-f** option, sets the number of channels to its argument.

-e when combined with the **-f** option, sets the encoding to its argument. Possible values are **mulaw**, **ulaw**, **alaw**, **slinear**, **linear**, **ulinear**, **adpcm**, **ADPCM**, **slinear_le**, **linear_le**, **ulinear_le**, **slinear_be**, **linear_be**, **ulinear_be**, **mpeg_l1_stream**, **mpeg_l1_packets**, **mpeg_l1_system**, **mpeg_l2_stream**, **mpeg_l2_packets**, and **mpeg_l2_system**.

-P when combined with the **-f** option, sets the precision to its argument. This value must be either 4, 8, 16, 24 or 32.

-s when combined with the **-f** option, sets the sample rate to its argument. This value must be a valid value for the audio device or an error will be returned.

ENVIRONMENT

AUDIOCTLDEVICE the audio control device to be used.

AUDIODEVICE the audio device to be used.

EXAMPLES

Play a raw dump taken from an audio CD ROM:

```
audioplay -f -c 2 -P 16 -s 44100 -e slinear_le filename
```

The audioctl(1) program can be used to show the available supported encodings:

```
audioctl encodings
```

NOTES

audioplay can be used to play Sun/NeXT audio files, and also RIFF WAVE audio files. **audioplay** can be configured in the "Netscape" web browser as the program to use when playing audio files.

ERRORS

If the audio device or the control device can not be opened, and error is returned.

If an invalid parameter is specified, an error is returned. The set of valid values for any audio parameter is specified by the hardware driver.

SEE ALSO

audioctl(1), audiorecord(1), aria(4), audio(4), audioamd(4), auich(4), autri(4), auvia(4), clcs(4), clct(4), cmpci(4), eap(4), emuxki(4), esm(4), eso(4), ess(4), fms(4), gus(4), guspnp(4), neo(4), sb(4), sv(4), wss(4), yds(4), ym(4)

HISTORY

The **audioplay** program was first seen in SunOS 5. The NetBSD **audioplay** was first made available in NetBSD 1.4.

AUTHORS

The **audioplay** program was written by Matthew R. Green ⟨mrg@eterna.com.au⟩.

NAME

audiorecord — record audio files

SYNOPSIS

audiorecord [**-afhqV**] [**-b** *balance*] [**-c** *channels*] [**-d** *device*] [**-e** *encoding*] [**-F** *format*] [**-i** *info*] [**-m** *monvol*] [**-P** *precision*] [**-p** *port*] [**-s** *rate*] [**-t** *time*] [**-v** *volume*] *file*

DESCRIPTION

The **audiorecord** program copies the audio device to the named audiofile or, if the file name is -, to the standard output.

The output file will contain either a Sun/NeXT audio header, a RIFF/WAVE audio header or no header at all. Sun output files using a linear PCM encoding are written with big-endian signed samples, possibly after converting these from little-endian or unsigned samples. RIFF/WAVE files are written in little-endian, signed samples, also converting if necessary. The default output is Sun/NeXT format, but if the output file *file* ends with a `.wav` file extension it will be written as RIFF/WAVE.

OPTIONS

The following options are available:

-a Append to the specified file, rather than overwriting.

-b *balance* Set the balance to *balance*. This value must be between 0 and 63.

-c *channels* Set number of channels to *channels*.

-d *device* Set the audio device to be *device*. The default is `/dev/sound`.

-e *encoding* Set encoding to either “alaw”, “ulaw” or “linear”, or any other value reported by **audioctl encodings**. The default encoding is “ulaw”. If the output format is “sun”, the file will contain slinear_be samples, if it is “wav”, then slinear_le, independent of the argument to **-e**. Setting the argument to **-e** still may be important since it is used in an `ioctl`(2) call to the kernel to choose the kind of data provided.

-F *format* Set the output header format to *format*. Currently supported formats are “sun”, “wav”, and “none” for Sun/NeXT audio, WAV, and no header, respectively.

-f Force. Normally when appending to audiofiles using the **-a** option, the sample rates must match. The **-f** option will allow a discrepancy to be ignored.

-h Print a help message.

-i *info* If supported by the **-F** format, add the string *info* to the output header.

-m *monvol* Set the monitor volume.

-P *precision* Set the precision. This value is the number of bits per sample, and is normally either “8” or “16”, though the values “4”, “24”, and “32” are also valid.

-p *port* Set the input port to *port*. The valid values of *port* are “cd”, “internal-cd”, “mic”, and “line”.

-s *rate* Set the sampling rate. This value is per-second. Typical values are 8000, 44100, and 48000, which are the telephone, CD Audio, and DAT Audio default sampling rates.

-t *time* Sets the maximum amount of time to record. Format is [hh:]mm:ss.

-q Be quiet.

-V Be verbose.

-v *volume* Set the volume (gain) to *volume*. This value must be between 0 and 255.

ENVIRONMENT

AUDIOCTLDEVICE
the audio control device to be used.

AUDIODEVICE the audio device to be used.

SEE ALSO

audioctl(1), audioplay(1), aria(4), audio(4), audioamd(4), auich(4), autri(4), auvia(4), clcs(4), clct(4), cmpci(4), eap(4), emuxki(4), esm(4), eso(4), ess(4), fms(4), gus(4), guspnp(4), neo(4), sb(4), sv(4), wss(4), yds(4), ym(4)

HISTORY

The **audiorecord** program was first seen in SunOS 5. It was first made available in NetBSD 1.4. RIFF/WAVE support, and support for converting signed/unsigned and big/little-endian samples was first made available in NetBSD 1.6.

AUTHORS

The **audiorecord** program was written by Matthew R. Green ⟨mrg@eterna.com.au⟩.

BUGS

WAV big-endian samples are converted to little-endian, rather than a RIFX header being written.

NAME

autoconf − Generate configuration scripts

SYNOPSIS

autoconf [*OPTION*] ... [*TEMPLATE-FILE*]

DESCRIPTION

Generate a configuration script from a TEMPLATE-FILE if given, or ‘configure.ac’ if present, or else ‘configure.in’. Output is sent to the standard output if TEMPLATE-FILE is given, else into ‘configure’.

Operation modes:

−h, **−−help**
print this help, then exit

−V, **−−version**
print version number, then exit

−v, **−−verbose**
verbosely report processing

−d, **−−debug**
don’t remove temporary files

−o, **−−output**=*FILE*
save output in FILE (stdout is the default)

−W, **−−warnings**=*CATEGORY*
report the warnings falling in CATEGORY [syntax]

Warning categories include:

‘cross’ cross compilation issues

‘obsolete’
obsolete constructs

‘syntax’
dubious syntactic constructs

‘all’ all the warnings

‘no-CATEGORY’
turn off the warnings on CATEGORY

‘none’ turn off all the warnings

‘error’ warnings are error

The environment variable ‘WARNINGS’ is honored.

Library directories:

−A, **−−autoconf−dir**=*ACDIR*
Autoconf’s macro files location (rarely needed)

−l, **−−localdir**=*DIR*
location of the ‘aclocal.m4’ file

Tracing:

−t, **−−trace**=*MACRO*
report the list of calls to MACRO

−i, **−−initialization**
also trace Autoconf’s initialization process

In tracing mode, no configuration script is created.

AUTHOR

Written by David J. MacKenzie.

REPORTING BUGS

Report bugs to <bug-autoconf@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

The full documentation for **autoconf** is maintained as a Texinfo manual. If the **info** and **autoconf** programs are properly installed at your site, the command

info autoconf

should give you access to the complete manual.

NAME

autoheader − Create a template header for configure

SYNOPSIS

autoheader [*OPTION*] ... [*TEMPLATE-FILE*]

DESCRIPTION

Create a template file of C ‘#define’ statements for ‘configure’ to use. To this end, scan TEMPLATE-FILE, or ‘configure.ac’ if present, or else ‘configure.in’.

−h, **−−help**
print this help, then exit

−V, **−−version**
print version number, then exit

−v, **−−verbose**
verbosely report processing

−d, **−−debug**
don’t remove temporary files

−W, **−−warnings**=*CATEGORY*
report the warnings falling in CATEGORY

Warning categories include:

‘obsolete’
obsolete constructs

‘all’ all the warnings

‘no-CATEGORY’
turn off the warnings on CATEGORY

‘none’ turn off all the warnings

‘error’ warnings are error

Library directories:

−A, **−−autoconf−dir**=*ACDIR*
Autoconf’s macro files location (rarely needed)

−l, **−−localdir**=*DIR*
location of ‘aclocal.m4’ and ‘acconfig.h’

AUTHOR

Written by Roland McGrath.

REPORTING BUGS

Report bugs to <bug-autoconf@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

The full documentation for **autoheader** is maintained as a Texinfo manual. If the **info** and **autoheader** programs are properly installed at your site, the command

info autoheader

should give you access to the complete manual.

NAME

autopoint − copies standard gettext infrastructure

SYNOPSIS

autopoint [*OPTION*]...

DESCRIPTION

Copies standard gettext infrastructure files into a source package.

OPTIONS

−−help print this help and exit

−−version
print version information and exit

−f, **−−force**
force overwriting of files that already exist

−n, **−−dry−run**
print modifications but don't perform them

AUTHOR

Written by Bruno Haible

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **autopoint** is maintained as a Texinfo manual. If the **info** and **autopoint** programs are properly installed at your site, the command

info autopoint

should give you access to the complete manual.

NAME

autoreconf − Update generated configuration files

SYNOPSIS

autoreconf [*OPTION*] ... [*TEMPLATE-FILE*]

DESCRIPTION

Run ‘autoconf’ (and ‘autoheader’, ‘aclocal’ and ‘automake’, where appropriate) repeatedly to remake the Autoconf ‘configure’ scripts and configuration header templates in the directory tree rooted at the current directory. By default, it only remakes those files that are older than their predecessors. If you install a new version of Autoconf, running ‘autoreconf’ remakes all of the files by giving it the ‘--force’ option.

Operation modes:

−h, **−−help**
print this help, then exit

−V, **−−version**
print version number, then exit

−v, **−−verbose**
verbosely report processing

−d, **−−debug**
don’t remove temporary files

−f, **−−force**
consider every files are obsolete

−i, **−−install**
copy missing auxiliary files

−s, **−−symlink**
instead of copying, install symbolic links

The option ‘--install’ is similar to the option ‘--add-missing’ in other tools.

Library directories:

−A, **−−autoconf−dir**=*ACDIR*
Autoconf’s macro files location (rarely needed)

−l, **−−localdir**=*DIR*
location of ‘aclocal.m4’ and ‘acconfig.h’

−M, **−−m4dir**=*M4DIR*
this package’s Autoconf extensions

Unless specified, heuristics try to compute ‘M4DIR’ from the ‘Makefile.am’, or defaults to ‘m4’ if it exists.

The following options are passed to ‘automake’:

−−cygnus
assume program is part of Cygnus-style tree

−−foreign
set strictness to foreign

−−gnits
set strictness to gnits

−−gnu set strictness to gnu

−−include−deps
include generated dependencies in Makefile.in

The environment variables AUTOCONF, AUTOHEADER, AUTOMAKE, and ACLOCAL are honored.

AUTHOR

Written by David J. MacKenzie.

REPORTING BUGS

Report bugs to <bug-autoconf@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

The full documentation for **autoreconf** is maintained as a Texinfo manual. If the **info** and **autoreconf** programs are properly installed at your site, the command

info autoreconf

should give you access to the complete manual.

NAME

autoscan − Generate a preliminary configure.in

SYNOPSIS

autoscan [*OPTION*] ... [*SRCDIR*]

DESCRIPTION

Examine source files in the directory tree rooted at SRCDIR, or the current directory if none is given. Search the source files for common portability problems, check for incompleteness of ‘configure.ac’, and create a file ‘configure.scan’ which is a preliminary ‘configure.ac’ for that package.

−h, **−−help**
print this help, then exit

−V, **−−version**
print version number, then exit

−v, **−−verbose**
verbosely report processing

Library directories:

−A, **−−autoconf−dir**=*ACDIR*
Autoconf’s files location (rarely needed)

−l, **−−localdir**=*DIR*
location of ‘aclocal.m4’ and ‘acconfig.h’

AUTHOR

Written by David J. MacKenzie.

REPORTING BUGS

Report bugs to <bug-autoconf@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

The full documentation for **autoscan** is maintained as a Texinfo manual. If the **info** and **autoscan** programs are properly installed at your site, the command

info autoscan

should give you access to the complete manual.

NAME

autoupdate − Update a configure.in to a newer Autoconf

SYNOPSIS

autoupdate [*OPTION*] ... [*TEMPLATE-FILE*...]

DESCRIPTION

Update the TEMPLATE-FILE... if given, or ‘configure.ac’ if present, or else ‘configure.in’, to the syntax of the current version of Autoconf. The original files are backed up.

Operation modes:

−h, **−−help**
print this help, then exit

−V, **−−version**
print version number, then exit

−v, **−−verbose**
verbosely report processing

−d, **−−debug**
don’t remove temporary files

Library directories:

−A, **−−autoconf−dir**=*ACDIR*
Autoconf’s macro files location (rarely needed)

−l, **−−localdir**=*DIR*
location of ‘aclocal.m4’

Environment variables:

M4 GNU M4 1.4 or above

AUTOCONF
autoconf 2.52

AUTHOR

Written by David J. MacKenzie and Akim Demaille.

REPORTING BUGS

Report bugs to <bug-autoconf@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

The full documentation for **autoupdate** is maintained as a Texinfo manual. If the **info** and **autoupdate** programs are properly installed at your site, the command

info autoupdate

should give you access to the complete manual.

NAME

awk − pattern-directed scanning and processing language

SYNOPSIS

awk [**−F** *fs*] [**−v** *var=value*] [*'prog'* | **−f** *progfile*] [*file ...*]

DESCRIPTION

Awk scans each input *file* for lines that match any of a set of patterns specified literally in *prog* or in one or more files specified as **−f** *progfile*. With each pattern there can be an associated action that will be performed when a line of a *file* matches the pattern. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern. The file name **−** means the standard input. Any *file* of the form *var=value* is treated as an assignment, not a filename, and is executed at the time it would have been opened if it were a filename. The option **−v** followed by *var=value* is an assignment to be done before *prog* is executed; any number of **−v** options may be present. The **−F** *fs* option defines the input field separator to be the regular expression *fs*.

An input line is normally made up of fields separated by white space, or by regular expression **FS**. The fields are denoted **$1**, **$2**, ..., while **$0** refers to the entire line. If **FS** is null, the input line is split into one field per character.

A pattern-action statement has the form

pattern **{** *action* **}**

A missing **{** *action* **}** means print the line; a missing pattern always matches. Pattern-action statements are separated by newlines or semicolons.

An action is a sequence of statements. A statement can be one of the following:

```
if( expression ) statement [ else statement ]
while( expression ) statement
for( expression ; expression ; expression ) statement
for( var in array ) statement
do statement while( expression )
break
continue
{ [ statement ... ] }
expression                          # commonly var = expression
print [ expression-list ] [ > expression ]
printf format [ , expression-list ] [ > expression ]
return [ expression ]
next                                # skip remaining patterns on this input line
nextfile                            # skip rest of this file, open next, start at top
delete array[ expression ]          # delete an array element
delete array                        # delete all elements of array
exit [ expression ]                 # exit immediately; status is expression
```

Statements are terminated by semicolons, newlines or right braces. An empty *expression-list* stands for **$0**. String constants are quoted **" "**, with the usual C escapes recognized within. Expressions take on string or numeric values as appropriate, and are built using the operators **+ − * / % ^** (exponentiation), and concatenation (indicated by white space). The operators **! ++ −− += −= *= /= %= ^= > >= < <= == != ?:** are also available in expressions. Variables may be scalars, array elements (denoted *x***[***i***]**) or fields. Variables are initialized to the null string. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. Multiple subscripts such as **[i,j,k]** are permitted; the constituents are concatenated, separated by the value of **SUBSEP**.

The **print** statement prints its arguments on the standard output (or on a file if **>** *file* or **>>** *file* is present or on a pipe if **|***cmd* is present), separated by the current output field separator, and terminated by the output record separator. *file* and *cmd* may be literal names or parenthesized expressions; identical string values in different statements denote the same open file. The **printf** statement formats its expression list according to the format (see *printf*(3)). The built-in function **close(***expr***)** closes the file or pipe *expr*. The built-in function **fflush(***expr***)** flushes any buffered output for the file or pipe *expr*.

The mathematical functions **exp**, **log**, **sqrt**, **sin**, **cos**, and **atan2** are built in. Other built-in functions:

length the length of its argument taken as a string, or of **$0** if no argument.

rand random number on (0,1)

srand sets seed for **rand** and returns the previous seed.

int truncates to an integer value

substr(*s***,** *m***,** *n***)**
the *n*-character substring of *s* that begins at position *m* counted from 1.

index(*s***,** *t***)**
the position in *s* where the string *t* occurs, or 0 if it does not.

match(*s***,** *r***)**
the position in *s* where the regular expression *r* occurs, or 0 if it does not. The variables **RSTART** and **RLENGTH** are set to the position and length of the matched string.

split(*s***,** *a***,** *fs***)**
splits the string *s* into array elements *a***[1]**, *a***[2]**, ..., *a***[***n***]**, and returns *n*. The separation is done with the regular expression *fs* or with the field separator **FS** if *fs* is not given. An empty string as field separator splits the string into one array element per character.

sub(*r***,** *t***,** *s***)**
substitutes *t* for the first occurrence of the regular expression *r* in the string *s*. If *s* is not given, **$0** is used.

gsub same as **sub** except that all occurrences of the regular expression are replaced; **sub** and **gsub** return the number of replacements.

sprintf(*fmt***,** *expr***,** ... **)**
the string resulting from formatting *expr* ... according to the *printf* (3) format *fmt*

system(*cmd***)**
executes *cmd* and returns its exit status

tolower(*str***)**
returns a copy of *str* with all upper-case characters translated to their corresponding lower-case equivalents.

toupper(*str***)**
returns a copy of *str* with all lower-case characters translated to their corresponding upper-case equivalents.

The ''function'' **getline** sets **$0** to the next input record from the current input file; **getline** < *file* sets **$0** to the next record from *file*. **getline** *x* sets variable *x* instead. Finally, *cmd* | **getline** pipes the output of *cmd* into **getline**; each call of **getline** returns the next line of output from *cmd*. In all cases, **getline** returns 1 for a successful input, 0 for end of file, and −1 for an error.

Patterns are arbitrary Boolean combinations (with **!** || **&&**) of regular expressions and relational expressions. Regular expressions are as in *egrep*; see *grep*(1). Isolated regular expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions, using the operators **~** and **!~**. */re/* is a constant regular expression; any string (constant or variable) may be used as a regular expression, except in the position of an isolated regular expression in a pattern.

A pattern may consist of two patterns separated by a comma; in this case, the action is performed for all lines from an occurrence of the first pattern though an occurrence of the second.

A relational expression is one of the following:

expression matchop regular-expression
expression relop expression
expression **in** *array-name*
(*expr***,***expr*,...**) in** *array-name*

where a relop is any of the six relational operators in C, and a matchop is either **~** (matches) or **!~** (does not

match). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of these.

The special patterns **BEGIN** and **END** may be used to capture control before the first input line is read and after the last. **BEGIN** and **END** do not combine with other patterns.

Variable names with special meanings:

CONVFMT
conversion format used when converting numbers (default **%.6g**)

FS regular expression used to separate fields; also settable by option **−F** *fs*.

NF number of fields in the current record

NR ordinal number of the current record

FNR ordinal number of the current record in the current file

FILENAME
the name of the current input file

RS input record separator (default newline)

OFS output field separator (default blank)

ORS output record separator (default newline)

OFMT output format for numbers (default **%.6g**)

SUBSEP
separates multiple subscripts (default 034)

ARGC argument count, assignable

ARGV argument array, assignable; non-null members are taken as filenames

ENVIRON
array of environment variables; subscripts are names.

Functions may be defined (at the position of a pattern-action statement) thus:

function foo(a, b, c) { ...; return x }

Parameters are passed by value if scalar and by reference if array name; functions may be called recursively. Parameters are local to the function; all other variables are global. Thus local variables may be created by providing excess parameters in the function definition.

EXAMPLES

```
length($0) > 72
```
Print lines longer than 72 characters.

```
{ print $2, $1 }
```
Print first two fields in opposite order.

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
      { print $2, $1 }
```
Same, with input fields separated by comma and/or blanks and tabs.

```
      { s += $1 }
END   { print "sum is", s, " average is", s/NR }
```
Add up first column, print sum and average.

```
/start/, /stop/
```
Print all lines between start/stop pairs.

```
BEGIN {     # Simulate echo(1)
      for (i = 1; i < ARGC; i++) printf "%s ", ARGV[i]
      printf "\n"
```

```
        exit }
```

SEE ALSO

lex(1), *sed*(1)

A. V. Aho, B. W. Kernighan, P. J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988. ISBN 0-201-07981-X

BUGS

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number add 0 to it; to force it to be treated as a string concatenate `""` to it.

The scope rules for variables in functions are a botch; the syntax is worse.

NAME

awk — pattern-directed scanning and processing language

SYNOPSIS

awk [**-F** *fs*] [**-v** *var=value*] [**-safe**] [**-d**[*N*]] [*prog* | **-f** *filename*] *file ...*
awk -V

DESCRIPTION

awk is the Bell Labs' implementation of the AWK programming language as described in the *The AWK Programming Language* by A. V. Aho, B. W. Kernighan, and P. J. Weinberger.

awk scans each input *file* for lines that match any of a set of patterns specified literally in *prog* or in one or more files specified as **-f** *filename*. With each pattern there can be an associated action that will be performed when a line of a matches the pattern. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern. The file name - means the standard input. Any *file* of the form *var=value* is treated as an assignment, not a filename, and is executed at the time it would have been opened if it were a filename.

The options are as follows:

-d[*N*] Set debug level to specified number *N*. If the number is omitted, debug level is set to 1.

-f *filename*
Read the AWK program source from specified file *filename*, instead of the first command line argument. Multiple **-f** options may be specified.

-F *fs* Set the input field separator *FS* to the regular expression *fs*.

-mr *NNN*, **-mf** *NNN*
Obsolete, no longer needed options. Set limit on maximum record or fields number.

-safe Potentially unsafe functions such as **system**() make the program abort (with a warning message).

-v *var=value*
Assign the value *value* to the variable *var* before *prog* is executed. Any number of **-v** options may be present.

-V Print **awk** version on standard output and exit.

An input line is normally made up of fields separated by white space, or by regular expression *FS*. The fields are denoted *$1*, *$2*, ..., while *$0* refers to the entire line. If *FS* is null, the input line is split into one field per character.

A pattern-action statement has the form

```
pattern { action }
```

A missing { action } means print the line; a missing pattern always matches. Pattern-action statements are separated by newlines or semicolons.

An action is a sequence of statements. Statements are terminated by semicolons, newlines or right braces. An empty *expression-list* stands for *$0*. String constants are quoted " ", with the usual C escapes recognized within. Expressions take on string or numeric values as appropriate, and are built using the **Operators** (see next subsection). Variables may be scalars, array elements (denoted *x[i]*) or fields. Variables are initialized to the null string. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. Multiple subscripts such as *[i,j,k]* are permitted; the constituents are concatenated, separated by the value of *SUBSEP*.

Operators

awk operators, in order of decreasing precedence, are:

(...) Grouping
$ Field reference
++ -- Increment and decrement, can be used either as postfix or prefix.
^ Exponentiation (the ** form is also supported, and ****=** for the assignment operator).
+ - ! Unary plus, unary minus and logical negation.
* / % Multiplication, division and modulus.
+ - Addition and subtraction.
space String concatenation.
< >
≤ ≥
!= == Regular relational operators
~ !~ Regular expression match and not match
in Array membership
&& Logical AND
|| Logical OR
?: C conditional expression. This is used as *expr1* ? *expr2* : *expr3* . If *expr1* is true, the result value is *expr2*, otherwise it is *expr3*. Only one of *expr2* and *expr3* is evaluated.
= += -=
***= /= %= ^=**
Assignment and Operator-Assignment

Control Statements

The control statements are as follows:

if (*expression* **)** *statement* [**else** *statement*]
while(*expression* **)** *statement*
for(*expression* **;** *expression* **;** *expression* **)** *statement*
for(*var* **in** *array* **)** *statement*
do *statement* **while(** *expression* **)**
break
continue
delete *array*[*expression*]
delete *array*
exit [*expression*] *expression*
return [*expression*]
{ [*statement ...*] **}**

I/O Statements

The input/output statements are as follows:

close(*expr*)
Closes the file or pipe *expr*. Returns zero on success; otherwise nonzero.

fflush(*expr*)
Flushes any buffered output for the file or pipe *expr*. Returns zero on success; otherwise nonzero.

getline [*var*]
Set *var* (or *$0 if var* is not specified) to the next input record from the current input file. **getline** returns 1 for a successful input, 0 for end of file, and −1 for an error.

getline [*var*] **<** *file*
Set *var* (or *$0 if var* is not specified) to the next input record from the specified file *file*.

expr **| getline**
Pipes the output of *expr* into **getline**; each call of **getline** returns the next line of output from *expr*.

next Skip remaining patterns on this input line.

nextfile
Skip rest of this file, open next, start at top.

print [*expr-list*] [**>** *file*]
The **print** statement prints its arguments on the standard output (or to a file if **> file** or to a pipe if | *expr* is present), separated by the current output field separator *OFS*, and terminated by the output record separator *ORS*. Both *file* and *expr* may be literal names or parenthesized expressions; identical string values in different statements denote the same open file.

printf *format* [, *expr-list*] [**>** *file*]
Format and print its expression list according to *format*. See printf(3) for list of supported formats and their meaning.

Mathematical and Numeric Functions

AWK has the following mathematical and numerical functions built-in:

atan2(*x*, *y*)
Returns the arctangent of *x / y* in radians. See also atan2(3).

cos(*expr*)
Computes the cosine of *expr*, measured in radians. See also cos(3).

exp(*expr*)
Computes the exponential value of the given argument *expr*. See also exp(3).

int(*expr*)
Truncates *expr* to integer.

log(*expr*)
Computes the value of the natural logarithm of argument *expr*. See also log(3).

rand() Returns random number between 0 and 1.

sin(*expr*)
Computes the sine of *expr*, measured in radians. See also sin(3).

sqrt(*expr*)
Computes the non-negative square root of *expr*. See also sqrt(3).

srand(*[expr]*)
Sets seed for random number generator (**rand**()) and returns the previous seed.

String Functions

AWK has the following string functions built-in:

gensub(*r*, *s*, *h*, *[t]*)
Search the target string *t* for matches of the regular expression *r*. If *h* is a string beginning with **g** or **G**, then replace all matches of *r* with *s*. Otherwise, *h* is a number indicating which match of *r* to replace. If no *t* is supplied, *$0* is used instead. Unlike **sub**() and **gsub**(), the modified string is returned as the result of the function, and the original target is *not* changed. Note that the \n

sequences within replacement string *s* supported by GNU **awk** are *not* supported at this moment.

gsub(*r*, *t*, *[s]*)
same as **sub**() except that all occurrences of the regular expression are replaced; **sub**() and **gsub**() return the number of replacements.

index(*s*, *t*)
the position in *s* where the string *t* occurs, or 0 if it does not.

length(*[string]*)
the length of its argument taken as a string, or of *$0* if no argument.

match(*s*, *r*)
the position in *s* where the regular expression *r* occurs, or 0 if it does not. The variables *RSTART* and *RLENGTH* are set to the position and length of the matched string.

split(*s*, *a*, *[fs]*)
splits the string *s* into array elements *a[1]*, *a[2]*, ..., *a[n]*, and returns *n*. The separation is done with the regular expression *fs* or with the field separator *FS* if *fs* is not given. An empty string as field separator splits the string into one array element per character.

sprintf(*fmt*, *expr*, *...*)
Returns the string resulting from formatting *expr* according to the printf(3) format *fmt*.

sub(*r*, *t*, *[s]*)
substitutes *t* for the first occurrence of the regular expression *r* in the string *s*. If *s* is not given, *$0* is used.

substr(*s*, *m*, *[n]*)
Returns the at most *n*-character substring of *s* starting at position *m*, counted from 1. If *n* is omitted, the rest of *s* is returned.

tolower(*str*)
returns a copy of *str* with all upper-case characters translated to their corresponding lower-case equivalents.

toupper(*str*)
returns a copy of *str* with all lower-case characters translated to their corresponding upper-case equivalents.

Time Functions

This **awk** provides the following two functions for obtaining time stamps and formatting them:

systime()
Returns the value of time in seconds since the start of Unix Epoch (Midnight, January 1, 1970, Coordinated Universal Time). See also time(3).

strftime(*[format [, timestamp]]*)
Formats the time *timestamp* according to the string *format*. *timestamp* should be in same form as value returned by **systime**(). If *timestamp* is missing, current time is used. If *format* is missing, a default format equivalent to the output of date(1) would be used. See the specification of ANSI C strftime(3) for the format conversions which are supported.

Other built-in functions

system(*cmd*)
executes *cmd* and returns its exit status

Patterns

Patterns are arbitrary Boolean combinations (with **! || &&**) of regular expressions and relational expressions. Regular expressions are as in egrep(1). Isolated regular expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions, using the operators ~ and **!~**. **/ re /** is a constant regular expression; any string (constant or variable) may be used as a regular expression, except in the position of an isolated regular expression in a pattern.

A pattern may consist of two patterns separated by a comma; in this case, the action is performed for all lines from an occurrence of the first pattern though an occurrence of the second.

A relational expression is one of the following:

expression matchop regular-expression
expression relop expression
expression **in** *array-name*
(*expr, expr,...* **) in** *array-name*

where a *relop* is any of the six relational operators in C, and a *matchop* is either ~ (matches) or **!~** (does not match). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of these.

The special patterns **BEGIN** and **END** may be used to capture control before the first input line is read and after the last. **BEGIN** and **END** do not combine with other patterns.

Built-in Variables

Variable names with special meanings:

ARGC	argument count, assignable
ARGV	argument array, assignable; non-null members are taken as filenames
CONVFMT	conversion format used when converting numbers (default "%.6g")
ENVIRON	array of environment variables; subscripts are names.
FILENAME	the name of the current input file
FNR	ordinal number of the current record in the current file
FS	regular expression used to separate fields; also settable by option **-F** *fs*.
NF	number of fields in the current record
NR	ordinal number of the current record
OFMT	output format for numbers (default "%.6g")
OFS	output field separator (default blank)
ORS	output record separator (default newline)
RS	input record separator (default newline)
RSTART	Position of the first character matched by **match**(); 0 if not match.
RLENGTH	Length of the string matched by **match**(); -1 if no match.
SUBSEP	separates multiple subscripts (default 034)

Functions

Functions may be defined (at the position of a pattern-action statement) thus:

```
function foo(a, b, c) { ...; return x }
```

Parameters are passed by value if scalar and by reference if array name; functions may be called recursively. Parameters are local to the function; all other variables are global. Thus local variables may be created by providing excess parameters in the function definition.

EXAMPLES

```
length($0) > 72
```
Print lines longer than 72 characters.

```
{ print $2, $1 }
```
Print first two fields in opposite order.

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
      { print $2, $1 }
```
Same, with input fields separated by comma and/or blanks and tabs.

```
    { s += $1 }
END { print "sum is", s, " average is ", s/NR }
```
Add up first column, print sum and average.

```
/start/, /stop/
```
Print all lines between start/stop pairs.

```
BEGIN { # Simulate echo(1)
     for (i = 1; i < ARGC; i++) printf "%s ", ARGV[i]
     printf "\n"
     exit }
```

SEE ALSO

`egrep`(1), `lex`(1), `sed`(1), `atan2`(3), `cos`(3), `exp`(3), `log`(3), `sin`(3), `sqrt`(3), `strftime`(3), `time`(3)

A. V. Aho, B. W. Kernighan, P. J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988. ISBN 0-201-07981-X

AWK Language Programming, Edition 1.0, published by the Free Software Foundation, 1995

HISTORY

nawk has been the default system **awk** since NetBSD 2.0, replacing the previously used GNU **awk**.

BUGS

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number add 0 to it; to force it to be treated as a string concatenate "" to it.
The scope rules for variables in functions are a botch; the syntax is worse.

NAME

banner — print strings in large letters

SYNOPSIS

banner [**-f** *fg*] [**-b** *bg*] [**-l**] *string ...*

DESCRIPTION

banner prints up to 10 chars of each *string* in large letters on the standard output.

The options are:

-b *bg*
: Use *bg* as the background color (character) rather than the default " ".

-f *fg*
: Use *fg* as the foreground color (character) rather than the default "#". If *fg* is "-", then the actual letter being printed is used as the foreground color in the same way that LPD does it. The results are quite variable.

-l
: Use the more attractive LPD font instead of the traditional **banner** font.

SEE ALSO

banner(6)

NAME

basename, **dirname** — return filename or directory portion of pathname

SYNOPSIS

basename *string* [*suffix*]
dirname *string*

DESCRIPTION

basename deletes any prefix ending with the last slash '/' character present in *string*, and a *suffix*, if given. The resulting filename is written to the standard output. A non-existent suffix is ignored.

dirname deletes the filename portion, beginning with the last slash '/' character to the end of *string*, and writes the result to the standard output.

EXIT STATUS

Both the **basename** and **dirname** utilities exit 0 on success, and >0 if an error occurs.

EXAMPLES

The following line sets the shell variable FOO to /usr/bin.

```
FOO=‘dirname /usr/bin/trail‘
```

SEE ALSO

csh(1), sh(1), basename(3), dirname(3)

STANDARDS

The **basename** and **dirname** utilities conform to IEEE Std 1003.2-1992 ("POSIX.2").

NAME

bc - An arbitrary precision calculator language

SYNTAX

bc [**-hlwsqv**] [long-options] [*file* ...]

VERSION

This man page documents GNU bc version 1.06.

DESCRIPTION

bc is a language that supports arbitrary precision numbers with interactive execution of statements. There are some similarities in the syntax to the C programming language. A standard math library is available by command line option. If requested, the math library is defined before processing any files. **bc** starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, **bc** reads from the standard input. All code is executed as it is read. (If a file contains a command to halt the processor, **bc** will never read from the standard input.)

This version of **bc** contains several extensions beyond traditional **bc** implementations and the POSIX draft standard. Command line options can cause these extensions to print a warning or to be rejected. This document describes the language accepted by this processor. Extensions will be identified as such.

OPTIONS

-h, --help
: Print the usage and exit.

-i, --interactive
: Force interactive mode.

-l, --mathlib
: Define the standard math library.

-w, --warn
: Give warnings for extensions to POSIX **bc**.

-s, --standard
: Process exactly the POSIX **bc** language.

-q, --quiet
: Do not print the normal GNU bc welcome.

-v, --version
: Print the version number and copyright and quit.

NUMBERS

The most basic element in **bc** is the number. Numbers are arbitrary precision numbers. This precision is both in the integer part and the fractional part. All numbers are represented internally in decimal and all computation is done in decimal. (This version truncates results from divide and multiply operations.) There are two attributes of numbers, the length and the scale. The length is the total number of significant decimal digits in a number and the scale is the total number of decimal digits after the decimal point. For example:

.000001 has a length of 6 and scale of 6.
1935.000 has a length of 7 and a scale of 3.

VARIABLES

Numbers are stored in two types of variables, simple variables and arrays. Both simple variables and array variables are named. Names begin with a letter followed by any number of letters, digits and underscores. All letters must be lower case. (Full alpha-numeric names are an extension. In POSIX **bc** all names are a single lower case letter.) The type of variable is clear by the context because all array variable names will be followed by brackets ([]).

There are four special variables, **scale, ibase, obase,** and **last**. **scale** defines how some operations use digits after the decimal point. The default value of **scale** is 0. **ibase** and **obase** define the conversion base for input and output numbers. The default for both input and output is base 10. **last** (an extension) is a

variable that has the value of the last printed number. These will be discussed in further detail where appropriate. All of these variables may have values assigned to them as well as used in expressions.

COMMENTS

Comments in **bc** start with the characters **/*** and end with the characters ***/**. Comments may start anywhere and appear as a single space in the input. (This causes comments to delimit other input items. For example, a comment can not be found in the middle of a variable name.) Comments include any newlines (end of line) between the start and the end of the comment.

To support the use of scripts for **bc**, a single line comment has been added as an extension. A single line comment starts at a **#** character and continues to the next end of the line. The end of line character is not part of the comment and is processed normally.

EXPRESSIONS

The numbers are manipulated by expressions and statements. Since the language was designed to be interactive, statements and expressions are executed as soon as possible. There is no "main" program. Instead, code is executed as it is encountered. (Functions, discussed in detail later, are defined when encountered.)

A simple expression is just a constant. **bc** converts constants into internal decimal numbers using the current input base, specified by the variable **ibase**. (There is an exception in functions.) The legal values of **ibase** are 2 through 16. Assigning a value outside this range to **ibase** will result in a value of 2 or 16. Input numbers may contain the characters 0-9 and A-F. (Note: They must be capitals. Lower case letters are variable names.) Single digit numbers always have the value of the digit regardless of the value of **ibase**. (i.e. A = 10.) For multi-digit numbers, **bc** changes all input digits greater or equal to ibase to the value of **ibase**-1. This makes the number **FFF** always be the largest 3 digit number of the input base.

Full expressions are similar to many other high level languages. Since there is only one kind of number, there are no rules for mixing types. Instead, there are rules on the scale of expressions. Every expression has a scale. This is derived from the scale of original numbers, the operation performed and in many cases, the value of the variable **scale**. Legal values of the variable **scale** are 0 to the maximum number representable by a C integer.

In the following descriptions of legal expressions, "expr" refers to a complete expression and "var" refers to a simple or an array variable. A simple variable is just a

name

and an array variable is specified as

name[*expr*]

Unless specifically mentioned the scale of the result is the maximum scale of the expressions involved.

- expr The result is the negation of the expression.

++ var The variable is incremented by one and the new value is the result of the expression.

-- var The variable is decremented by one and the new value is the result of the expression.

var ++ The result of the expression is the value of the variable and then the variable is incremented by one.

var -- The result of the expression is the value of the variable and then the variable is decremented by one.

expr + expr

The result of the expression is the sum of the two expressions.

expr - expr

The result of the expression is the difference of the two expressions.

expr * expr

The result of the expression is the product of the two expressions.

expr / expr

The result of the expression is the quotient of the two expressions. The scale of the result is the value of the variable **scale**.

expr % expr
: The result of the expression is the "remainder" and it is computed in the following way. To compute a%b, first a/b is computed to **scale** digits. That result is used to compute a-(a/b)*b to the scale of the maximum of **scale**+scale(b) and scale(a). If **scale** is set to zero and both expressions are integers this expression is the integer remainder function.

expr ^ expr
: The result of the expression is the value of the first raised to the second. The second expression must be an integer. (If the second expression is not an integer, a warning is generated and the expression is truncated to get an integer value.) The scale of the result is **scale** if the exponent is negative. If the exponent is positive the scale of the result is the minimum of the scale of the first expression times the value of the exponent and the maximum of **scale** and the scale of the first expression. (e.g. scale(a^b) = min(scale(a)*b, max(**scale,** scale(a))).) It should be noted that expr^0 will always return the value of 1.

(expr) This alters the standard precedence to force the evaluation of the expression.

var = expr
: The variable is assigned the value of the expression.

var <op>= expr
: This is equivalent to "var = var <op> expr" with the exception that the "var" part is evaluated only once. This can make a difference if "var" is an array.

Relational expressions are a special kind of expression that always evaluate to 0 or 1, 0 if the relation is false and 1 if the relation is true. These may appear in any legal expression. (POSIX bc requires that relational expressions are used only in if, while, and for statements and that only one relational test may be done in them.) The relational operators are

expr1 < expr2
: The result is 1 if expr1 is strictly less than expr2.

expr1 <= expr2
: The result is 1 if expr1 is less than or equal to expr2.

expr1 > expr2
: The result is 1 if expr1 is strictly greater than expr2.

expr1 >= expr2
: The result is 1 if expr1 is greater than or equal to expr2.

expr1 == expr2
: The result is 1 if expr1 is equal to expr2.

expr1 != expr2
: The result is 1 if expr1 is not equal to expr2.

Boolean operations are also legal. (POSIX **bc** does NOT have boolean operations). The result of all boolean operations are 0 and 1 (for false and true) as in relational expressions. The boolean operators are:

!expr The result is 1 if expr is 0.

expr && expr
: The result is 1 if both expressions are non-zero.

expr || expr
: The result is 1 if either expression is non-zero.

The expression precedence is as follows: (lowest to highest)
: || operator, left associative
: && operator, left associative
: ! operator, nonassociative
: Relational operators, left associative
: Assignment operator, right associative

+ and - operators, left associative
*, / and % operators, left associative
^ operator, right associative
unary - operator, nonassociative
++ and -- operators, nonassociative

This precedence was chosen so that POSIX compliant **bc** programs will run correctly. This will cause the use of the relational and logical operators to have some unusual behavior when used with assignment expressions. Consider the expression:

a = 3 < 5

Most C programmers would assume this would assign the result of "3 < 5" (the value 1) to the variable "a". What this does in **bc** is assign the value 3 to the variable "a" and then compare 3 to 5. It is best to use parenthesis when using relational and logical operators with the assignment operators.

There are a few more special expressions that are provided in **bc**. These have to do with user defined functions and standard functions. They all appear as "*name***(***parameters***)**". See the section on functions for user defined functions. The standard functions are:

length (expression)
The value of the length function is the number of significant digits in the expression.

read ( ) The read function (an extension) will read a number from the standard input, regardless of where the function occurs. Beware, this can cause problems with the mixing of data and program in the standard input. The best use for this function is in a previously written program that needs input from the user, but never allows program code to be input from the user. The value of the read function is the number read from the standard input using the current value of the variable **ibase** for the conversion base.

scale (expression)
The value of the scale function is the number of digits after the decimal point in the expression.

sqrt (expression)
The value of the sqrt function is the square root of the expression. If the expression is negative, a run time error is generated.

STATEMENTS

Statements (as in most algebraic languages) provide the sequencing of expression evaluation. In **bc** statements are executed "as soon as possible." Execution happens when a newline in encountered and there is one or more complete statements. Due to this immediate execution, newlines are very important in **bc**. In fact, both a semicolon and a newline are used as statement separators. An improperly placed newline will cause a syntax error. Because newlines are statement separators, it is possible to hide a newline by using the backslash character. The sequence "\<nl>", where <nl> is the newline appears to **bc** as whitespace instead of a newline. A statement list is a series of statements separated by semicolons and newlines. The following is a list of **bc** statements and what they do: (Things enclosed in brackets ([]) are optional parts of the statement.)

expression
This statement does one of two things. If the expression starts with "<variable> <assignment> ...", it is considered to be an assignment statement. If the expression is not an assignment statement, the expression is evaluated and printed to the output. After the number is printed, a newline is printed. For example, "a=1" is an assignment statement and "(a=1)" is an expression that has an embedded assignment. All numbers that are printed are printed in the base specified by the variable **obase**. The legal values for **obase** are 2 through BC_BASE_MAX. (See the section LIMITS.) For bases 2 through 16, the usual method of writing numbers is used. For bases greater than 16, **bc** uses a multi-character digit method of printing the numbers where each higher base digit is printed as a base 10 number. The multi-character digits are separated by spaces. Each digit contains the number of characters required to represent the base ten value of "obase-1". Since numbers are of arbitrary precision, some numbers may not be printable on a single output line. These long numbers will be split across lines using the "\" as the last character on a line. The maximum

number of characters printed per line is 70. Due to the interactive nature of **bc**, printing a number causes the side effect of assigning the printed value to the special variable **last**. This allows the user to recover the last value printed without having to retype the expression that printed the number. Assigning to **last** is legal and will overwrite the last printed value with the assigned value. The newly assigned value will remain until the next number is printed or another value is assigned to **last**. (Some installations may allow the use of a single period (.) which is not part of a number as a short hand notation for for **last**.)

string The string is printed to the output. Strings start with a double quote character and contain all characters until the next double quote character. All characters are take literally, including any newline. No newline character is printed after the string.

print list

The print statement (an extension) provides another method of output. The "list" is a list of strings and expressions separated by commas. Each string or expression is printed in the order of the list. No terminating newline is printed. Expressions are evaluated and their value is printed and assigned to the variable **last**. Strings in the print statement are printed to the output and may contain special characters. Special characters start with the backslash character (\). The special characters recognized by **bc** are "a" (alert or bell), "b" (backspace), "f" (form feed), "n" (newline), "r" (carriage return), "q" (double quote), "t" (tab), and "\" (backslash). Any other character following the backslash will be ignored.

{ statement_list }

This is the compound statement. It allows multiple statements to be grouped together for execution.

if (expression) statement1 [**else** statement2]

The if statement evaluates the expression and executes statement1 or statement2 depending on the value of the expression. If the expression is non-zero, statement1 is executed. If statement2 is present and the value of the expression is 0, then statement2 is executed. (The else clause is an extension.)

while (expression) statement

The while statement will execute the statement while the expression is non-zero. It evaluates the expression before each execution of the statement. Termination of the loop is caused by a zero expression value or the execution of a break statement.

for ([expression1] ; [expression2] ; [expression3]) statement

The for statement controls repeated execution of the statement. Expression1 is evaluated before the loop. Expression2 is evaluated before each execution of the statement. If it is non-zero, the statement is evaluated. If it is zero, the loop is terminated. After each execution of the statement, expression3 is evaluated before the reevaluation of expression2. If expression1 or expression3 are missing, nothing is evaluated at the point they would be evaluated. If expression2 is missing, it is the same as substituting the value 1 for expression2. (The optional expressions are an extension. POSIX **bc** requires all three expressions.) The following is equivalent code for the for statement:

```
expression1;
while (expression2) {
  statement;
  expression3;
}
```

break This statement causes a forced exit of the most recent enclosing while statement or for statement.

continue

The continue statement (an extension) causes the most recent enclosing for statement to start the next iteration.

halt The halt statement (an extension) is an executed statement that causes the **bc** processor to quit only when it is executed. For example, "if (0 == 1) halt" will not cause **bc** to terminate because the halt is not executed.

return Return the value 0 from a function. (See the section on functions.)

return (expression)
Return the value of the expression from a function. (See the section on functions.) As an extension, the parenthesis are not required.

PSEUDO STATEMENTS

These statements are not statements in the traditional sense. They are not executed statements. Their function is performed at "compile" time.

limits Print the local limits enforced by the local version of **bc**. This is an extension.

quit When the quit statement is read, the **bc** processor is terminated, regardless of where the quit statement is found. For example, "if (0 == 1) quit" will cause **bc** to terminate.

warranty
Print a longer warranty notice. This is an extension.

FUNCTIONS

Functions provide a method of defining a computation that can be executed later. Functions in **bc** always compute a value and return it to the caller. Function definitions are "dynamic" in the sense that a function is undefined until a definition is encountered in the input. That definition is then used until another definition function for the same name is encountered. The new definition then replaces the older definition. A function is defined as follows:

define *name* **(** *parameters* **) {** *newline*
auto_list statement_list **}**

A function call is just an expression of the form "*name***(***parameters***)**".

Parameters are numbers or arrays (an extension). In the function definition, zero or more parameters are defined by listing their names separated by commas. Numbers are only call by value parameters. Arrays are only call by variable. Arrays are specified in the parameter definition by the notation "*name***[]**". In the function call, actual parameters are full expressions for number parameters. The same notation is used for passing arrays as for defining array parameters. The named array is passed by variable to the function. Since function definitions are dynamic, parameter numbers and types are checked when a function is called. Any mismatch in number or types of parameters will cause a runtime error. A runtime error will also occur for the call to an undefined function.

The *auto_list* is an optional list of variables that are for "local" use. The syntax of the auto list (if present) is "**auto** *name*, ... ;". (The semicolon is optional.) Each *name* is the name of an auto variable. Arrays may be specified by using the same notation as used in parameters. These variables have their values pushed onto a stack at the start of the function. The variables are then initialized to zero and used throughout the execution of the function. At function exit, these variables are popped so that the original value (at the time of the function call) of these variables are restored. The parameters are really auto variables that are initialized to a value provided in the function call. Auto variables are different than traditional local variables because if function A calls function B, B may access function A's auto variables by just using the same name, unless function B has called them auto variables. Due to the fact that auto variables and parameters are pushed onto a stack, **bc** supports recursive functions.

The function body is a list of **bc** statements. Again, statements are separated by semicolons or newlines. Return statements cause the termination of a function and the return of a value. There are two versions of the return statement. The first form, "**return**", returns the value 0 to the calling expression. The second form, "**return (** *expression* **)**", computes the value of the expression and returns that value to the calling expression. There is an implied "**return (0)**" at the end of every function. This allows a function to terminate and return 0 without an explicit return statement.

Functions also change the usage of the variable **ibase**. All constants in the function body will be converted using the value of **ibase** at the time of the function call. Changes of **ibase** will be ignored during the execution of the function except for the standard function **read**, which will always use the current value of **ibase** for conversion of numbers.

As an extension, the format of the definition has been slightly relaxed. The standard requires the opening

brace be on the same line as the **define** keyword and all other parts must be on following lines. This version of **bc** will allow any number of newlines before and after the opening brace of the function. For example, the following definitions are legal.

```
define d (n) { return (2*n); }
define d (n)
  { return (2*n); }
```

MATH LIBRARY

If **bc** is invoked with the **-l** option, a math library is preloaded and the default scale is set to 20. The math functions will calculate their results to the scale set at the time of their call. The math library defines the following functions:

s (*x*) The sine of x, x is in radians.

c (*x*) The cosine of x, x is in radians.

a (*x*) The arctangent of x, arctangent returns radians.

l (*x*) The natural logarithm of x.

e (*x*) The exponential function of raising e to the value x.

j (*n*,*x*) The Bessel function of integer order n of x.

EXAMPLES

In /bin/sh, the following will assign the value of "pi" to the shell variable **pi**.

```
pi=$(echo "scale=10; 4*a(1)" | bc -l)
```

The following is the definition of the exponential function used in the math library. This function is written in POSIX **bc**.

```
scale = 20

/* Uses the fact that e^x = (e^(x/2))^2
   When x is small enough, we use the series:
     e^x = 1 + x + x^2/2! + x^3/3! + ...
*/

define e(x) {
  auto  a, d, e, f, i, m, v, z

  /* Check the sign of x. */
  if (x<0) {
    m = 1
    x = -x
  }

  /* Precondition x. */
  z = scale;
  scale = 4 + z + .44*x;
  while (x > 1) {
    f += 1;
    x /= 2;
  }

  /* Initialize the variables. */
  v = 1+x
  a = x
  d = 1

  for (i=2; 1; i++) {
```

```
        e = (a *= x) / (d *= i)
        if (e == 0) {
          if (f>0) while (f--)  v = v*v;
          scale = z
          if (m) return (1/v);
          return (v/1);
        }
        v += e
      }
    }
```

The following is code that uses the extended features of **bc** to implement a simple program for calculating checkbook balances. This program is best kept in a file so that it can be used many times without having to retype it at every use.

```
scale=2
print "\nCheck book program!\n"
print "  Remember, deposits are negative transactions.\n"
print "  Exit by a 0 transaction.\n\n"

print "Initial balance? "; bal = read()
bal /= 1
print "\n"
while (1) {
  "current balance = "; bal
  "transaction? "; trans = read()
  if (trans == 0) break;
  bal -= trans
  bal /= 1
}
quit
```

The following is the definition of the recursive factorial function.

```
define f (x) {
  if (x <= 1) return (1);
  return (f(x-1) * x);
}
```

READLINE AND LIBEDIT OPTIONS

GNU **bc** can be compiled (via a configure option) to use the GNU **readline** input editor library or the BSD **libedit** library. This allows the user to do editing of lines before sending them to **bc**. It also allows for a history of previous lines typed. When this option is selected, **bc** has one more special variable. This special variable, **history** is the number of lines of history retained. For **readline**, a value of -1 means that an unlimited number of history lines are retained. Setting the value of **history** to a positive number restricts the number of history lines to the number given. The value of 0 disables the history feature. The default value is 100. For more information, read the user manuals for the GNU **readline**, **history** and BSD **libedit** libraries. One can not enable both **readline** and **libedit** at the same time.

DIFFERENCES

This version of **bc** was implemented from the POSIX P1003.2/D11 draft and contains several differences and extensions relative to the draft and traditional implementations. It is not implemented in the traditional way using *dc(1)*. This version is a single process which parses and runs a byte code translation of the program. There is an "undocumented" option (-c) that causes the program to output the byte code to the standard output instead of running it. It was mainly used for debugging the parser and preparing the math library.

A major source of differences is extensions, where a feature is extended to add more functionality and additions, where new features are added. The following is the list of differences and extensions.

LANG environment
: This version does not conform to the POSIX standard in the processing of the LANG environment variable and all environment variables starting with LC_.

names
: Traditional and POSIX **bc** have single letter names for functions, variables and arrays. They have been extended to be multi-character names that start with a letter and may contain letters, numbers and the underscore character.

Strings
: Strings are not allowed to contain NUL characters. POSIX says all characters must be included in strings.

last
: POSIX **bc** does not have a **last** variable. Some implementations of **bc** use the period (.) in a similar way.

comparisons
: POSIX **bc** allows comparisons only in the if statement, the while statement, and the second expression of the for statement. Also, only one relational operation is allowed in each of those statements.

if statement, else clause
: POSIX **bc** does not have an else clause.

for statement
: POSIX **bc** requires all expressions to be present in the for statement.

&&, ||, !
: POSIX **bc** does not have the logical operators.

read function
: POSIX **bc** does not have a read function.

print statement
: POSIX **bc** does not have a print statement .

continue statement
: POSIX **bc** does not have a continue statement.

return statement
: POSIX **bc** requires parentheses around the return expression.

array parameters
: POSIX **bc** does not (currently) support array parameters in full. The POSIX grammar allows for arrays in function definitions, but does not provide a method to specify an array as an actual parameter. (This is most likely an oversight in the grammar.) Traditional implementations of **bc** have only call by value array parameters.

function format
: POSIX **bc** requires the opening brace on the same line as the **define** key word and the **auto** statement on the next line.

=+, =-, =*, =/, =%, =^
: POSIX **bc** does not require these "old style" assignment operators to be defined. This version may allow these "old style" assignments. Use the limits statement to see if the installed version supports them. If it does support the "old style" assignment operators, the statement "a =- 1" will decrement **a** by 1 instead of setting **a** to the value -1.

spaces in numbers
: Other implementations of **bc** allow spaces in numbers. For example, "x=1 3" would assign the value 13 to the variable x. The same statement would cause a syntax error in this version of **bc**.

errors and execution
: This implementation varies from other implementations in terms of what code will be executed when syntax and other errors are found in the program. If a syntax error is found in a function definition, error recovery tries to find the beginning of a statement and continue to parse the

function. Once a syntax error is found in the function, the function will not be callable and becomes undefined. Syntax errors in the interactive execution code will invalidate the current execution block. The execution block is terminated by an end of line that appears after a complete sequence of statements. For example,

```
a = 1
b = 2
```

has two execution blocks and

```
{ a = 1
  b = 2 }
```

has one execution block. Any runtime error will terminate the execution of the current execution block. A runtime warning will not terminate the current execution block.

Interrupts

During an interactive session, the SIGINT signal (usually generated by the control-C character from the terminal) will cause execution of the current execution block to be interrupted. It will display a "runtime" error indicating which function was interrupted. After all runtime structures have been cleaned up, a message will be printed to notify the user that **bc** is ready for more input. All previously defined functions remain defined and the value of all non-auto variables are the value at the point of interruption. All auto variables and function parameters are removed during the clean up process. During a non-interactive session, the SIGINT signal will terminate the entire run of **bc**.

LIMITS

The following are the limits currently in place for this **bc** processor. Some of them may have been changed by an installation. Use the limits statement to see the actual values.

BC_BASE_MAX

The maximum output base is currently set at 999. The maximum input base is 16.

BC_DIM_MAX

This is currently an arbitrary limit of 65535 as distributed. Your installation may be different.

BC_SCALE_MAX

The number of digits after the decimal point is limited to INT_MAX digits. Also, the number of digits before the decimal point is limited to INT_MAX digits.

BC_STRING_MAX

The limit on the number of characters in a string is INT_MAX characters.

exponent

The value of the exponent in the raise operation (^) is limited to LONG_MAX.

variable names

The current limit on the number of unique names is 32767 for each of simple variables, arrays and functions.

ENVIRONMENT VARIABLES

The following environment variables are processed by **bc**:

POSIXLY_CORRECT

This is the same as the **-s** option.

BC_ENV_ARGS

This is another mechanism to get arguments to **bc**. The format is the same as the command line arguments. These arguments are processed first, so any files listed in the environment arguments are processed before any command line argument files. This allows the user to set up "standard" options and files to be processed at every invocation of **bc**. The files in the environment variables would typically contain function definitions for functions the user wants defined every time **bc** is run.

BC_LINE_LENGTH

This should be an integer specifying the number of characters in an output line for numbers. This includes the backslash and newline characters for long numbers.

DIAGNOSTICS

If any file on the command line can not be opened, **bc** will report that the file is unavailable and terminate. Also, there are compile and run time diagnostics that should be self-explanatory.

BUGS

Error recovery is not very good yet.

Email bug reports to **bug-bc@gnu.org**. Be sure to include the word ‘‘bc’’ somewhere in the ‘‘Subject:’’ field.

AUTHOR

Philip A. Nelson
philnelson@acm.org

ACKNOWLEDGEMENTS

The author would like to thank Steve Sommars (Steve.Sommars@att.com) for his extensive help in testing the implementation. Many great suggestions were given. This is a much better product due to his involvement.

NAME

bdes — encrypt/decrypt using the Data Encryption Standard

SYNOPSIS

bdes [**-abdp**] [**-F** *N*] [**-f** *N*] [**-k** *key*] [**-m** *N*] [**-o** *N*] [**-v** *vector*]

DESCRIPTION

bdes implements all DES modes of operation described in FIPS PUB 81, including alternative cipher feedback mode and both authentication modes. **bdes** reads from the standard input and writes to the standard output. By default, the input is encrypted using cipher block chaining mode. Using the same key for encryption and decryption preserves plain text.

All modes but the electronic code book mode require an initialization vector; if none is supplied, the zero vector is used. If no *key* is specified on the command line, the user is prompted for one (see `getpass`(3) for more details).

The options are as follows:

-a The key and initialization vector strings are to be taken as ASCII, suppressing the special interpretation given to leading “0X”, “0x”, “0B”, and “0b” characters. This flag applies to *both* the key and initialization vector.

-b Use electronic code book mode. This is not recommended for messages longer than 8 bytes, as patterns in the input will show through to the output.

-d Decrypt the input.

-F *N* Use *N*-bit alternative cipher feedback mode. Currently *N* must be a multiple of 7 between 7 and 56 inclusive (this does not conform to the alternative CFB mode specification).

-f *N* Use *N*-bit cipher feedback mode. Currently *N* must be a multiple of 8 between 8 and 64 inclusive (this does not conform to the standard CFB mode specification).

-k *key* Use *key* as the cryptographic key.

-m *N* Compute a message authentication code (MAC) of *N* bits on the input. The value of *N* must be between 1 and 64 inclusive; if *N* is not a multiple of 8, enough 0 bits will be added to pad the MAC length to the nearest multiple of 8. Only the MAC is output. MACs are only available in cipher block chaining mode or in cipher feedback mode.

-o *N* Use *N*-bit output feedback mode. Currently *N* must be a multiple of 8 between 8 and 64 inclusive (this does not conform to the OFB mode specification).

-p Disable the resetting of the parity bit. This flag forces the parity bit of the key to be used as typed, rather than making each character be of odd parity. It is used only if the key is given in ASCII.

-v *vector*
Set the initialization vector to *vector*; the vector is interpreted in the same way as the key. The vector is ignored in electronic codebook mode. For best security, a different initialization vector should be used for each file.

The key and initialization vector are taken as sequences of ASCII characters which are then mapped into their bit representations. If either begins with “0X” or “0x”, that one is taken as a sequence of hexadecimal digits indicating the bit pattern; if either begins with “0B” or “0b”, that one is taken as a sequence of binary digits indicating the bit pattern. In either case, only the leading 64 bits of the key or initialization vector are used, and if fewer than 64 bits are provided, enough 0 bits are appended to pad the key to 64 bits.

According to the DES standard, the low-order bit of each character in the key string is deleted. Since most ASCII representations set the high-order bit to 0, simply deleting the low-order bit effectively reduces the size of the key space from 2^{56} to 2^{48} keys. To prevent this, the high-order bit must be a function depending in part upon the low-order bit; so, the high-order bit is set to whatever value gives odd parity. This preserves the key space size. Note this resetting of the parity bit is *not* done if the key is given in binary or hex, and can be disabled for ASCII keys as well.

The DES is considered a very strong cryptosystem hobbled by a short key, and other than table lookup attacks, key search attacks, and Hellman's time-memory tradeoff (all of which are very expensive and time-consuming), no practical cryptanalytic methods for breaking the DES are known in the open literature. As of this writing, the best known cryptanalytic method is linear cryptanalysis, which requires an average of 2^{43} known plaintext-ciphertext pairs to succeed. Unfortunately for the DES, key search attacks (requiring only a single known plaintext-ciphertext pair and trying 2^{55} keys on average) are becoming practical.

As with all cryptosystems, the choice of keys and key security remain the most vulnerable aspect of **bdes**.

IMPLEMENTATION NOTES

For implementors wishing to write software compatible with this program, the following notes are provided. This software is believed to be compatible with the implementation of the data encryption standard distributed by Sun Microsystems, Inc.

In the ECB and CBC modes, plaintext is encrypted in units of 64 bits (8 bytes, also called a block). To ensure that the plaintext file is encrypted correctly, **bdes** will (internally) append from 1 to 8 bytes, the last byte containing an integer stating how many bytes of that final block are from the plaintext file, and encrypt the resulting block. Hence, when decrypting, the last block may contain from 0 to 7 characters present in the plaintext file, and the last byte tells how many. Note that if during decryption the last byte of the file does not contain an integer between 0 and 7, either the file has been corrupted or an incorrect key has been given. A similar mechanism is used for the OFB and CFB modes, except that those simply require the length of the input to be a multiple of the mode size, and the final byte contains an integer between 0 and one less than the number of bytes being used as the mode. (This was another reason that the mode size must be a multiple of 8 for those modes.)

Unlike Sun's implementation, unused bytes of that last block are not filled with random data, but instead contain what was in those byte positions in the preceding block. This is quicker and more portable, and does not weaken the encryption significantly.

If the key is entered in ASCII, the parity bits of the key characters are set so that each key character is of odd parity. Unlike Sun's implementation, it is possible to enter binary or hexadecimal keys on the command line, and if this is done, the parity bits are *not* reset. This allows testing using arbitrary bit patterns as keys.

The Sun implementation always uses an initialization vector of 0 (that is, all zeroes). By default, **bdes** does too, but this may be changed from the command line.

SEE ALSO

`crypt`(3), `getpass`(3)

Data Encryption Standard, Federal Information Processing Standard #46, National Bureau of Standards, U.S. Department of Commerce, January 1977, Washington DC.

DES Modes of Operation, Federal Information Processing Standard #81, National Bureau of Standards, U.S. Department of Commerce, December 1980, Washington DC.

Dorothy Denning, *Cryptography and Data Security*, *Addison-Wesley Publishing Co.*, 1982, Reading, MA.

Matt Bishop, *Implementation Notes on bdes(1)*, Technical Report PCS-TR-91-158, Department of Mathematics and Computer Science, Dartmouth College, April 1991, Hanover, NH 03755.

M.J. Wiener, *Efficient DES Key Search*, Technical Report 244, School of Computer Science, Carleton University, May 1994.

Bruce Schneier, *Applied Cryptography (2nd edition)*, *John Wiley & Sons, Inc.*, 1996, New York, NY.

M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, *Springer-Verlag*, Advances in Cryptology -- Eurocrypt '93 Proceedings, 1994.

Blaze, Diffie, Rivest, Schneier, Shimomura, Thompson, and Wiener, *Minimal Key Lengths for Symmetric Ciphers To Provide Adequate Commercial Security*, *Business Software Alliance*, January 1996, http://www.bsa.org/policy/encryption/cryptographers.html.

BUGS

When this document was originally written, there was a controversy raging over whether the DES would still be secure in a few years. There is now near-universal consensus in the cryptographic community that the key length of the DES is far too short. The advent of special-purpose hardware could reduce the cost of any of the methods of attack named above so that they are no longer computationally infeasible; in addition, the explosive growth in the number and speed of modern microprocessors as well as advances in programmable logic devices has brought an attack using only commodity hardware into the realm of possibility. Schneier and others currently recommend using cryptosystems with keys of at least 90 bits when long-term security is needed.

As the key or key schedule is stored in memory, the encryption can be compromised if memory is readable. Additionally, programs which display programs' arguments may compromise the key and initialization vector, if they are specified on the command line. To avoid this **bdes** overwrites its arguments, however, the obvious race cannot currently be avoided.

Certain specific keys should be avoided because they introduce potential weaknesses; these keys, called the *weak* and *semiweak* keys, are (in hex notation, where p is either 0 or 1, and P is either e or f):

`0x0p0p0p0p0p0p0p0p`	`0x0p1P0p1P0p0P0p0P`
`0x0pep0pep0pfp0pfp`	`0x0pfP0pfP0pfP0pfP`
`0x1P0p1P0p0P0p0P0p`	`0x1P1P1P1P0P0P0P0P`
`0x1Pep1Pep0Pfp0Pfp`	`0x1PfP1PfP0PfP0PfP`
`0xep0pep0pfp0pfp0p`	`0xep1Pep1pfp0Pfp0P`
`0xepepepepepepepep`	`0xepfPepfPfpfPfpfP`
`0xfP0pfP0pfP0pfP0p`	`0xfP1PfP1PfP0PfP0P`
`0xfPepfPepfPepfPep`	`0xfPfPfPfPfPfPfPfP`

This is inherent in the DES algorithm (see Moore and Simmons, "Cycle structure of the DES with weak and semi-weak keys", *Advances in Cryptology − Crypto '86 Proceedings*, Springer-Verlag New York, ©1987, pp. 9-32.)

NAME

bellctrl — opm bell emulation preference utility

SYNOPSIS

bellctrl [-**b**] [**b** *on/off*] [**b** [*volume* [*pitch* [*duration*]]] [**v** *default*] [**-v** *voicefile*]

DESCRIPTION

bellctrl is used to set various preference options of the opm bell.

The options are as follows:

-b The b option controls bell volume, pitch and duration. This option accepts up to three numerical parameters, a preceding dash(-), or a 'on/off' flag. If no parameters are given, or the 'on' flag is used, the system defaults will be used. If the dash or 'off' are given, the bell will be turned off. If only one numerical parameter is given, the bell volume will be set to that value, as a percentage of its maximum. Like wise, the second numerical parameter specifies the bell pitch, in hertz, and the third numerical parameter specifies the duration in milliseconds.

v *default* The *default* argument cause the voice parameter to be reset to the system default.

-v *voicefile*
The **-v** option sets the voice parameter to given voice parameter file (*voicefile*).

AUTHORS

bellctrl was written by Takuya Harakawa <ussy@int.titech.ac.jp>.

NAME

biff — be notified if mail arrives and who it is from

SYNOPSIS

biff [**ny**]

DESCRIPTION

biff informs the system whether you want to be notified when mail arrives during the current terminal session.

Options supported by **biff**:

n Disables notification.

y Enables notification.

When mail notification is enabled, the header and first few lines of the message will be printed on your screen whenever mail arrives. A "`biff y`" command is often included in the file `.login` or `.profile` to be executed at each login.

biff operates asynchronously. For synchronous notification use the *MAIL* variable of `sh`(1) or the *mail* variable of `csh`(1).

SEE ALSO

`csh`(1), `mail`(1), `sh`(1), `comsat`(8)

HISTORY

The **biff** command appeared in 4.0BSD.

NAME

bpm — menu-based binary package manager

SYNOPSIS

bpm [**-hnVv**] [**-b** *baseURL*] [**-m** *machine*] [**-r** *release*] [**-w** *seconds*]

DESCRIPTION

The **bpm** command is used to locate and install binary packages from any reachable URL.

The following command-line options are supported:

-b *baseURL*
: Specify a base URL from which to download binary packages. The default URL is `ftp://ftp.NetBSD.org/pub/pkgsrc/packages`.

-h
: Print a help message and then exit.

-m *machine*
: Use *machine* as the machine architecture to be used, instead of that returned by `uname`(1).

-n
: Don't actually execute the commands to add the package.

-r *release*
: Use *release* as the operating system release to be used, instead of that returned by `uname`(1).

-V
: Print version number and exit.

-v
: Turn on verbose output.

-w *seconds*
: The number of *seconds* to wait after displaying an error message and returning to normal menu operations.

bpm provides a menu-based binary package manager for NetBSD. **bpm** first connects to the URL using `ftp`(1), and displays a list of categories for which binary packages exist. If no categories are displayed, it could be that the machine architecture or operating system release string have been wrongly interpreted, and that it will be necessary to override this values by means of the command line options. Within a category, a list of packages will be displayed, and by selecting one using the number assigned to it, the package will be downloaded automatically, and installed, using the `pkg_add`(1) utility. It is also possible to change the category currently being examined, and to quit from the utility, simply by selecting the appropriate choices on the menu.

ENVIRONMENT

The environment variables which govern the behaviour of `ftp`(1) and `pkg_add`(1) are valid for **bpm**.

SEE ALSO

`ftp`(1), `pkg_add`(1), `uname`(1)

AUTHORS

The **bpm** utility was written by Alistair Crooks ⟨agc@NetBSD.org⟩.

NAME

cpio — copy files to and from archives

SYNOPSIS

cpio {**-i**} [*options*] [*pattern ...*] [< *archive*]
cpio {**-o**} [*options*] < *name-list* [> *archive*]
cpio {**-p**} [*options*] *dest-dir* < *name-list*

DESCRIPTION

cpio copies files between archives and directories. This implementation can extract from tar, pax, cpio, zip, jar, ar, and ISO 9660 cdrom images and can create tar, pax, cpio, ar, and shar archives.

The first option to **cpio** is a mode indicator from the following list:

-i Input. Read an archive from standard input (unless overriden) and extract the contents to disk or (if the **-t** option is specified) list the contents to standard output. If one or more file patterns are specified, only files matching one of the patterns will be extracted.

-o Output. Read a list of filenames from standard input and produce a new archive on standard output (unless overriden) containing the specified items.

-p Pass-through. Read a list of filenames from standard input and copy the files to the specified directory.

OPTIONS

Unless specifically stated otherwise, options are applicable in all operating modes.

-A (o mode only) Append to the specified archive. (Not yet implemented.)

-a (o and p modes) Reset access times on files after they are read.

-B (o mode only) Block output to records of 5120 bytes.

-C *size*
(o mode only) Block output to records of *size* bytes.

-c (o mode only) Use the old POSIX portable character format. Equivalent to **--format** *odc*.

-d (i and p modes) Create directories as necessary.

-E *file*
(i mode only) Read list of file name patterns from *file* to list and extract.

-F *file*
Read archive from or write archive to *file*.

-f *pattern*
(i mode only) Ignore files that match *pattern*.

--format *format*
(o mode only) Produce the output archive in the specified format. Supported formats include:

cpio	Synonym for *odc*.
newc	The SVR4 portable cpio format.
odc	The old POSIX.1 portable octet-oriented cpio format.
pax	The POSIX.1 pax format, an extension of the ustar format.
ustar	The POSIX.1 tar format.

The default format is *odc*. See libarchive_formats(5) for more complete information about the formats currently supported by the underlying libarchive(3) library.

-I *file*
Read archive from *file*.

-i Input mode. See above for description.

--insecure
(i and p mode only) Disable security checks during extraction or copying. This allows extraction via symbolic links and path names containing '..' in the name.

-L (o and p modes) All symbolic links will be followed. Normally, symbolic links are archived and copied as symbolic links. With this option, the target of the link will be archived or copied instead.

-l (p mode only) Create links from the target directory to the original files, instead of copying.

-m (i and p modes) Set file modification time on created files to match those in the source.

-O *file*
Write archive to *file*.

-o Output mode. See above for description.

-p Pass-through mode. See above for description.

--quiet
Suppress unnecessary messages.

-R [user][:][group]
Set the owner and/or group on files in the output. If group is specified with no user (for example, **-R** *:wheel*) then the group will be set but not the user. If the user is specified with a trailing colon and no group (for example, **-R** *root:*) then the group will be set to the user's default group. If the user is specified with no trailing colon, then the user will be set but not the group. In **-i** and **-p** modes, this option can only be used by the super-user. (For compatibility, a period can be used in place of the colon.)

-r (All modes.) Rename files interactively. For each file, a prompt is written to `/dev/tty` containing the name of the file and a line is read from `/dev/tty`. If the line read is blank, the file is skipped. If the line contains a single period, the file is processed normally. Otherwise, the line is taken to be the new name of the file.

-t (i mode only) List the contents of the archive to stdout; do not restore the contents to disk.

-u (i and p modes) Unconditionally overwrite existing files. Ordinarily, an older file will not overwrite a newer file on disk.

-v Print the name of each file to stderr as it is processed. With **-t**, provide a detailed listing of each file.

--version
Print the program version information and exit.

-y (o mode only) Compress the archive with bzip2-compatible compression before writing it. In input mode, this option is ignored; bzip2 compression is recognized automatically on input.

-Z (o mode only) Compress the archive with compress-compatible compression before writing it. In input mode, this option is ignored; compression is recognized automatically on input.

-z (o mode only) Compress the archive with gzip-compatible compression before writing it. In input mode, this option is ignored; gzip compression is recognized automatically on input.

ENVIRONMENT

The following environment variables affect the execution of **cpio**:

`LANG` The locale to use. See `environ`(7) for more information.

`TZ` The timezone to use when displaying dates. See `environ`(7) for more information.

EXIT STATUS

The **cpio** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

The **cpio** command is traditionally used to copy file heirarchies in conjunction with the `find`(1) command. The first example here simply copies all files from `src` to `dest`:

```
find src | cpio -pmud dest
```

By carefully selecting options to the `find`(1) command and combining it with other standard utilities, it is possible to exercise very fine control over which files are copied. This next example copies files from `src` to `dest` that are more than 2 days old and whose names match a particular pattern:

```
find src -mtime +2 | grep foo[bar] | cpio -pdmu dest
```

This example copies files from `src` to `dest` that are more than 2 days old and which contain the word "foobar":

```
find src -mtime +2 | xargs grep -l foobar | cpio -pdmu dest
```

COMPATIBILITY

The mode options i, o, and p and the options a, B, c, d, f, l, m, r, t, u, and v comply with SUSv2.

The old POSIX.1 standard specified that only **-i**, **-o**, and **-p** were interpreted as command-line options. Each took a single argument of a list of modifier characters. For example, the standard syntax allows **-imu** but does not support **-miu** or **-i -m -u**, since *m* and *u* are only modifiers to **-i**, they are not command-line options in their own right. The syntax supported by this implementation is backwards-compatible with the standard. For best compatibility, scripts should limit themselves to the standard syntax.

SEE ALSO

`bzip2`(1), `tar`(1), `gzip`(1), `mt`(1), `pax`(1), `libarchive`(3), `cpio`(5), `libarchive-formats`(5), `tar`(5)

STANDARDS

There is no current POSIX standard for the cpio command; it appeared in ISO/IEC 9945-1:1996 ("POSIX.1") but was dropped from IEEE Std 1003.1-2001 ("POSIX.1").

The cpio, ustar, and pax interchange file formats are defined by IEEE Std 1003.1-2001 ("POSIX.1") for the pax command.

HISTORY

The original **cpio** and **find** utilities were written by Dick Haight while working in AT&T's Unix Support Group. They first appeared in 1977 in PWB/UNIX 1.0, the "Programmer's Work Bench" system developed for use within AT&T. They were first released outside of AT&T as part of System III Unix in 1981. As a result, **cpio** actually predates **tar**, even though it was not well-known outside of AT&T until some time later.

This is a complete re-implementation based on the `libarchive`(3) library.

BUGS

The cpio archive format has several basic limitations: It does not store user and group names, only numbers. As a result, it cannot be reliably used to transfer files between systems with dissimilar user and group numbering. Older cpio formats limit the user and group numbers to 16 or 18 bits, which is insufficient for modern systems. The cpio archive formats cannot support files over 4 gigabytes, except for the “odc” variant, which can support files up to 8 gigabytes.

NAME

tar — manipulate tape archives

SYNOPSIS

tar [*bundled-flags* ⟨args⟩] [⟨*file*⟩ | ⟨*pattern*⟩ ...]
tar {**-c**} [*options*] [*files* | *directories*]
tar {**-r** | **-u**} **-f** *archive-file* [*options*] [*files* | *directories*]
tar {**-t** | **-x**} [*options*] [*patterns*]

DESCRIPTION

tar creates and manipulates streaming archive files. This implementation can extract from tar, pax, cpio, zip, jar, ar, and ISO 9660 cdrom images and can create tar, pax, cpio, ar, and shar archives.

The first synopsis form shows a "bundled" option word. This usage is provided for compatibility with historical implementations. See COMPATIBILITY below for details.

The other synopsis forms show the preferred usage. The first option to **tar** is a mode indicator from the following list:

-c Create a new archive containing the specified items.

-r Like **-c**, but new entries are appended to the archive. Note that this only works on uncompressed archives stored in regular files. The **-f** option is required.

-t List archive contents to stdout.

-u Like **-r**, but new entries are added only if they have a modification date newer than the corresponding entry in the archive. Note that this only works on uncompressed archives stored in regular files. The **-f** option is required.

-x Extract to disk from the archive. If a file with the same name appears more than once in the archive, each copy will be extracted, with later copies overwriting (replacing) earlier copies.

In **-c**, **-r**, or **-u** mode, each specified file or directory is added to the archive in the order specified on the command line. By default, the contents of each directory are also archived.

In extract or list mode, the entire command line is read and parsed before the archive is opened. The pathnames or patterns on the command line indicate which items in the archive should be processed. Patterns are shell-style globbing patterns as documented in `tcsh`(1).

OPTIONS

Unless specifically stated otherwise, options are applicable in all operating modes.

`@archive`

(c and r mode only) The specified archive is opened and the entries in it will be appended to the current archive. As a simple example,

```
tar -c -f - newfile @original.tar
```

writes a new archive to standard output containing a file `newfile` and all of the entries from `original.tar`. In contrast,

```
tar -c -f - newfile original.tar
```

creates a new archive with only two entries. Similarly,

```
tar -czf - --format pax @-
```

reads an archive from standard input (whose format will be determined automatically) and converts it into a gzip-compressed pax-format archive on stdout. In this way, **tar** can be used to convert archives from one format to another.

-b *blocksize*

Specify the block size, in 512-byte records, for tape drive I/O. As a rule, this argument is only needed when reading from or writing to tape drives, and usually not even then as the default block size of 20 records (10240 bytes) is very common.

-C *directory*
In c and r mode, this changes the directory before adding the following files. In x mode, change directories after opening the archive but before extracting entries from the archive.

--check-links (**-W check-links**)
(c and r modes only) Issue a warning message unless all links to each file are archived.

--chroot (**-W chroot**)
(x mode only) **chroot**() to the current directory after processing any **-C** options and before extracting any files.

--exclude *pattern* (**-W exclude**=*pattern*)
Do not process files or directories that match the specified pattern. Note that exclusions take precedence over patterns or filenames specified on the command line.

--format *format* (**-W format**=*format*)
(c, r, u mode only) Use the specified format for the created archive. Supported formats include "cpio", "pax", "shar", and "ustar". Other formats may also be supported; see `libarchive-formats`(5) for more information about currently-supported formats. In r and u modes, when extending an existing archive, the format specified here must be compatible with the format of the existing archive on disk.

-f *file*
Read the archive from or write the archive to the specified file. The filename can be `-` for standard input or standard output. If not specified, the default tape device will be used. (On FreeBSD, the default tape device is `/dev/sa0`.)

-H (c and r mode only) Symbolic links named on the command line will be followed; the target of the link will be archived, not the link itself.

-h (c and r mode only) Synonym for **-L**.

-I Synonym for **-T**.

--include *pattern* (**-W include**=*pattern*)
Process only files or directories that match the specified pattern. Note that exclusions specified with **--exclude** take precedence over inclusions. If no inclusions are explicitly specified, all entries are processed by default. The **--include** option is especially useful when filtering archives. For example, the command

```
tar -c -f new.tar --include='*foo*' @old.tgz
```

creates a new archive `new.tar` containing only the entries from `old.tgz` containing the string 'foo'.

-j (c mode only) Compress the resulting archive with `bzip2`(1). In extract or list modes, this option is ignored. Note that, unlike other **tar** implementations, this implementation recognizes bzip2 compression automatically when reading archives.

-k (x mode only) Do not overwrite existing files. In particular, if a file appears more than once in an archive, later copies will not overwrite earlier copies.

--keep-newer-files (**-W keep-newer-files**)
(x mode only) Do not overwrite existing files that are newer than the versions appearing in the archive being extracted.

-L (c and r mode only) All symbolic links will be followed. Normally, symbolic links are archived as such. With this option, the target of the link will be archived instead.

-l This is a synonym for the **--check-links** option.

-m (x mode only) Do not extract modification time. By default, the modification time is set to the time stored in the archive.

-n (c, r, u modes only) Do not recursively archive the contents of directories.

--newer *date* (**-W newer**=*date*)
(c, r, u modes only) Only include files and directories newer than the specified date. This compares ctime entries.

--newer-mtime *date* (**-W newer-mtime**=*date*)
(c, r, u modes only) Like **--newer**, except it compares mtime entries instead of ctime entries.

--newer-than file (**-W newer-than**=file)
(c, r, u modes only) Only include files and directories newer than the specified file. This compares ctime entries.

--newer-mtime-than file (**-W newer-mtime-than**=file)
(c, r, u modes only) Like **--newer-than**, except it compares mtime entries instead of ctime entries.

--nodump (**-W nodump**)
(c and r modes only) Honor the nodump file flag by skipping this file.

--null (**-W null**)
(use with **-I**, **-T**, or **-X**) Filenames or patterns are separated by null characters, not by newlines. This is often used to read filenames output by the **-print0** option to find(1).

--numeric-owner
(x mode only) Ignore symbolic user and group names when restoring archives to disk, only numeric uid and gid values will be obeyed.

-O (x, t modes only) In extract (-x) mode, files will be written to standard out rather than being extracted to disk. In list (-t) mode, the file listing will be written to stderr rather than the usual stdout.

-o (x mode) Use the user and group of the user running the program rather than those specified in the archive. Note that this has no significance unless **-p** is specified, and the program is being run by the root user. In this case, the file modes and flags from the archive will be restored, but ACLs or owner information in the archive will be discarded.

-o (c, r, u mode) A synonym for **--format** *ustar*

--one-file-system (**-W one-file-system**)
(c, r, and u modes) Do not cross mount points.

-P Preserve pathnames. By default, absolute pathnames (those that begin with a / character) have the leading slash removed both when creating archives and extracting from them. Also, **tar** will refuse to extract archive entries whose pathnames contain **..** or whose target directory would be altered by a symlink. This option suppresses these behaviors.

-p (x mode only) Preserve file permissions. Attempt to restore the full permissions, including owner, file modes, file flags and ACLs, if available, for each item extracted from the archive. By default, newly-created files are owned by the user running **tar**, the file mode is restored for newly-created regular files, and all other types of entries receive default permissions. If **tar** is being run by root, the default is to restore the owner unless the **-o** option is also specified.

-q (**--fast-read**)
(x and t mode only) Extract or list only the first archive entry that matches each pattern or filename operand. Exit as soon as each specified pattern or filename has been matched. By default, the archive is always read to the very end, since there can be multiple entries with the same name and, by convention, later entries overwrite earlier entries. This option is provided as a performance optimization.

-S (x mode only) Extract files as sparse files. For every block on disk, check first if it contains only NULL bytes and seek over it otherwise. This works similiar to the conv=sparse option of dd.

--strip-components *count* (**-W strip-components**=*count*)
(x and t mode only) Remove the specified number of leading path elements. Pathnames with fewer elements will be silently skipped. Note that the pathname is edited after checking inclusion/exclusion patterns but before security checks.

-s *pattern*
Modify file or archive member names according to `pattern`. The pattern has the format /old/new/[gps]. old is a basic regular expression. If it doesn't apply, the pattern is skipped. new is the replacement string of the matched part. ~ is substituted with the match, 1 to 9 with the content of the corresponding captured group. The optional trailing g specifies that matching should continue after the matched part and stopped on the first unmatched pattern. The optional trailing s specifies that the pattern applies to the value of symbolic links. The optional trailing p specifies that after a successful substitution the original path name and the new path name should be printed to standard error.

-T *filename*
In x or t mode, **tar** will read the list of names to be extracted from `filename`. In c mode, **tar** will read names to be archived from `filename`. The special name "-C" on a line by itself will cause the current directory to be changed to the directory specified on the following line. Names are terminated by newlines unless **--null** is specified. Note that **--null** also disables the special handling of lines containing "-C".

-U (x mode only) Unlink files before creating them. Without this option, **tar** overwrites existing files, which preserves existing hardlinks. With this option, existing hardlinks will be broken, as will any symlink that would affect the location of an extracted file.

--use-compress-program *program*
Pipe the input (in x or t mode) or the output (in c mode) through `program` instead of using the builtin compression support.

-v Produce verbose output. In create and extract modes, **tar** will list each file name as it is read from or written to the archive. In list mode, **tar** will produce output similar to that of `ls`(1). Additional **-v** options will provide additional detail.

-W *longopt=value*
Long options (preceded by **--**) are only supported directly on systems that have the `getopt_long`(3) function. The **-W** option can be used to access long options on systems that do not support this function.

-w Ask for confirmation for every action.

-X *filename*
Read a list of exclusion patterns from the specified file. See **--exclude** for more information about the handling of exclusions.

-y (c mode only) Compress the resulting archive with bzip2(1). In extract or list modes, this option is ignored. Note that, unlike other **tar** implementations, this implementation recognizes bzip2 compression automatically when reading archives.

-z (c mode only) Compress the resulting archive with gzip(1). In extract or list modes, this option is ignored. Note that, unlike other **tar** implementations, this implementation recognizes gzip compression automatically when reading archives.

-Z (c mode only) Compress the resulting archive with compress(1). In extract or list modes, this option is ignored. Note that, unlike other **tar** implementations, this implementation recognizes compress compression automatically when reading archives.

ENVIRONMENT

The following environment variables affect the execution of **tar**:

LANG The locale to use. See environ(7) for more information.

TAPE The default tape device. The **-f** option overrides this.

TZ The timezone to use when displaying dates. See environ(7) for more information.

FILES

/dev/sa0 The default tape device, if not overridden by the TAPE environment variable or the **-f** option.

EXIT STATUS

The **tar** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

The following creates a new archive called *file.tar.gz* that contains two files *source.c* and *source.h*:

```
tar -czf file.tar.gz source.c source.h
```

To view a detailed table of contents for this archive:

```
tar -tvf file.tar.gz
```

To extract all entries from the archive on the default tape drive:

```
tar -x
```

To examine the contents of an ISO 9660 cdrom image:

```
tar -tf image.iso
```

To move file hierarchies, invoke **tar** as

```
tar -cf - -C srcdir . | tar -xpf - -C destdir
```

or more traditionally

```
cd srcdir ; tar -cf - . | (cd destdir ; tar -xpf -)
```

In create mode, the list of files and directories to be archived can also include directory change instructions of the form **-C**foo/baz and archive inclusions of the form @archive-file. For example, the command line

```
tar -c -f new.tar foo1 @old.tgz -C/tmp foo2
```

will create a new archive new.tar. **tar** will read the file foo1 from the current directory and add it to the output archive. It will then read each entry from old.tgz and add those entries to the output archive. Finally, it will switch to the /tmp directory and add foo2 to the output archive.

An input file in mtree(5) format can be used to create an output archive with arbitrary ownership, permissions, or names that differ from existing data on disk:

```
$ cat input.mtree
usr/bin uid=0 gid=0 mode=0755 type=dir
usr/bin/ls uid=0 gid=0 mode=0755 type=file content=myls
$ tar -cvf output.tar @input.mtree
```

The **--newer** and **--newer-mtime** switches accept a variety of common date and time specifications, including “12 Mar 2005 7:14:29pm”, “2005-03-12 19:14”, “5 minutes ago”, and “19:14 PST May 1”.

COMPATIBILITY

The bundled-arguments format is supported for compatibility with historic implementations. It consists of an initial word (with no leading - character) in which each character indicates an option. Arguments follow as separate words. The order of the arguments must match the order of the corresponding characters in the bundled command word. For example,

tar tbf 32 file.tar

specifies three flags **t**, **b**, and **f**. The **b** and **f** flags both require arguments, so there must be two additional items on the command line. The *32* is the argument to the **b** flag, and *file.tar* is the argument to the **f** flag.

The mode options c, r, t, u, and x and the options b, f, l, m, o, v, and w comply with SUSv2.

For maximum portability, scripts that invoke **tar** should use the bundled-argument format above, should limit themselves to the **c**, **t**, and **x** modes, and the **b**, **f**, **m**, **v**, and **w** options.

On systems that support getopt_long(), additional long options are available to improve compatibility with other tar implementations.

SECURITY

Certain security issues are common to many archiving programs, including **tar**. In particular, carefully-crafted archives can request that **tar** extract files to locations outside of the target directory. This can potentially be used to cause unwitting users to overwrite files they did not intend to overwrite. If the archive is being extracted by the superuser, any file on the system can potentially be overwritten. There are three ways this can happen. Although **tar** has mechanisms to protect against each one, savvy users should be aware of the implications:

- Archive entries can have absolute pathnames. By default, **tar** removes the leading **/** character from filenames before restoring them to guard against this problem.
- Archive entries can have pathnames that include **..** components. By default, **tar** will not extract files containing **..** components in their pathname.
- Archive entries can exploit symbolic links to restore files to other directories. An archive can restore a symbolic link to another directory, then use that link to restore a file into that directory. To guard against this, **tar** checks each extracted path for symlinks. If the final path element is a symlink, it will be removed and replaced with the archive entry. If **-U** is specified, any intermediate symlink will also be unconditionally removed. If neither **-U** nor **-P** is specified, **tar** will refuse to extract the entry.

To protect yourself, you should be wary of any archives that come from untrusted sources. You should examine the contents of an archive with

tar -tf filename

before extraction. You should use the **-k** option to ensure that **tar** will not overwrite any existing files or the **-U** option to remove any pre-existing files. You should generally not extract archives while running with super-user privileges. Note that the **-P** option to **tar** disables the security checks above and allows you to extract an archive while preserving any absolute pathnames, **..** components, or symlinks to other directories.

SEE ALSO

bzip2(1), compress(1), cpio(1), gzip(1), mt(1), pax(1), shar(1), libarchive(3), libarchive-formats(5), tar(5)

STANDARDS

There is no current POSIX standard for the tar command; it appeared in ISO/IEC 9945-1:1996 ("POSIX.1") but was dropped from IEEE Std 1003.1-2001 ("POSIX.1"). The options used by this implementation were developed by surveying a number of existing tar implementations as well as the old POSIX specification for tar and the current POSIX specification for pax.

The ustar and pax interchange file formats are defined by IEEE Std 1003.1-2001 ("POSIX.1") for the pax command.

HISTORY

A **tar** command appeared in Seventh Edition Unix, which was released in January, 1979. There have been numerous other implementations, many of which extended the file format. John Gilmore's **pdtar** public-domain implementation (circa November, 1987) was quite influential, and formed the basis of GNU tar. GNU tar was included as the standard system tar in FreeBSD beginning with FreeBSD 1.0.

This is a complete re-implementation based on the libarchive(3) library.

BUGS

This program follows ISO/IEC 9945-1:1996 ("POSIX.1") for the definition of the **-l** option. Note that GNU tar prior to version 1.15 treated **-l** as a synonym for the **--one-file-system** option.

The **-C** dir option may differ from historic implementations.

All archive output is written in correctly-sized blocks, even if the output is being compressed. Whether or not the last output block is padded to a full block size varies depending on the format and the output device. For tar and cpio formats, the last block of output is padded to a full block size if the output is being written to standard output or to a character or block device such as a tape drive. If the output is being written to a regular file, the last block will not be padded. Many compressors, including gzip(1) and bzip2(1), complain about the null padding when decompressing an archive created by **tar**, although they still extract it correctly.

The compression and decompression is implemented internally, so there may be insignificant differences between the compressed output generated by

```
tar -czf - file
```

and that generated by

```
tar -cf - file | gzip
```

The default should be to read and write archives to the standard I/O paths, but tradition (and POSIX) dictates otherwise.

The **r** and **u** modes require that the archive be uncompressed and located in a regular file on disk. Other archives can be modified using **c** mode with the @archive-file extension.

To archive a file called @foo or -foo you must specify it as ./@foo or ./-foo, respectively.

In create mode, a leading ./ is always removed. A leading / is stripped unless the **-P** option is specified.

There needs to be better support for file selection on both create and extract.

There is not yet any support for multi-volume archives or for archiving sparse files.

Converting between dissimilar archive formats (such as tar and cpio) using the @- convention can cause hard link information to be lost. (This is a consequence of the incompatible ways that different archive formats store hardlink information.)

There are alternative long options for many of the short options that are deliberately not documented.

NAME

bthset — Bluetooth Headset utility

SYNOPSIS

bthset [**-v**] [**-c** *command*] [**-m** *mixer*] [**-p** *file*] [**-s** *channel*]
bthset [**-h**]

DESCRIPTION

The **bthset** utility is used to access a Bluetooth Headset with the btsco(4) audio device. It opens the mixer(4) device and creates the control connection to the headset, then conducts volume settings each way. When **bthset** receives a SIGUSR1 signal, it will start sending RING codes to the headset until the headset button is pressed or a SIGUSR2 signal is received. If the headset is ringing and **bthset** receives a button press notification, the specified *command* (if any) is executed.

When the server channel is specified with the **-s** option, instead of opening the control connection, **bthset** will listen on the *channel* for incoming connections from the Headset and register as Headset Audio Gateway with the local SDP server.

The options are as follows:

-c *command*
: Specify an optional command to be executed when the headset responds to ringing with a button press event. This defaults to the contents of the BTHSET_COMMAND environment variable, if set.

-h
: Print usage message.

-m *mixer*
: Specify the path to the mixer device. This defaults to the contents of the BTHSET_MIXER environment variable, or /dev/mixer if not set.

-p *file*
: Write the PID to *file* so that other programs can find us later. This defaults to the contents of the BTHSET_PIDFILE environment variable, if set.

-s *channel*
: Register with the local SDP server as a Headset Audio Gateway and listen for connections on the given RFCOMM server channel.

-v
: Be verbose.

EXIT STATUS

The **bthset** utility exits 0 on success, and >0 if an error occurs.

FILES

/dev/mixer

SEE ALSO

btsco(4), mixer(4)

AUTHORS

Iain Hibbert for Itronix, Inc

NAME

btkey — Bluetooth Link Key management utility

SYNOPSIS

btkey [**-cCrRwW**] [**-k** *key*] **-a** *address* **-d** *device*
btkey -lL [**-d** *device*]

DESCRIPTION

The **btkey** program is used to manage Bluetooth Link Key storage. Keys are normally handled by the bthcid(8) daemon which caches them in the /var/db/bthcid.keys file and provides them as required when Bluetooth connections need to be authenticated.

These keys are required for connections between remote devices and the specific controller (not the Operating System) and so for multi-boot systems where it may not always be possible to specify the same key across all OS's it can be better to have the Bluetooth controller provide the keys directly from its semi-permanent memory once devices are paired. **btkey** will read, write or clear keys in device memory or the key cache as required.

Note that without the bthcid(8) daemon running users will be unable to supply PINs, and Link Keys resulting from new pairings will not be stored. If no new pairings are expected and the keys are stored in the controller then bthcid(8) is not required.

The options are as follows:

-a *address* Specify the remote device address. May be given as a BDADDR or a name.

-c Clear key from file.

-C Clear key from device.

-d *device* Specify the local device address. May be given as a BDADDR or a name.

-k *key* Supply a Link Key as a string of hexadecimal digits. Up to 32 digits will be processed and the resulting key will be zero padded to 16 octets.

-l List keys stored in file.

-L List keys stored in device.

-r Read key from file.

-R Read key from device.

-w Write key to file.

-W Write key to device.

Super-user privileges are required to read or write link keys.

EXAMPLES

Read key for mouse at ubt0 from file and write to device

```
btkey -d ubt0 -a mouse -rW
```

Write new key for keyboard at ubt0 to file

```
btkey -d ubt0 -a keyboard -k 92beda6cd8b8f66ebd2af270d55d70ec -w
```

Clear key for phone at bt3c0 from file and device

```
btkey -d bt3c0 -a phone -cC
```

EXIT STATUS

The **btkey** utility exits 0 on success, and >0 if an error occurs.

FILES

`/var/db/bthcid.keys`

SEE ALSO

`btpin`(1), `btconfig`(8), `bthcid`(8)

AUTHORS

Iain Hibbert

NAME

btpin — Bluetooth PIN utility

SYNOPSIS

btpin [**-d** *device*] [**-s** *path*] { **-p** *pin* | **-r** [**-l** *len*]} **-a** *address*

DESCRIPTION

The **btpin** utility is used to register a temporary PIN with the bthcid(8) daemon for the purposes of pairing Bluetooth devices. The PIN will be valid for 5 minutes or until used, whichever comes first.

The options are as follows:

-a *address*
: Specify the remote device address. The address can be specified as BD_ADDR or a name. If a name was specified, the **btpin** utility attempts to resolve the name via bt_gethostbyname(3).

-d *device*
: Specify the local device address. The device can be specified by BD_ADDR or device name. See btconfig(8) for a list of devices available. If no device is specified, the PIN will be valid for any local device.

-l *len*
: Specify length of PIN to generate, where $1 \leq len \leq 16$.

-p *pin*
: The PIN to register. The PIN may be up to 16 bytes in length.

-r
: Generate a random PIN, the default length is 4 bytes.

-s *path*
: Specify path to the control socket. The default path is /var/run/bthcid.

EXIT STATUS

The **btpin** utility exits 0 on success, and >0 if an error occurs.

FILES

/var/run/bthcid

SEE ALSO

btkey(1), btconfig(8), bthcid(8)

AUTHORS

Iain Hibbert for Itronix, Inc

NAME

bzcmp, bzdiff − compare bzip2 compressed files

SYNOPSIS

bzcmp [cmp_options] file1 [file2]
bzdiff [diff_options] file1 [file2]

DESCRIPTION

Bzcmp and *bzdiff* are used to invoke the *cmp* or the *diff* program on bzip2 compressed files. All options specified are passed directly to *cmp* or *diff*. If only 1 file is specified, then the files compared are *file1* and an uncompressed *file1*.bz2. If two files are specified, then they are uncompressed if necessary and fed to *cmp* or *diff*. The exit status from *cmp* or *diff* is preserved.

SEE ALSO

cmp(1), diff(1), bzmore(1), bzless(1), bzgrep(1), bzip2(1)

BUGS

Messages from the *cmp* or *diff* programs refer to temporary filenames instead of those specified.

NAME

bzgrep, bzfgrep, bzegrep − search possibly bzip2 compressed files for a regular expression

SYNOPSIS

bzgrep [grep_options] **[-e]** *pattern filename*...
bzegrep [egrep_options] **[-e]** *pattern filename*...
bzfgrep [fgrep_options] **[-e]** *pattern filename*...

DESCRIPTION

Bzgrep is used to invoke the *grep* on bzip2-compressed files. All options specified are passed directly to *grep*. If no file is specified, then the standard input is decompressed if necessary and fed to grep. Otherwise the given files are uncompressed if necessary and fed to *grep*.

If *bzgrep* is invoked as *bzegrep* or *bzfgrep* then *egrep* or *fgrep* is used instead of *grep*. If the GREP environment variable is set, *bzgrep* uses it as the *grep* program to be invoked. For example:

for sh: GREP=fgrep bzgrep string files
for csh: (setenv GREP fgrep; bzgrep string files)

AUTHOR

Charles Levert (charles@comm.polymtl.ca). Adapted to bzip2 by Philippe Troin <phil@fifi.org> for Debian GNU/Linux.

SEE ALSO

grep(1), egrep(1), fgrep(1), bzdiff(1), bzmore(1), bzless(1), bzip2(1)

NAME

bzip2, bunzip2 − a block-sorting file compressor, v1.0.5
bzcat − decompresses files to stdout
bzip2recover − recovers data from damaged bzip2 files

SYNOPSIS

bzip2 [**−cdfkqstvzVL123456789**] [*filenames ...*]
bunzip2 [**−fkvsVL**] [*filenames ...*]
bzcat [**−s**] [*filenames ...*]
bzip2recover *filename*

DESCRIPTION

bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78-based compressors, and approaches the performance of the PPM family of statistical compressors.

The command-line options are deliberately very similar to those of *GNU gzip,* but they are not identical.

bzip2 expects a list of file names to accompany the command-line flags. Each file is replaced by a compressed version of itself, with the name "original_name.bz2". Each compressed file has the same modification date, permissions, and, when possible, ownership as the corresponding original, so that these properties can be correctly restored at decompression time. File name handling is naive in the sense that there is no mechanism for preserving original file names, permissions, ownerships or dates in filesystems which lack these concepts, or have serious file name length restrictions, such as MS-DOS.

bzip2 and *bunzip2* will by default not overwrite existing files. If you want this to happen, specify the −f flag.

If no file names are specified, *bzip2* compresses from standard input to standard output. In this case, *bzip2* will decline to write compressed output to a terminal, as this would be entirely incomprehensible and therefore pointless.

bunzip2 (or *bzip2 −d)* decompresses all specified files. Files which were not created by *bzip2* will be detected and ignored, and a warning issued. *bzip2* attempts to guess the filename for the decompressed file from that of the compressed file as follows:

filename.bz2 becomes filename
filename.bz becomes filename
filename.tbz2 becomes filename.tar
filename.tbz becomes filename.tar
anyothername becomes anyothername.out

If the file does not end in one of the recognised endings, *.bz2, .bz, .tbz2* or *.tbz, bzip2* complains that it cannot guess the name of the original file, and uses the original name with *.out* appended.

As with compression, supplying no filenames causes decompression from standard input to standard output.

bunzip2 will correctly decompress a file which is the concatenation of two or more compressed files. The result is the concatenation of the corresponding uncompressed files. Integrity testing (−t) of concatenated compressed files is also supported.

You can also compress or decompress files to the standard output by giving the −c flag. Multiple files may be compressed and decompressed like this. The resulting outputs are fed sequentially to stdout.

Compression of multiple files in this manner generates a stream containing multiple compressed file representations. Such a stream can be decompressed correctly only by *bzip2* version 0.9.0 or later. Earlier versions of *bzip2* will stop after decompressing the first file in the stream.

bzcat (or *bzip2 -dc)* decompresses all specified files to the standard output.

bzip2 will read arguments from the environment variables *BZIP2* and *BZIP*, in that order, and will process them before any arguments read from the command line. This gives a convenient way to supply default arguments.

Compression is always performed, even if the compressed file is slightly larger than the original. Files of less than about one hundred bytes tend to get larger, since the compression mechanism has a constant overhead in the region of 50 bytes. Random data (including the output of most file compressors) is coded at about 8.05 bits per byte, giving an expansion of around 0.5%.

As a self-check for your protection, *bzip2* uses 32-bit CRCs to make sure that the decompressed version of a file is identical to the original. This guards against corruption of the compressed data, and against undetected bugs in *bzip2* (hopefully very unlikely). The chances of data corruption going undetected is microscopic, about one chance in four billion for each file processed. Be aware, though, that the check occurs upon decompression, so it can only tell you that something is wrong. It can't help you recover the original uncompressed data. You can use *bzip2recover* to try to recover data from damaged files.

Return values: 0 for a normal exit, 1 for environmental problems (file not found, invalid flags, I/O errors, &c), 2 to indicate a corrupt compressed file, 3 for an internal consistency error (e.g., bug) which caused *bzip2* to panic.

OPTIONS

−c --stdout
Compress or decompress to standard output.

−d --decompress
Force decompression. *bzip2*, *bunzip2* and *bzcat* are really the same program, and the decision about what actions to take is done on the basis of which name is used. This flag overrides that mechanism, and forces *bzip2* to decompress.

−z --compress
The complement to −d: forces compression, regardless of the invocation name.

−t --test
Check integrity of the specified file(s), but don't decompress them. This really performs a trial decompression and throws away the result.

−f --force
Force overwrite of output files. Normally, *bzip2* will not overwrite existing output files. Also forces *bzip2* to break hard links to files, which it otherwise wouldn't do.

bzip2 normally declines to decompress files which don't have the correct magic header bytes. If forced (-f), however, it will pass such files through unmodified. This is how GNU gzip behaves.

−k --keep
Keep (don't delete) input files during compression or decompression.

−s --small
Reduce memory usage, for compression, decompression and testing. Files are decompressed and tested using a modified algorithm which only requires 2.5 bytes per block byte. This means any file can be decompressed in 2300k of memory, albeit at about half the normal speed.

During compression, −s selects a block size of 200k, which limits memory use to around the same

figure, at the expense of your compression ratio. In short, if your machine is low on memory (8 megabytes or less), use −s for everything. See MEMORY MANAGEMENT below.

−q --quiet
Suppress non-essential warning messages. Messages pertaining to I/O errors and other critical events will not be suppressed.

−v --verbose
Verbose mode -- show the compression ratio for each file processed. Further −v's increase the verbosity level, spewing out lots of information which is primarily of interest for diagnostic purposes.

−L --license -V --version
Display the software version, license terms and conditions.

−1 (or −−fast) to −9 (or −−best)
Set the block size to 100 k, 200 k .. 900 k when compressing. Has no effect when decompressing. See MEMORY MANAGEMENT below. The −−fast and −−best aliases are primarily for GNU gzip compatibility. In particular, −−fast doesn't make things significantly faster. And −−best merely selects the default behaviour.

−- Treats all subsequent arguments as file names, even if they start with a dash. This is so you can handle files with names beginning with a dash, for example: bzip2 −- −myfilename.

−-repetitive-fast --repetitive-best
These flags are redundant in versions 0.9.5 and above. They provided some coarse control over the behaviour of the sorting algorithm in earlier versions, which was sometimes useful. 0.9.5 and above have an improved algorithm which renders these flags irrelevant.

MEMORY MANAGEMENT

bzip2 compresses large files in blocks. The block size affects both the compression ratio achieved, and the amount of memory needed for compression and decompression. The flags −1 through −9 specify the block size to be 100,000 bytes through 900,000 bytes (the default) respectively. At decompression time, the block size used for compression is read from the header of the compressed file, and *bunzip2* then allocates itself just enough memory to decompress the file. Since block sizes are stored in compressed files, it follows that the flags −1 to −9 are irrelevant to and so ignored during decompression.

Compression and decompression requirements, in bytes, can be estimated as:

Compression: 400k + (8 x block size)

Decompression: 100k + (4 x block size), or
100k + (2.5 x block size)

Larger block sizes give rapidly diminishing marginal returns. Most of the compression comes from the first two or three hundred k of block size, a fact worth bearing in mind when using *bzip2* on small machines. It is also important to appreciate that the decompression memory requirement is set at compression time by the choice of block size.

For files compressed with the default 900k block size, *bunzip2* will require about 3700 kbytes to decompress. To support decompression of any file on a 4 megabyte machine, *bunzip2* has an option to decompress using approximately half this amount of memory, about 2300 kbytes. Decompression speed is also halved, so you should use this option only where necessary. The relevant flag is -s.

In general, try and use the largest block size memory constraints allow, since that maximises the compression achieved. Compression and decompression speed are virtually unaffected by block size.

Another significant point applies to files which fit in a single block -- that means most files you'd encounter using a large block size. The amount of real memory touched is proportional to the size of the file, since

the file is smaller than a block. For example, compressing a file 20,000 bytes long with the flag -9 will cause the compressor to allocate around 7600k of memory, but only touch 400k + 20000 * 8 = 560 kbytes of it. Similarly, the decompressor will allocate 3700k but only touch 100k + 20000 * 4 = 180 kbytes.

Here is a table which summarises the maximum memory usage for different block sizes. Also recorded is the total compressed size for 14 files of the Calgary Text Compression Corpus totalling 3,141,622 bytes. This column gives some feel for how compression varies with block size. These figures tend to understate the advantage of larger block sizes for larger files, since the Corpus is dominated by smaller files.

Flag	Compress usage	Decompress usage	Decompress -s usage	Corpus Size
-1	1200k	500k	350k	914704
-2	2000k	900k	600k	877703
-3	2800k	1300k	850k	860338
-4	3600k	1700k	1100k	846899
-5	4400k	2100k	1350k	845160
-6	5200k	2500k	1600k	838626
-7	6100k	2900k	1850k	834096
-8	6800k	3300k	2100k	828642
-9	7600k	3700k	2350k	828642

RECOVERING DATA FROM DAMAGED FILES

bzip2 compresses files in blocks, usually 900kbytes long. Each block is handled independently. If a media or transmission error causes a multi-block .bz2 file to become damaged, it may be possible to recover data from the undamaged blocks in the file.

The compressed representation of each block is delimited by a 48-bit pattern, which makes it possible to find the block boundaries with reasonable certainty. Each block also carries its own 32-bit CRC, so damaged blocks can be distinguished from undamaged ones.

bzip2recover is a simple program whose purpose is to search for blocks in .bz2 files, and write each block out into its own .bz2 file. You can then use *bzip2* −t to test the integrity of the resulting files, and decompress those which are undamaged.

bzip2recover takes a single argument, the name of the damaged file, and writes a number of files "rec00001file.bz2", "rec00002file.bz2", etc, containing the extracted blocks. The output filenames are designed so that the use of wildcards in subsequent processing -- for example, "bzip2 -dc rec*file.bz2 > recovered_data" -- processes the files in the correct order.

bzip2recover should be of most use dealing with large .bz2 files, as these will contain many blocks. It is clearly futile to use it on damaged single-block files, since a damaged block cannot be recovered. If you wish to minimise any potential data loss through media or transmission errors, you might consider compressing with a smaller block size.

PERFORMANCE NOTES

The sorting phase of compression gathers together similar strings in the file. Because of this, files containing very long runs of repeated symbols, like "aabaabaabaab ..." (repeated several hundred times) may compress more slowly than normal. Versions 0.9.5 and above fare much better than previous versions in this respect. The ratio between worst-case and average-case compression time is in the region of 10:1. For previous versions, this figure was more like 100:1. You can use the −vvvv option to monitor progress in great detail, if you want.

Decompression speed is unaffected by these phenomena.

bzip2 usually allocates several megabytes of memory to operate in, and then charges all over it in a fairly random fashion. This means that performance, both for compressing and decompressing, is largely determined by the speed at which your machine can service cache misses. Because of this, small changes to the code to reduce the miss rate have been observed to give disproportionately large performance improvements. I imagine *bzip2* will perform best on machines with very large caches.

CAVEATS

I/O error messages are not as helpful as they could be. *bzip2* tries hard to detect I/O errors and exit cleanly, but the details of what the problem is sometimes seem rather misleading.

This manual page pertains to version 1.0.5 of *bzip2*. Compressed data created by this version is entirely forwards and backwards compatible with the previous public releases, versions 0.1pl2, 0.9.0, 0.9.5, 1.0.0, 1.0.1, 1.0.2 and 1.0.3, but with the following exception: 0.9.0 and above can correctly decompress multiple concatenated compressed files. 0.1pl2 cannot do this; it will stop after decompressing just the first file in the stream.

bzip2recover versions prior to 1.0.2 used 32-bit integers to represent bit positions in compressed files, so they could not handle compressed files more than 512 megabytes long. Versions 1.0.2 and above use 64-bit ints on some platforms which support them (GNU supported targets, and Windows). To establish whether or not bzip2recover was built with such a limitation, run it without arguments. In any event you can build yourself an unlimited version if you can recompile it with MaybeUInt64 set to be an unsigned 64-bit integer.

AUTHOR

Julian Seward, jsewardbzip.org.

http://www.bzip.org

The ideas embodied in *bzip2* are due to (at least) the following people: Michael Burrows and David Wheeler (for the block sorting transformation), David Wheeler (again, for the Huffman coder), Peter Fenwick (for the structured coding model in the original *bzip*, and many refinements), and Alistair Moffat, Radford Neal and Ian Witten (for the arithmetic coder in the original *bzip)*. I am much indebted for their help, support and advice. See the manual in the source distribution for pointers to sources of documentation. Christian von Roques encouraged me to look for faster sorting algorithms, so as to speed up compression. Bela Lubkin encouraged me to improve the worst-case compression performance. Donna Robinson XMLised the documentation. The bz* scripts are derived from those of GNU gzip. Many people sent patches, helped with portability problems, lent machines, gave advice and were generally helpful.

NAME

bzmore, bzless − file perusal filter for crt viewing of bzip2 compressed text

SYNOPSIS

bzmore [name ...]
bzless [name ...]

NOTE

In the following description, *bzless* and *less* can be used interchangeably with *bzmore* and *more.*

DESCRIPTION

Bzmore is a filter which allows examination of compressed or plain text files one screenful at a time on a soft-copy terminal. *bzmore* works on files compressed with *bzip2* and also on uncompressed files. If a file does not exist, *bzmore* looks for a file of the same name with the addition of a .bz2 suffix.

Bzmore normally pauses after each screenful, printing --More-- at the bottom of the screen. If the user then types a carriage return, one more line is displayed. If the user hits a space, another screenful is displayed. Other possibilities are enumerated later.

Bzmore looks in the file */etc/termcap* to determine terminal characteristics, and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size is 22 lines. Other sequences which may be typed when *bzmore* pauses, and their effects, are as follows (*i* is an optional integer argument, defaulting to 1) :

i <space>
: display *i* more lines, (or another screenful if no argument is given)

ˆD
: display 11 more lines (a ‘‘scroll’’). If *i* is given, then the scroll size is set to *i* .

d
: same as ˆD (control-D)

i z
: same as typing a space except that *i* , if present, becomes the new window size. Note that the window size reverts back to the default at the end of the current file.

i s
: skip *i* lines and print a screenful of lines

i f
: skip *i* screenfuls and print a screenful of lines

q or Q
: quit reading the current file; go on to the next (if any)

e or q
: When the prompt --More--(Next file: *file*) is printed, this command causes bzmore to exit.

s
: When the prompt --More--(Next file: *file*) is printed, this command causes bzmore to skip the next file and continue.

=
: Display the current line number.

i /expr
: search for the *i* -th occurrence of the regular expression *expr.* If the pattern is not found, *bzmore* goes on to the next file (if any). Otherwise, a screenful is displayed, starting two lines before the place where the expression was found. The user’s erase and kill characters may be used to edit the regular expression. Erasing back past the first column cancels the search command.

i n
: search for the *i* -th occurrence of the last regular expression entered.

!command
: invoke a shell with *command* . The character ‘!’ in "command" are replaced with the previous shell command. The sequence "\!" is replaced by "!".

:q or :Q
quit reading the current file; go on to the next (if any) (same as q or Q).

. (dot) repeat the previous command.

The commands take effect immediately, i.e., it is not necessary to type a carriage return. Up to the time when the command character itself is given, the user may hit the line kill character to cancel the numerical argument being formed. In addition, the user may hit the erase character to redisplay the --More-- message.

At any time when output is being sent to the terminal, the user can hit the quit key (normally control−\). *Bzmore* will stop sending output, and will display the usual --More-- prompt. The user may then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

The terminal is set to *noecho* mode by this program so that the output can be continuous. What you type will thus not show on your terminal, except for the / and ! commands.

If the standard output is not a teletype, then *bzmore* acts just like *bzcat*, except that a header is printed before each file.

FILES

/etc/termcap Terminal data base

SEE ALSO

more(1), less(1), bzip2(1), bzdiff(1), bzgrep(1)

NAME

c89 — ANSI (1989) C compiler

SYNOPSIS

c89 [**-pedantic**] [**-pedantic-errors**] [**-D_ANSI_SOURCE**] [options ...]

DESCRIPTION

Calls the C compiler (cc) with the given *options*, using a C language environment compatible with the ANSI X3.159-1989 ("ANSI C89") specification.

This includes proper handling of trigraphs, disabling non-ANSI compiler features (such as *asm*, *inline*, *typeof*, and the $ character in identifiers), and definition of the preprocessor symbol __STRICT_ANSI__.

The following options are available:

-pedantic Issue extra warnings defined by ANSI for use of non-ANSI features.

-pedantic-errors Issue errors instead of warnings that normally would be presented by **-pedantic**.

-D_ANSI_SOURCE Tell the system header file set to use an ANSI-conformant "clean" namespace.

SEE ALSO

cc(1)

STANDARDS

c89 conforms to IEEE Std 1003.2-1992 ("POSIX.2").

HISTORY

c89 appeared in NetBSD 1.4.

BUGS

Since **c89** is a shell wrapper script to *cc*, compile errors are prefixed by "cc:".

NAME

cal — displays a calendar

SYNOPSIS

cal [**-3hjry**] [**-A** *after*] [**-B** *before*] [**-d** *day-of-week*] [**-R** *reform-spec*] [[*month*] *year*]

DESCRIPTION

cal displays a simple calendar. If arguments are not specified, the current month is displayed. The options are as follows:

-3 Same as "**-A** *1* **-B** *1*".

-A *after*
Display *after* months after the specified month.

-B *before*
Display *before* months before the specified month.

-d *day-of-week*
Specifies the day of the week on which the calendar should start. Valid values are 0 through 6, presenting Sunday through Saturday, inclusively. The default output starts on Sundays.

-h Highlight the current day, if present in the displayed calendar. If output is to a terminal, then the appropriate terminal sequences are used, otherwise overstriking is used. If more than one **-h** is used and output is to a terminal, the current date will be highlighted in inverse video instead of bold.

-j Display Julian dates (days one-based, numbered from January 1).

-R *reform-spec*
Selects an alternate Gregorian reform point from the default of September 3rd, 1752. The *reform-spec* can be selected by one of the built-in names (see **NOTES** for a list) or by a date of the form YYYY/MM/DD. The date and month may be omitted, provided that what is specified uniquely selects a given built-in reform point. If an exact date is specified, then that date is taken to be the first missing date of the Gregorian Reform to be applied.

-r Display the month in which the Gregorian Reform adjustment was applied, if no other *month* or *year* information is given. If used in conjunction with **-y**, then the entire year is displayed.

-y Display a calendar for the current year.

If no parameters are specified, the current month's calendar is displayed. A single parameter specifies the year and optionally the month in ISO format: "`cal 2007-12`" Two parameters denote the month (1 - 12) and year. Note that the century must be included in the year.

A year starts on Jan 1.

NOTES

In the USA and Great Britain the Gregorian Reformation occurred in 1752. By this time, most countries had recognized the reformation (although a few did not recognize it until the 1900's.) Eleven days following September 2, 1752 were eliminated by the reformation, so the calendar for that month is a bit unusual.

In view of the chaotic way the Gregorian calendar was adopted throughout the world in the years between 1582 and 1928 make sure to take into account the date of the Gregorian Reformation in your region if you are checking a calendar for a very old date.

cal has a decent built-in list of Gregorian Reform dates and the names of the countries where the reform was adopted:

Italy	Oct. 5, 1582	Denmark	Feb. 19, 1700
Spain	Oct. 5, 1582	Great Britain	Sep. 3, 1752
Portugal	Oct. 5, 1582	Sweden	Feb. 18, 1753
Poland	Oct. 5, 1582	Finland	Feb. 18, 1753
France	Dec. 12, 1582	Japan	Dec. 20, 1872
Luxembourg	Dec. 22, 1582	China	Nov. 7, 1911
Netherlands	Dec. 22, 1582	Bulgaria	Apr. 1, 1916
Bavaria	Oct. 6, 1583	U.S.S.R.	Feb. 1, 1918
Austria	Jan. 7, 1584	Serbia	Jan. 19, 1919
Switzerland	Jan. 12, 1584	Romania	Jan. 19, 1919
Hungary	Oct. 22, 1587	Greece	Mar. 10, 1924
Germany	Feb. 19, 1700	Turkey	Dec. 19, 1925
Norway	Feb. 19, 1700	Egypt	Sep. 18, 1928

The country known as *Great Britain* can also be referred to as *England* since that has less letters and no spaces in it. This is meant only as a measure of expediency, not as a possible slight to anyone involved.

HISTORY

A **cal** command appeared in Version 6 AT&T UNIX.

NAME

calendar — reminder service

SYNOPSIS

calendar [**-ax**] [**-d** *MMDD[[YY]YY]*] [**-f** *file*] [**-l** *days*] [**-w** *days*]

DESCRIPTION

The **calendar** utility processes text files and displays lines that match certain dates.

The following options are available:

-a Process the "calendar" files of all users and mail the results to them. This requires super-user privileges.

-d *MMDD[[YY]YY]*
Display lines for the given date. By default, the current date is used. The year, which may be given in either two or four digit format, is used only for purposes of determining wether the given date falls on a Friday in that year (see below). If the year is not specified, the current year is assumed.

-f *file*
Display matching calendar files from the given filename. By default, the following filenames are checked for:

```
~/calendar
~/.calendar
/etc/calendar
```

and the first which is found is used. The filename may be absolute. If not absolute, it is taken relative to the current directory or the directory specified by the `CALENDAR_DIR` environment variable. Or, if the **-a** flag is given, a non-absolute filename is taken relative to each user's home directory in turn.

-l *days*
Causes the program to "look ahead" a given number of days (default one) from the specified date and display their entries as well.

-w *days*
Causes the program to add the specified number of days to the "look ahead" number if and only if the day specified is a Friday. The default value is two, which causes **calendar** to print entries through the weekend on Fridays.

-x Causes **calendar** not to set the `CPP_RESTRICTED` environment variable. Passing this flag allows users the (somewhat obscure) option of including a named pipe via `cpp`(1)'s `#include` syntax, but opens up the possibility of **calendar** hanging indefinitely if users do so incorrectly. For this reason, the **-x** flag should never be used with **calendar -a**.

Lines should begin with a month and day. They may be entered in almost any format, either numeric or as character strings. A single asterisk (`*`) matches every month, or every day if a month has been provided. This means that two asterisks (`**`) matches every day of the year, and is thus useful for ToDo tasks. A day without a month matches that day of every week. A month without a day matches the first of that month. Two numbers default to the month followed by the day. Lines with leading tabs default to the last entered date, allowing multiple line specifications for a single date. By convention, dates followed by an asterisk are not fixed, i.e., change from year to year.

The "calendar" file is preprocessed by `cpp`(1), allowing the inclusion of shared files such as company holidays or meetings. If the shared file is not referenced by a full pathname, `cpp`(1) searches in the current (or home) directory first, and then in the directory `/usr/share/calendar`. Empty lines and lines protected by the C commenting syntax (`/* ... */`) are ignored.

Some possible calendar entries:

```
#include       <calendar.usholiday>
#include       <calendar.birthday>

6/15           ... June 15 (if ambiguous, will default to month/day).
Jun. 15        ... June 15.
15 June        ... June 15.
Thursday       ... Every Thursday.
June           ... Every June 1st.
15 *           ... 15th of every month.
*15            ... 15th of every month.
June*          ... Every day of June.
**             ... Every day
```

FILES

The following default calendar files are provided:

`calendar.birthday`	Births and deaths of famous (and not-so-famous) people.
`calendar.christian`	Christian holidays. This calendar should be updated yearly by the local system administrator so that roving holidays are set correctly for the current year.
`calendar.computer`	Days of special significance to computer people.
`calendar.history`	Everything else, mostly U.S. historical events.
`calendar.holiday`	Other holidays, including the not-well-known, obscure, and *really* obscure.
`calendar.judaic`	Jewish holidays. This calendar should be updated yearly by the local system administrator so that roving holidays are set correctly for the current year.
`calendar.lotr`	Important dates in the Lord of the Rings series.
`calendar.music`	Musical events, births, and deaths. Strongly oriented toward rock 'n' roll.
`calendar.netbsd`	Important dates in the history of the NetBSD project. Mostly releases and port additions.
`calendar.usholiday`	U.S. holidays. This calendar should be updated yearly by the local system administrator so that roving holidays are set correctly for the current year.

SEE ALSO

`at`(1), `cpp`(1), `cron`(8)

COMPATIBILITY

The **`calendar`** program previously selected lines which had the correct date anywhere in the line. This is no longer true, the date is only recognized when it occurs first on the line.

In NetBSD 3.0, the **`calendar`** command was modified to search the user's home directory instead of the current directory by default. Users desiring the historical behavior should set the `CALENDAR_DIR` environment variable to `.`, or use the **`-f`** flag.

HISTORY

A **`calendar`** command appeared in Version 7 AT&T UNIX.

BUGS

`calendar` doesn't handle events that move around from year to year, i.e., "the last Monday in April".

The **`-a`** option ignores the user's `CALENDAR_DIR` environment variable.

NAME

cap_mkdb — create capability database

SYNOPSIS

cap_mkdb [**-b** | **-l**] [**-v**] [**-f** *outfile*] *file1* [*file2 . . .*]

DESCRIPTION

cap_mkdb builds a hashed database out of the getcap(3) logical database constructed by the concatenation of the specified files.

The database is named by the basename of the first file argument and the string ".db". The getcap(3) routines can access the database in this form much more quickly than they can the original text file(s).

The ''tc'' capabilities of the records are expanded before the record is stored into the database.

The options are as follows:

-b Use big-endian byte order for database metadata.

-f *outfile*
Specify a different database basename.

-l Use little-endian byte order for database metadata.

-v Print out the number of capability records in the database.

The **-b** and the **-l** flags are mutually exclusive. The default byte ordering is the current host order.

FORMAT

The following is a description of the hashed database created by **cap_mkdb**. For a description of the format of the input files see termcap(5).

Each record is stored in the database using two different types of keys.

The first type is a key which consists of the first capability of the record (not including the trailing colon (":")) with a data field consisting of a special byte followed by the rest of the record. The special byte is either a 0 or 1, where a 0 means that the record is okay, and a 1 means that there was a ''tc'' capability in the record that couldn't be expanded.

The second type is a key which consists of one of the names from the first capability of the record with a data field consisting a special byte followed by the first capability of the record. The special byte is a 2.

In normal operation names are looked up in the database, resulting in a key/data pair of the second type. The data field of this key/data pair is used to look up a key/data pair of the first type which has the real data associated with the name.

EXIT STATUS

The **cap_mkdb** utility exits 0 on success and >0 if an error occurs.

SEE ALSO

dbopen(3), getcap(3), termcap(5)

NAME

cat — concatenate and print files

SYNOPSIS

cat [**-beflnstuv**] [**-**] [*file ...*]

DESCRIPTION

The **cat** utility reads files sequentially, writing them to the standard output. The *file* operands are processed in command line order. A single dash represents the standard input, and may appear multiple times in the *file* list.

The word "concatenate" is just a verbose synonym for "catenate".

The options are as follows:

-b Implies the **-n** option but doesn't number blank lines.

-e Implies the **-v** option, and displays a dollar sign ('$') at the end of each line as well.

-f Only attempt to display regular files.

-l Set an exclusive advisory lock on the standard output file descriptor. This lock is set using fcntl(2) with the F_SETLKW command. If the output file is already locked, **cat** will block until the lock is acquired.

-n Number the output lines, starting at 1.

-s Squeeze multiple adjacent empty lines, causing the output to be single spaced.

-t Implies the **-v** option, and displays tab characters as '^I' as well.

-u The **-u** option guarantees that the output is unbuffered.

-v Displays non-printing characters so they are visible. Control characters print as '^X' for control-X; the delete character (octal 0177) prints as '^?'. Non-ascii characters (with the high bit set) are printed as 'M-' (for meta) followed by the character for the low 7 bits.

EXIT STATUS

The **cat** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

The command:

```
cat file1
```

will print the contents of *file1* to the standard output.

The command:

```
cat file1 file2 > file3
```

will sequentially print the contents of *file1* and *file2* to the file *file3*, truncating *file3* if it already exists. See the manual page for your shell (i.e., sh(1)) for more information on redirection.

The command:

```
cat file1 - file2 - file3
```

will print the contents of *file1*, print data it receives from the standard input until it receives an EOF ('^D') character, print the contents of *file2*, read and output contents of the standard input again, then finally output the contents of *file3*. Note that if the standard input referred to a file, the second dash on the

command-line would have no effect, since the entire contents of the file would have already been read and printed by **cat** when it encountered the first '-' operand.

SEE ALSO

head(1), hexdump(1), lpr(1), more(1), pr(1), tail(1), view(1), vis(1), fcntl(2)

Rob Pike, "UNIX Style, or cat -v Considered Harmful", *USENIX Summer Conference Proceedings*, 1983.

STANDARDS

The **cat** utility is expected to conform to the IEEE Std 1003.2-1992 ("POSIX.2") specification.

The flags [**-belnstv**] are extensions to the specification.

HISTORY

A **cat** utility appeared in Version 1 AT&T UNIX. Dennis Ritchie designed and wrote the first man page. It appears to have been cat(1).

BUGS

Because of the shell language mechanism used to perform output redirection, the command "`cat file1 file2 > file1`" will cause the original data in file1 to be destroyed! This is performed by the shell before **cat** is run.

NAME

cc — front-end to the C compiler

SYNOPSIS

cc [**-cEgkLMPOStvxX**] [**-fPIC**] [**-fpic**] [**-m***option*] [**-nostartfiles**] [**-nostdinc**] [**-nostdlib**] [**-pg**] [**-pthread**] [**-static**] [**-B** *prefix*] [**-D** *macro[=value]*] [**-d** *option*] [**-I** *directory*] [**-include** *path*] [**-isystem** *path*] [**-o** *outfile*] [**-Wl** *flags*] [*file . . .*]

DESCRIPTION

The **cc** utility provides a front-end to the "portable C compiler." Multiple files may be given on the command line. Unrecognized options are all sent directly to ld(1).

Filenames that end with **.c** are passed via cpp(1) -> ccom(1) -> as(1) -> ld(1).

Filenames that end with **.i** are passed via ccom(1) -> as(1) -> ld(1).

Filenames that end with **.s** are passed via as(1) -> ld(1).

Filenames that end with **.o** are passed directly to ld(1).

The options are as follows:

-B *prefix*
: Define alternate prefix path for cpp(1), ccom(1), as(1), or ld(1) executables.

-C
: Passed to the cpp(1) preprocessor to not discard comments.

-c
: Only compile or assemble and then stop. Do not link. The resulting object output is saved as a filename with a ".o" suffix unless **-o** option is used. Note: cannot be combined with **-o** if multiple files are given.

-D *macro[=value]*
: Passed to the cpp(1) preprocessor to define *macro*.

-d *option*
: Passed to the as(1) assembler.

-E
: Stop after preprocessing with cpp(1). Do not compile, assemble, or link. Output is sent to standard output unless the **-o** option is used.

-fPIC
: Generate PIC code.

-fpic
: Tells C compiler to generate PIC code and tells assembler that PIC code has been generated.

-g
: Send **-g** flag to ccom(1) to create debug output. This unsets the **-O** option.

-I *path*
: Passed to the cpp(1) preprocessor to add header search directory to override system defaults.

-include *file*
: Tells the cpp(1) preprocessor to include the *file* during preprocessing.

-isystem *path*
: Defines *path* as a system header directory for the cpp(1) preprocessor.

-k
: Generate PIC code. See **-fpic** option.

-L

-M Pass **-M** flag to cpp(1) to generate dependencies for make(1).

-moption
Target-dependent option.

-nostartfiles
Do not link with the system startup files (crt0.c, etc)

-nostdinc
Do not use the system include paths (/usr/include, etc)

-nostdlib
Do not link with the system C library (libc).

-O Enable optimizations. Currently passes **-xdeljumps** and **-xtemps** to ccom(1). Note: this is unset if **-g** option is set.

-o *outfile*
Save result to *outfile*.

-P TODO: what is this?

-pg Not implemented.

-pthread
Defines **_PTHREADS** preprocessor directive for cpp(1). Uses **-lpthread** for ld(1) linker.

-S Stop after compilation by ccom(1). Do not assemble and do not link. The resulting assembler-language output is saved as a filename with a ".s" suffix unless the **-o** option is used. Note: cannot be combined with **-o** if multiple files are given.

-static
Do not use dynamic linkage. By default, it will link using the dynamic linker options and/or shared objects for the platform.

-t Passes **-t** to cpp(1) for traditional C preprocessor syntax.

-U *macro*
Passes to the cpp(1) preprocessor to remove the initial macro definition.

-v Outputs the version of **cc** and shows what commands will be ran with their command line arguments.

-Wl *flags*
Options for the linker

-X Don't remove temporary files on exit.

-x TODO

Predefined Macros

A few macros are predefined by **cc** when sent to cpp(1).

__PCC__ Set to the major version of pcc(1). These macros can be used to select code based on pcc(1) compatibility. See **-v** option.

__PCC_MINOR__ Set to the minor version.

__PCC_MINORMINOR__ Set to the minor-minor version -- the number after the minor version.

_PTHREADS Defined when **-pthread** switch is used.

Also system- and/or machine-dependent macros may also be predefined; for example: __NetBSD__, __ELF__, and __i386__.

SEE ALSO

as(1), ccom(1), cpp(1), ld(1)

HISTORY

The **cc** command comes from the original Portable C Compiler by S. C. Johnson, written in the late 70's.

This product includes software developed or owned by Caldera International, Inc.

NAME

ccom — C compiler

SYNOPSIS

ccom [**-gs**] [**-W** *flags*] [**-X** *flags*] [**-x** *optimizations*] [**-Z** *flags*] [infile] [outfile]

DESCRIPTION

The **ccom** utility provides a C compiler. The frontend is usually pcc(1). It is **not** intended to be run directly.

ccom reads the C source from *infile* or standard input and writes the assembler source to *outfile* or to standard output.

The options are as follows:

-g Enable debugging.

-k Generate PIC code.

-s Print statistics to standard error when complete. This includes: name table entries, name string size, permanent allocated memory, temporary allocated memory, lost memory, argument list unions, dimension/function unions, struct/union/enum blocks, inline node count, inline control blocks, and permanent symtab entries.

-v Display version.

-W *flags*

Report warnings. (Do some basic checks.) NOTE! These are subject to change RSN! *flags* is one or more of the following:

implicit
Implies **implicit-function-declaration** and **implicit-int**.

implicit-function-declaration
Report if no prototype for function.

implicit-int
TODO

missing-prototypes
TODO

strict-prototypes
TODO

-X *flags*

C specific debugging where *flags* is one or more of the following:

b Building of parse trees

d Declarations (multiple **d** flags gives more output)

e Pass1 trees at exit

i Initializations

n Memory allocations

o Turn off optimisations

p Prototypes

s Inlining

t Type conversions

x Target-specific flag, used in machine-dependent code

-x *optimizations*

optimizations is one or more of the following:

deljumps Delete redundant jumps and dead code.

ssa Convert statements into SSA form for optimization. Not yet finished.

tailcall Currently not implemented.

temps Setting this flag allows variables to be put into registers, for further optimization by the register allocator.

-Z *flags*

Code generator (pass2) specific debugging where *flags* is one or more of the following:

b Basic block and SSA building

c Code printout

e Trees when entering pass2

f Instruction matcher, may provide much output

n Memory allocation

o Instruction generator

r Register allocator

s Shape matching in instruction generator

t Type matching in instruction generator

u Sethi-Ullman computations

x Target-specific flag, used in machine-dependent code

SEE ALSO

as(1), cpp(1), pcc(1)

HISTORY

The **ccom** compiler is based on the original Portable C Compiler by S. C. Johnson, written in the late 70's. Even though much of the compiler has been rewritten, some of the basics still remain. About 50% of the frontend code and 80% of the backend code has been rewritten. Most is written by Anders Magnusson, with the exception of the data-flow analysis part and the SSA conversion code which is written by Peter A Jonsson, and the Mips port that were written as part of a project by undergraduate students at Lulea University of Technology.

This product includes software developed or owned by Caldera International, Inc.

NAME

cd — change working directory

SYNOPSIS

cd *directory*

DESCRIPTION

Directory is an absolute or relative pathname which becomes the new working directory. The interpretation of a relative pathname by cd depends on the CDPATH environment variable (see below).

ENVIRONMENT

The following environment variables affect the execution of cd:

CDPATH If the *directory* operand does not begin with a slash (/) character, and the first component is not dot (.) or dot-dot (..), **cd** searches for the directory relative to each directory named in the CDPATH variable, in the order listed. The new working directory is set to the first matching directory found. An empty string in place of a directory pathname represents the current directory. If the new working directory was derived from CDPATH, it will be printed to the standard output.

HOME If **cd** is invoked without arguments and the HOME environment variable exists and contains a directory name, that directory becomes the new working directory.

See csh(1) for more information on environment variables.

The **cd** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

csh(1), pwd(1), sh(1), chdir(2)

STANDARDS

The **cd** command is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

cdplay — compact disc player

SYNOPSIS

cdplay [**-a** *audio device*] [**-f** *device*] [*command ...*]

DESCRIPTION

cdplay is a program to control the audio features of a CD-ROM drive.

If no command is given, then **cdplay** enters interactive mode, reading commands from the standard input.

The following options are available:

-a *audio device*
: Specify the audio device to use. Used only in digital transfer mode. If not specified, the environment variables AUDIODEV and SPEAKER will be tried (in this order) to find the device; as a last resort, /dev/sound will be used.

-f *device*
: Specify the control device to use. Both absolute paths and paths relative to /dev are accepted. The suffix 'c' (or 'd' on some architectures, see disklabel(8) for details) is added to the device name if needed. If the device not specified, the environment variables MUSIC_CD, CD_DRIVE, DISC and CDPLAY will be tried (in this order) to find the device.

The available commands are listed below. Only as many characters as are required to uniquely identify a command need be specified. The word *play* can be omitted in all cases.

play [*first_track* [*last_track*]]
: Play from track *first_track* to track *last_track*. The first track has number 1.

play [*start_m:start_s.start_f* [*end_m:end_s.end_f*]]
: Play from the absolute address (MSF) defined by *start_m* in minutes, *start_s*, in seconds and *start_f* (frame number) to the absolute address defined by *end_m* in minutes, *end_s*, in seconds and *end_f* (frame number). Minutes are in the range 0-99. Seconds are in the range 0-59. Frame numbers are in the range 0-74.

play [*#start_block* [length]]
: Play starting from the logical block *start_block* using *length* logical blocks.

next Skip to the next track.

prev Skip to the previous track.

pause Stop playing. Do not stop the disc.

resume Resume playing. Used after the *pause* command.

shuffle Select shuffle play. Only valid in interactive mode.

single [*track*]
: Pick a single track and play it repeatedly. If a *track* argument is not given the **single** command will shut shuffle mode off and play the disc normally. Only valid in interactive mode.

skip Skip to another track. Only valid when shuffle play is selected.

stop Stop the disc.

eject Eject the disc.

close Inject the disc.

volume *left_channel right_channel*
Set the volume of left channel to *left_channel* and the volume of right channel to *right_channel*. Allowed values are in the range 0-255.

volume *value*
Set the volume of both left channel right channel to *value*. Allowed values are in the range 0-255.

volume mute
Turn the sound off.

volume mono
Set the mono mode.

volume stereo
Set the stereo mode.

volume left
Play the left subtrack on both left and right channels.

volume right
Play the right subtrack on both left and right channels.

info Print the table of contents.

status Display the current audio, media and volume status.

digital *n*
Turn on digital transfer mode. In this mode, **cdplay** reads digital data from disc and sends it to an audio device. A SCSI or ATAPI CD-ROM is required; and the audio device must support CD audio format (44100 Hz sampling rate, 16-bit stereo samples). Audio data are read and written in groups of *n* frames (5 by default, or 1/15 seconds).

help Print the list of available commands.

reset Perform the hardware reset of the device.

set msf Set minute-second-frame ioctl mode (default).

set lba Set LBA ioctl mode.

quit Quit the program.

FILES

`/dev/cd??`
`/dev/mcd??`

SEE ALSO

`disklabel`(8), `mscdlabel`(8)

HISTORY

The **cdplay** command first appeared in NetBSD 1.5. Support for digital transfer mode was added in NetBSD 4.0.

NAME

checknr — check nroff/troff files

SYNOPSIS

checknr [**-fs**] [**-a***.x1.y1.x2.y2. . . . .xn.yn*] [**-c***.x1.x2.x3 . . . .xn*] *file*

DESCRIPTION

checknr checks a list of nroff(1) or troff(1) input files for certain kinds of errors involving mismatched opening and closing delimiters and unknown commands. If no files are specified, **checknr** checks the standard input.

Recognized options are:

- **-a** Add additional pairs of macros to the list of known macros. This must be followed by groups of six characters, each group defining a pair of macros. The six characters are a period, the first macro name, another period, and the second macro name. For example, to define a pair .BS and .ES, use '-a.BS.ES'.
- **-c** Define commands which would otherwise be complained about as undefined.
- **-f** Request **checknr** to ignore '\f' font changes.
- **-s** Ignore '\s' size changes.

Delimiters checked are:

1. Font changes using \fx ... \fP.
2. Size changes using \sx ... \s0.
3. Macros that come in open ... close forms, for example, the .TS and .TE macros which must always come in pairs.

checknr is intended for use on documents that are prepared with **checknr** in mind, much the same as lint(1). It expects a certain document writing style for '\f' and '\s' commands, in that each \fx must be terminated with \fP and each \sx must be terminated with \s0. While it will work to directly go into the next font or explicitly specify the original font or point size, and many existing documents actually do this, such a practice will produce complaints from **checknr**. Since it is probably better to use the \fP and \s0 forms anyway, you should think of this as a contribution to your document preparation style.

checknr knows about the ms(7) and me(7) macro packages, as well as the macros from mdoc(7).

DIAGNOSTICS

Complaints about unmatched delimiters. Complaints about unrecognized commands. Various complaints about the syntax of commands.

SEE ALSO

nroff(1), troff(1), mdoc(7), me(7), ms(7)

HISTORY

The **checknr** command appeared in 4.0BSD. Basic mdoc(7) support appeared in NetBSD 1.6.

BUGS

There is no way to define a 1 character macro name using **-a**.

Does not correctly recognize certain reasonable constructs, such as conditionals.

mdoc(7) macros that are not at the beginning of the line are not recognized. Among others, this results in too many Unmatched Zz errors.

NAME

chflags — change file flags

SYNOPSIS

chflags [**-R** [**-H** | **-L** | **-P**]] [**-h**] *flags file ...*

DESCRIPTION

The **chflags** utility modifies the file flags of the listed files as specified by the *flags* operand.

The options are as follows:

-H If the **-R** option is specified, symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.)

-L If the **-R** option is specified, all symbolic links are followed.

-P If the **-R** option is specified, no symbolic links are followed.

-R Change the file flags for the file hierarchies rooted in the files instead of just the files themselves.

-h If the *file* or a file encountered during directory traversal is a symbolic link, the file flags of the link itself is changed.

Flags are a comma separated list of keywords. The following keywords are currently defined:

```
arch    set the archived flag (super-user only)
opaque  set the opaque flag (owner or super-user only)
nodump  set the nodump flag (owner or super-user only)
sappnd  set the system append-only flag (super-user only)
schg    set the system immutable flag (super-user only)
uappnd  set the user append-only flag (owner or super-user only)
uchg    set the user immutable flag (owner or super-user only)
```

Putting the letters “no” before an option causes the flag to be turned off. For example:

```
nouchg  the immutable bit should be cleared
```

The **-H**, **-L** and **-P** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command’s actions are determined by the last one specified.

The **-o** option of ls(1) is used to display the flags.

The **chflags** utility exits 0 on success, and >0 if an error occurs.

The kernel does not allow the flags on block and character devices to be changed except by the super-user.

SEE ALSO

ls(1), chflags(2), lchflags(2), stat(2), fts(3), symlink(7), dump(8), init(8)

NAME

chgrp — change group

SYNOPSIS

chgrp [**-R** [**-H** | **-L** | **-P**]] [**-fhv**] *group files ...*

DESCRIPTION

The **chgrp** utility sets the group ID of the file named by each *file* operand to the *group* ID specified by the group operand.

Options:

-H If the **-R** option is specified, symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.)

-L If the **-R** option is specified, all symbolic links are followed.

-P If the **-R** option is specified, no symbolic links are followed.

-R Change the group ID for the file hierarchies rooted in the files instead of just the files themselves.

-f The force option ignores errors, except for usage errors and doesn't query about strange modes (unless the user does not have proper permissions).

-h If *file* is a symbolic link, the group of the link is changed.

-v Cause **chgrp** to be verbose, showing files as they are processed.

If **-h** is not given, unless the **-H** or **-L** option is set, **chgrp** on a symbolic link always succeeds and has no effect. The **-H**, **-L** and **-P** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command's actions are determined by the last one specified.

The *group* operand can be either a group name from the group database, or a numeric group ID. Since it is valid to have a group name that is numeric (and doesn't have the numeric ID that matches its name) the name lookup is always done first. Preceding the ID with a ''#'' character will force it to be taken as a number.

The user invoking **chgrp** must belong to the specified group and be the owner of the file, or be the super-user.

Unless invoked by the super-user, **chgrp** clears the set-user-id and set-group-id bits on a file to prevent accidental or mischievous creation of set-user-id or set-group-id programs.

The **chgrp** utility exits 0 on success, and >0 if an error occurs.

FILES

`/etc/group` Group ID file

SEE ALSO

chown(2), lchown(2), fts(3), group(5), passwd(5), symlink(7), chown(8)

STANDARDS

The **chgrp** utility is expected to be POSIX 1003.2 compatible.

The **-v** option and the use of ''#'' to force a numeric group ID are extensions to IEEE Std 1003.2 ("POSIX.2").

NAME

chio — medium changer control utility

SYNOPSIS

chio [**-f** *changer*] *command arg1 arg2* [*arg3* [...]]

DESCRIPTION

chio is used to control the operation of medium changers, such as those found in tape and optical disk juke-boxes.

The options are as follows:

-f *changer*
: Use the device changer rather than the default device /dev/ch0.

The default changer may be overridden by setting the environment variable CHANGER to the desired changer device.

A medium changer apparatus is made up of elements. There are four element types: picker (medium transport), slot (storage), portal (import/export), and drive (data transfer). In this command description, the shorthand **ET** will be used to represent an element type, and **EU** will be used to represent an element unit. For example, to represent the first robotic arm in the changer, the ET would be "picker" and the EU would be "0".

SUPPORTED COMMANDS

chio move <*from ET*> <*from EU*> <*to ET*> <*to EU*> [*inv*]

Moves the media unit from <from ET/EU> to <to ET/EU>. If the optional modifier inv is specified, the media unit will be inverted before insertion.

chio exchange <*src ET*> <*src EU*> <*dst1 ET*> <*dst1 EU*> [<*dst2 ET*> <*dst2 ET*>] [*inv1*] [*inv2*]

Performs a media unit exchange operation. The media unit in <src ET/EU> is moved to <dst1 ET/EU> and the media unit previously in <dst1 ET/EU> is moved to <dst2 ET/EU>. In the case of a simple exchange, <dst2 ET/EU> is omitted and the values <src ET/EU> are used in their place. The optional modifiers inv1 and inv2 specify whether the media units are to be inverted before insertion into <dst1 ET/EU> and <dst2 ET/EU> respectively.

Note that not all medium changers support the **exchange** operation; The changer must have multiple free pickers or emulate multiple free pickers with transient storage.

chio position <*to ET*> <*to EU*> [*inv*]

Position the picker in front of the element described by <to ET/EU>. If the optional modifier inv is specified, the media unit will be inverted before insertion.

Note that not all changers behave as expected when issued this command.

chio params

Report the number of slots, drives, pickers, and portals in the changer, and which picker unit the changer is currently configured to use.

chio getpicker

Report which picker unit the changer is currently configured to use.

chio setpicker <*unit*>

Configure the changer to use picker <unit>.

chio status [<*type*> [unit [count]]] [*voltags*]

Report the status of all elements in the changer. If <type> is specified, report the status of all elements of type <type>.

The status bits are defined as follows:

FULL Element contains a media unit.

IMPEXP Media was deposited into element by an outside human operator.

EXCEPT Element is in an abnormal state.

ACCESS Media in this element is accessible by a picker.

EXENAB Element supports passing media (exporting) to an outside human operator.

INENAB Element supports receiving media (importing) from an outside human operator.

If the element is a drive, the device name of the drive will be reported if it is available.

If the [*voltags*] option is specified, primary and alternate volume tag information will be reported, if available.

If the previous location of the media is available, it will also be reported.

chio ielem

Perform an *INITIALIZE ELEMENT STATUS* operation on the changer.

chio cdlu <*sub-command*> <*slot*>

This command is provided for controlling CD-ROM changer mechanisms which cannot use the standard changer control interface. ATAPI CD-ROM changers fall into this category. There are 3 sub-commands:

load Loads the media from the specified slot into the CD-ROM drive.

unload Unloads the media from the CD-ROM drive and returns it to the specified slot.

abort Aborts any pending load or unload operation.

FILES

/dev/ch0 - default changer device

EXAMPLES

```
chio -f /dev/ch0 move slot 3 drive 0
```

Moves the media in slot 3 (fourth slot) to drive 0 (first drive).

```
chio setpicker 2
```

Configures the changer to use picker 2 (third picker) for operations.

```
chio -f /dev/cd0a cdlu load 1
```

Loads the media from slot (second slot) into the CD-ROM drive.

```
chio -f /dev/ch1 status
```

Returns status of all elements in the second changer.

SEE ALSO

`mt`(1), `mount`(8)

AUTHORS

The **`chio`** program and SCSI changer driver were originally written by Jason R. Thorpe for And Communications, http://www.and.com/. Additional development was done by Jason R. Thorpe for the Numerical Aerospace Simulation Facility, NASA Ames Research Center.

NAME

chmod — change file modes

SYNOPSIS

chmod [**-R** [**-H** | **-L** | **-P**]] [**-h**] *mode file ...*

DESCRIPTION

The **chmod** utility modifies the file mode bits of the listed files as specified by the *mode* operand.

The options are as follows:

-H If the **-R** option is specified, symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.)

-L If the **-R** option is specified, all symbolic links are followed.

-P If the **-R** option is specified, no symbolic links are followed.

-R Change the modes of the file hierarchies rooted in the files instead of just the files themselves.

-h If *file* is symbolic link, the mode of the link is changed.

The **-H**, **-L** and **-P** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command's actions are determined by the last one specified.

Only the owner of a file or the super-user is permitted to change the mode of a file.

EXIT STATUS

The **chmod** utility exits 0 on success, and >0 if an error occurs.

MODES

Modes may be absolute or symbolic. An absolute mode is an octal number constructed by *or*'ing the following values:

4000	set-user-ID-on-execution
2000	set-group-ID-on-execution
1000	sticky bit, see chmod(2)
0400	read by owner
0200	write by owner
0100	execute (or search for directories) by owner
0070	read, write, execute/search by group
0007	read, write, execute/search by others

The read, write, and execute/search values for group and others are encoded as described for owner.

The symbolic mode is described by the following grammar:

```
mode         ::= clause [, clause ...]
clause       ::= [who ...] [action ...] last_action
action       ::= op [perm ...]
last_action  ::= op [perm ...]
who          ::= a | u | g | o
op           ::= + | - | =
perm         ::= r | s | t | w | x | X | u | g | o
```

The *who* symbols ''u'', ''g'', and ''o'' specify the user, group, and other parts of the mode bits, respectively. The *who* symbol ''a'' is equivalent to ''ugo''.

The *perm* symbols represent the portions of the mode bits as follows:

r	The read bits.
s	The set-user-ID-on-execution and set-group-ID-on-execution bits.
t	The sticky bit.
w	The write bits.
x	The execute/search bits.
X	The execute/search bits if the file is a directory or any of the execute/search bits are set in the original (unmodified) mode. Operations with the *perm* symbol ‘‘X’’ are only meaningful in conjunction with the *op* symbol ‘‘+’’, and are ignored in all other cases.
u	The user permission bits in the mode of the original file.
g	The group permission bits in the mode of the original file.
o	The other permission bits in the mode of the original file.

The *op* symbols represent the operation performed, as follows:

+ If no value is supplied for *perm*, the ‘‘+’’ operation has no effect. If no value is supplied for *who*, each permission bit specified in *perm*, for which the corresponding bit in the file mode creation mask is clear, is set. Otherwise, the mode bits represented by the specified *who* and *perm* values are set.

− If no value is supplied for *perm*, the ‘‘−’’ operation has no effect. If no value is supplied for *who*, each permission bit specified in *perm*, for which the corresponding bit in the file mode creation mask is clear, is cleared. Otherwise, the mode bits represented by the specified *who* and *perm* values are cleared.

= The mode bits specified by the *who* value are cleared, or, if no who value is specified, the owner, group and other mode bits are cleared. Then, if no value is supplied for *who*, each permission bit specified in *perm*, for which the corresponding bit in the file mode creation mask is clear, is set. Otherwise, the mode bits represented by the specified *who* and *perm* values are set.

Each *clause* specifies one or more operations to be performed on the mode bits, and each operation is applied to the mode bits in the order specified.

Operations upon the other permissions only (specified by the symbol ‘‘o’’ by itself), in combination with the *perm* symbols ‘‘s’’ or ‘‘t’’, are ignored.

EXAMPLES

`644`	make a file readable by anyone and writable by the owner only.
`go-w`	deny write permission to group and others.
`=rw,+X`	set the read and write permissions to the usual defaults, but retain any execute permissions that are currently set.
`+X`	make a directory or file searchable/executable by everyone if it is already searchable/executable by anyone.

`755`
`u=rwx,go=rx`
`u=rwx,go=u-w`
	make a file readable/executable by everyone and writable by the owner only.

`go=`	clear all mode bits for group and others.
`g=u-w`	set the group bits equal to the user bits, but clear the group write bit.

SEE ALSO

`chflags`(1), `install`(1), `chmod`(2), `stat`(2), `umask`(2), `fts`(3), `setmode`(3), `symlink`(7), `chown`(8)

STANDARDS

The **chmod** utility is expected to be IEEE Std 1003.2-1992 (“POSIX.2”) compatible with the exception of the *perm* symbol “t” which is not included in that standard.

BUGS

There’s no *perm* option for the naughty bits.

NAME

chpass, **chfn**, **chsh** — add or change user database information

SYNOPSIS

chpass [**-a** *list*] [**-s** *newshell*] [**-l**] [user]
chpass [**-a** *list*] [**-s** *newshell*] [**-y**] [user]

DESCRIPTION

chpass allows editing of the user database information associated with *user* or, by default, the current user. The information is formatted and supplied to an editor for changes.

Only the information that the user is allowed to change is displayed.

The options are as follows:

-a The super-user is allowed to directly supply a user database entry, in the format specified by passwd(5), as an argument. This argument must be a colon (":") separated list of all the user database fields, although they may be empty.

-s The **-s** option attempts to change the user's shell to *newshell*.

-l This option causes the password to be updated only in the local password file. When changing only the local password, pwd_mkdb(8) is used to update the password databases.

-y This forces the YP password database entry to be changed, even if the user has an entry in the local database. The rpc.yppasswdd(8) daemon should be running on the YP master server.

Possible display items are as follows:

Login:	user's login name
Password:	user's encrypted password
Uid:	user's login
Gid:	user's login group
Change:	password change time
Expire:	account expiration time
Class:	user's general classification
Home Directory:	user's home directory
Shell:	user's login shell
Full Name:	user's real name
Location:	user's normal location
Home Phone:	user's home phone
Office Phone:	user's office phone

The *login* field is the user name used to access the computer account.

The *password* field contains the encrypted form of the user's password.

The *uid* field is the number associated with the *login* field. Both of these fields should be unique across the system (and often across a group of systems) as they control file access.

While it is possible to have multiple entries with identical login names and/or identical user id's, it is usually a mistake to do so. Routines that manipulate these files will often return only one of the multiple entries, and that one by random selection.

The *group* field is the group that the user will be placed in at login. Since BSD supports multiple groups (see groups(1)) this field currently has little special meaning. This field may be filled in with either a number or a group name (see group(5)).

The *change* field is the date by which the password must be changed.

The *expire* field is the date on which the account expires.

Both the *change* and *expire* fields should be entered in the form "month day year" where *month* is the month name (the first three characters are sufficient), *day* is the day of the month, and *year* is the year.

The *class* field is a key for a user's login class. Login classes are defined in login.conf(5), which is a termcap(5) style database of user attributes, accounting, resource and environment settings.

The user's *home directory* is the full UNIX path name where the user will be placed at login.

The *shell* field is the command interpreter the user prefers. If the *shell* field is empty, the Bourne shell, /bin/sh, is assumed. When altering a login shell, and not the super-user, the user may not change from a non-standard shell or to a non-standard shell. Non-standard is defined as a shell not found in /etc/shells.

The last four fields are for storing the user's *full name*, *office location*, and *home* and *work telephone* numbers.

Once the information has been verified, **chpass** uses pwd_mkdb(8) to update the user database.

ENVIRONMENT

The vi(1) editor will be used unless the environment variable EDITOR is set to an alternative editor. When the editor terminates, the information is re-read and used to update the user database itself. Only the user, or the super-user, may edit the information associated with the user.

FILES

/etc/master.passwd	The user database
/etc/passwd	A Version 7 format password file
/etc/ptmp	Lock file for the passwd database
/tmp/pw.XXXXXX	Temporary copy of the user passwd information
/etc/shells	The list of approved shells

SEE ALSO

finger(1), login(1), passwd(1), pwhash(1), getusershell(3), passwd(5), passwd.conf(5), pwd_mkdb(8), vipw(8)

Robert Morris and Ken Thompson, *UNIX Password Security*.

HISTORY

The **chpass** command appeared in 4.3BSD−Reno.

BUGS

This program's interface is poorly suited to cryptographic systems such as Kerberos, and consequently Kerberos password changing is not a feature of this program.

User information should (and eventually will) be stored elsewhere.

NAME

ci − check in RCS revisions

SYNOPSIS

ci [*options*] *file* . . .

DESCRIPTION

ci stores new revisions into RCS files. Each pathname matching an RCS suffix is taken to be an RCS file. All others are assumed to be working files containing new revisions. **ci** deposits the contents of each working file into the corresponding RCS file. If only a working file is given, **ci** tries to find the corresponding RCS file in an RCS subdirectory and then in the working file's directory. For more details, see FILE NAMING below.

For **ci** to work, the caller's login must be on the access list, except if the access list is empty or the caller is the superuser or the owner of the file. To append a new revision to an existing branch, the tip revision on that branch must be locked by the caller. Otherwise, only a new branch can be created. This restriction is not enforced for the owner of the file if non-strict locking is used (see **rcs**(1)). A lock held by someone else can be broken with the **rcs** command.

Unless the **−f** option is given, **ci** checks whether the revision to be deposited differs from the preceding one. If not, instead of creating a new revision **ci** reverts to the preceding one. To revert, ordinary **ci** removes the working file and any lock; **ci −l** keeps and **ci −u** removes any lock, and then they both generate a new working file much as if **co −l** or **co −u** had been applied to the preceding revision. When reverting, any **−n** and **−s** options apply to the preceding revision.

For each revision deposited, **ci** prompts for a log message. The log message should summarize the change and must be terminated by end-of-file or by a line containing **.** by itself. If several files are checked in **ci** asks whether to reuse the previous log message. If the standard input is not a terminal, **ci** suppresses the prompt and uses the same log message for all files. See also **−m**.

If the RCS file does not exist, **ci** creates it and deposits the contents of the working file as the initial revision (default number: **1.1**). The access list is initialized to empty. Instead of the log message, **ci** requests descriptive text (see **−t** below).

The number *rev* of the deposited revision can be given by any of the options **−f**, **−i**, **−I**, **−j**, **−k**, **−l**, **−M**, **−q**, **−r**, or **−u**. *rev* can be symbolic, numeric, or mixed. Symbolic names in *rev* must already be defined; see the **−n** and **−N** options for assigning names during checkin. If *rev* is **$**, **ci** determines the revision number from keyword values in the working file.

If *rev* begins with a period, then the default branch (normally the trunk) is prepended to it. If *rev* is a branch number followed by a period, then the latest revision on that branch is used.

If *rev* is a revision number, it must be higher than the latest one on the branch to which *rev* belongs, or must start a new branch.

If *rev* is a branch rather than a revision number, the new revision is appended to that branch. The level number is obtained by incrementing the tip revision number of that branch. If *rev* indicates a non-existing branch, that branch is created with the initial revision numbered *rev***.1**.

If *rev* is omitted, **ci** tries to derive the new revision number from the caller's last lock. If the caller has locked the tip revision of a branch, the new revision is appended to that branch. The new revision number is obtained by incrementing the tip revision number. If the caller locked a non-tip revision, a new branch is started at that revision by incrementing the highest branch number at that revision. The default initial branch and level numbers are **1**.

If *rev* is omitted and the caller has no lock, but owns the file and locking is not set to *strict*, then the revision is appended to the default branch (normally the trunk; see the **−b** option of **rcs**(1)).

Exception: On the trunk, revisions can be appended to the end, but not inserted.

OPTIONS

−r*rev* Check in revision *rev*.

−r The bare **−r** option (without any revision) has an unusual meaning in **ci**. With other RCS commands, a bare **−r** option specifies the most recent revision on the default branch, but with **ci**, a bare **−r** option reestablishes the default behavior of releasing a lock and removing the working file, and is used to override any default **−l** or **−u** options established by shell aliases or scripts.

−l[*rev*] works like **−r**, except it performs an additional **co −l** for the deposited revision. Thus, the deposited revision is immediately checked out again and locked. This is useful for saving a revision although one wants to continue editing it after the checkin.

−u[*rev*] works like **−l**, except that the deposited revision is not locked. This lets one read the working file immediately after checkin.

The **−l**, bare **−r**, and **−u** options are mutually exclusive and silently override each other. For example, **ci −u −r** is equivalent to **ci −r** because bare **−r** overrides **−u**.

−f[*rev*] forces a deposit; the new revision is deposited even it is not different from the preceding one.

−k[*rev*] searches the working file for keyword values to determine its revision number, creation date, state, and author (see **co**(1)), and assigns these values to the deposited revision, rather than computing them locally. It also generates a default login message noting the login of the caller and the actual checkin date. This option is useful for software distribution. A revision that is sent to several sites should be checked in with the **−k** option at these sites to preserve the original number, date, author, and state. The extracted keyword values and the default log message can be overridden with the options **−d**, **−m**, **−s**, **−w**, and any option that carries a revision number.

−q[*rev*] quiet mode; diagnostic output is not printed. A revision that is not different from the preceding one is not deposited, unless **−f** is given.

−i[*rev*] initial checkin; report an error if the RCS file already exists. This avoids race conditions in certain applications.

−j[*rev*] just checkin and do not initialize; report an error if the RCS file does not already exist.

−I[*rev*] interactive mode; the user is prompted and questioned even if the standard input is not a terminal.

−d[*date*]
uses *date* for the checkin date and time. The *date* is specified in free format as explained in **co**(1). This is useful for lying about the checkin date, and for **−k** if no date is available. If *date* is empty, the working file's time of last modification is used.

−M[*rev*]
Set the modification time on any new working file to be the date of the retrieved revision. For example, **ci −d −M −u** *f* does not alter *f*'s modification time, even if *f*'s contents change due to keyword substitution. Use this option with care; it can confuse **make**(1).

−m*msg* uses the string *msg* as the log message for all revisions checked in. By convention, log messages that start with **#** are comments and are ignored by programs like GNU Emacs's **vc** package. Also, log messages that start with **{***clumpname***}** (followed by white space) are meant to be clumped together if possible, even if they are associated with different files; the **{***clumpname***}** label is used only for clumping, and is not considered to be part of the log message itself.

−n*name*
assigns the symbolic name *name* to the number of the checked-in revision. **ci** prints an error message if *name* is already assigned to another number.

−N*name*
same as **−n**, except that it overrides a previous assignment of *name*.

−s*state* sets the state of the checked-in revision to the identifier *state*. The default state is **Exp**.

−t *file* writes descriptive text from the contents of the named *file* into the RCS file, deleting the existing text. The *file* cannot begin with **−**.

−t−*string*

Write descriptive text from the *string* into the RCS file, deleting the existing text.

The **−t** option, in both its forms, has effect only during an initial checkin; it is silently ignored otherwise.

During the initial checkin, if **−t** is not given, **ci** obtains the text from standard input, terminated by end-of-file or by a line containing **.** by itself. The user is prompted for the text if interaction is possible; see **−I**.

For backward compatibility with older versions of RCS, a bare **−t** option is ignored.

−T Set the RCS file's modification time to the new revision's time if the former precedes the latter and there is a new revision; preserve the RCS file's modification time otherwise. If you have locked a revision, **ci** usually updates the RCS file's modification time to the current time, because the lock is stored in the RCS file and removing the lock requires changing the RCS file. This can create an RCS file newer than the working file in one of two ways: first, **ci −M** can create a working file with a date before the current time; second, when reverting to the previous revision the RCS file can change while the working file remains unchanged. These two cases can cause excessive recompilation caused by a **make**(1) dependency of the working file on the RCS file. The **−T** option inhibits this recompilation by lying about the RCS file's date. Use this option with care; it can suppress recompilation even when a checkin of one working file should affect another working file associated with the same RCS file. For example, suppose the RCS file's time is 01:00, the (changed) working file's time is 02:00, some other copy of the working file has a time of 03:00, and the current time is 04:00. Then **ci −d −T** sets the RCS file's time to 02:00 instead of the usual 04:00; this causes **make**(1) to think (incorrectly) that the other copy is newer than the RCS file.

−w*login*

uses *login* for the author field of the deposited revision. Useful for lying about the author, and for **−k** if no author is available.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*

specifies the suffixes for RCS files. A nonempty suffix matches any pathname ending in the suffix. An empty suffix matches any pathname of the form **RCS/***path* or *path1***RCS/***path2*. The **−x** option can specify a list of suffixes separated by /. For example, **−x,v/** specifies two suffixes: **,v** and the empty suffix. If two or more suffixes are specified, they are tried in order when looking for an RCS file; the first one that works is used for that file. If no RCS file is found but an RCS file can be created, the suffixes are tried in order to determine the new RCS file's name. The default for *suffixes* is installation-dependent; normally it is **,v/** for hosts like UNIX that permit commas in filenames, and is empty (i.e. just the empty suffix) for other hosts.

−z*zone* specifies the date output format in keyword substitution, and specifies the default time zone for *date* in the **−d***date* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

The **−z** option does not affect dates stored in RCS files, which are always UTC.

FILE NAMING

Pairs of RCS files and working files can be specified in three ways (see also the example section).

1) Both the RCS file and the working file are given. The RCS pathname is of the form *path1*/*workfileX* and the working pathname is of the form *path2*/*workfile* where *path1*/ and *path2*/ are (possibly different or empty) paths, *workfile* is a filename, and *X* is an RCS suffix. If *X* is empty, *path1*/ must start with **RCS/** or must contain **/RCS/**.

2) Only the RCS file is given. Then the working file is created in the current directory and its name is derived from the name of the RCS file by removing *path1*/ and the suffix *X*.

3) Only the working file is given. Then **ci** considers each RCS suffix *X* in turn, looking for an RCS file of the form *path2*/**RCS**/*workfileX* or (if the former is not found and *X* is nonempty) *path2*/*workfileX*.

If the RCS file is specified without a path in 1) and 2), **ci** looks for the RCS file first in the directory **./RCS** and then in the current directory.

ci reports an error if an attempt to open an RCS file fails for an unusual reason, even if the RCS file's pathname is just one of several possibilities. For example, to suppress use of RCS commands in a directory *d*, create a regular file named *d*/**RCS** so that casual attempts to use RCS commands in *d* fail because *d*/**RCS** is not a directory.

EXAMPLES

Suppose **,v** is an RCS suffix and the current directory contains a subdirectory **RCS** with an RCS file **io.c,v**. Then each of the following commands check in a copy of **io.c** into **RCS/io.c,v** as the latest revision, removing **io.c**.

```
ci io.c;   ci RCS/io.c,v;   ci io.c,v;
ci io.c RCS/io.c,v;   ci io.c io.c,v;
ci RCS/io.c,v io.c;   ci io.c,v io.c;
```

Suppose instead that the empty suffix is an RCS suffix and the current directory contains a subdirectory **RCS** with an RCS file **io.c**. The each of the following commands checks in a new revision.

```
ci io.c;   ci RCS/io.c;
ci io.c RCS/io.c;
ci RCS/io.c io.c;
```

FILE MODES

An RCS file created by **ci** inherits the read and execute permissions from the working file. If the RCS file exists already, **ci** preserves its read and execute permissions. **ci** always turns off all write permissions of RCS files.

FILES

Temporary files are created in the directory containing the working file, and also in the temporary directory (see **TMPDIR** under **ENVIRONMENT**). A semaphore file or files are created in the directory containing the RCS file. With a nonempty suffix, the semaphore names begin with the first character of the suffix; therefore, do not specify an suffix whose first character could be that of a working filename. With an empty suffix, the semaphore names end with **_** so working filenames should not end in **_**.

ci never changes an RCS or working file. Normally, **ci** unlinks the file and creates a new one; but instead of breaking a chain of one or more symbolic links to an RCS file, it unlinks the destination file instead. Therefore, **ci** breaks any hard or symbolic links to any working file it changes; and hard links to RCS files are ineffective, but symbolic links to RCS files are preserved.

The effective user must be able to search and write the directory containing the RCS file. Normally, the real user must be able to read the RCS and working files and to search and write the directory containing the working file; however, some older hosts cannot easily switch between real and effective users, so on these hosts the effective user is used for all accesses. The effective user is the same as the real user unless your copies of **ci** and **co** have setuid privileges. As described in the next section, these privileges yield extra security if the effective user owns all RCS files and directories, and if only the effective user can write RCS directories.

Users can control access to RCS files by setting the permissions of the directory containing the files; only users with write access to the directory can use RCS commands to change its RCS files. For example, in

hosts that allow a user to belong to several groups, one can make a group's RCS directories writable to that group only. This approach suffices for informal projects, but it means that any group member can arbitrarily change the group's RCS files, and can even remove them entirely. Hence more formal projects sometimes distinguish between an RCS administrator, who can change the RCS files at will, and other project members, who can check in new revisions but cannot otherwise change the RCS files.

SETUID USE

To prevent anybody but their RCS administrator from deleting revisions, a set of users can employ setuid privileges as follows.

- Check that the host supports RCS setuid use. Consult a trustworthy expert if there are any doubts. It is best if the **seteuid** system call works as described in POSIX 1003.1a Draft 5, because RCS can switch back and forth easily between real and effective users, even if the real user is **root**. If not, the second best is if the **setuid** system call supports saved setuid (the {_POSIX_SAVED_IDS} behavior of POSIX 1003.1-1990); this fails only if the real or effective user is **root**. If RCS detects any failure in setuid, it quits immediately.
- Choose a user *A* to serve as RCS administrator for the set of users. Only *A* can invoke the **rcs** command on the users' RCS files. *A* should not be **root** or any other user with special powers. Mutually suspicious sets of users should use different administrators.
- Choose a pathname *B* to be a directory of files to be executed by the users.
- Have *A* set up *B* to contain copies of **ci** and **co** that are setuid to *A* by copying the commands from their standard installation directory *D* as follows:

  mkdir *B*
  cp *D*/**c[io]** *B*
  chmod go−w,u+s *B*/**c[io]**

- Have each user prepend *B* to their path as follows:

  PATH=*B*:**$PATH; export PATH** # ordinary shell
  set path=(*B* **$path)** # C shell

- Have *A* create each RCS directory *R* with write access only to *A* as follows:

  mkdir *R*
  chmod go−w *R*

- If you want to let only certain users read the RCS files, put the users into a group *G*, and have *A* further protect the RCS directory as follows:

  chgrp *G R*
  chmod g−w,o−rwx *R*

- Have *A* copy old RCS files (if any) into *R*, to ensure that *A* owns them.
- An RCS file's access list limits who can check in and lock revisions. The default access list is empty, which grants checkin access to anyone who can read the RCS file. If you want limit checkin access, have *A* invoke **rcs −a** on the file; see **rcs**(1). In particular, **rcs −e −a***A* limits access to just *A*.
- Have *A* initialize any new RCS files with **rcs −i** before initial checkin, adding the **−a** option if you want to limit checkin access.
- Give setuid privileges only to **ci**, **co**, and **rcsclean**; do not give them to **rcs** or to any other command.
- Do not use other setuid commands to invoke RCS commands; setuid is trickier than you think!

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

TMPDIR

Name of the temporary directory. If not set, the environment variables **TMP** and **TEMP** are inspected instead and the first value found is taken; if none of them are set, a host-dependent default is used, typically **/tmp**.

DIAGNOSTICS

For each revision, **ci** prints the RCS file, the working file, and the number of both the deposited and the preceding revision. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

co(1), emacs(1), ident(1), make(1), rcs(1), rcsclean(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rlog(1), setuid(2), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

cksum, **md2**, **md4**, **md5**, **sha1**, **rmd160**, **sum** — display file checksums and block counts

SYNOPSIS

cksum [**-nw**][**-a** *algorithm* | **-c** [*file*]|[**-o** *1* | *2*]][*file ...*]
sum [**-w**][**-c** [*file*]][*file ...*]
md2 [**-nw**][**-c** [*file*]| **-p** | **-t** | **-x** | **-s** *string*][*file ...*]
md4 [**-nw**][**-c** [*file*]| **-p** | **-t** | **-x** | **-s** *string*][*file ...*]
md5 [**-nw**][**-c** [*file*]| **-p** | **-t** | **-x** | **-s** *string*][*file ...*]
sha1 [**-nw**][**-c** [*file*]| **-p** | **-t** | **-x** | **-s** *string*][*file ...*]
rmd160 [**-nw**][**-c** [*file*]| **-p** | **-t** | **-x** | **-s** *string*][*file ...*]

DESCRIPTION

The **cksum** utility writes to the standard output three whitespace separated fields for each input file. These fields are a checksum CRC, the total number of octets in the file and the file name. If no file name is specified, the standard input is used and no file name is written.

The **sum** utility is identical to the **cksum** utility, except that it defaults to using historic algorithm 1, as described below. It is provided for compatibility only.

The **md5** utility takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to product two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key encryption system such as RSA.

The **md2** and **md4** utilities behave in exactly the same manner as **md5** but use different algorithms.

The **sha1** and **rmd160** utilities also produce message digests, however the output from these two programs is 160 bits in length, as opposed to 128.

The options are as follows:

-a *algorithm*
: When invoked as **cksum**, use the specified *algorithm*. Valid algorithms are MD2, MD4, MD5, SHA1, RMD160, SHA256, SHA384, SHA512, CRC, old1, and old2. Old1 and old2 are equal to **-o** *1* and **-o** *2*, respectively. The default is CRC.

-c [*file*]
: Verify (check) files against a list of checksums. The list is read from *file*, or from stdin if no filename is given. E.g. first run

```
md5 *.tgz > MD5
sha1 *.tgz > SHA1
```

to generate a list of MD5 checksums in MD5, then use the following command to verify them:

```
cat MD5 SHA1 | cksum -c
```

If an error is found during checksum verification, an error message is printed, and the program returns an error code of 1.

-o
: Use historic algorithms instead of the (superior) default one.

Algorithm 1 is the algorithm used by historic BSD systems as the sum(1) algorithm and by historic AT&T System V UNIX systems as the sum(1) algorithm when using the **-r** option. This is a 16-bit checksum, with a right rotation before each addition; overflow is discarded.

Algorithm 2 is the algorithm used by historic AT&T System V UNIX systems as the default `sum`(1) algorithm. This is a 32-bit checksum, and is defined as follows:

```
s = sum of all bytes;
r = s % 2^16 + (s % 2^32) / 2^16;
cksum = (r % 2^16) + r / 2^16;
```

Both algorithm 1 and 2 write to the standard output the same fields as the default algorithm except that the size of the file in bytes is replaced with the size of the file in blocks. For historic reasons, the block size is 1024 for algorithm 1 and 512 for algorithm 2. Partial blocks are rounded up.

-w Print warnings about malformed checksum files when verifying checksums with **-c**.

The following options apply only when using the one of the message digest algorithms:

-n Print the hash and the filename in the normal sum output form, with the hash at the left and the filename following on the right.

-p Echo input from standard input to standard output, and append the selected message digest.

-s *string*
Print the hash of the given string *string*.

-t Run a built-in message digest time trial.

-x Run a built-in message digest test script. The tests that are run are supposed to encompass all the various tests in the suites that accompany the algorithms' descriptions with the exception of the last test for the SHA-1 algorithm and the RIPEMD-160 algorithm. The last test for these is one million copies of the lower letter a.

The default CRC used is based on the polynomial used for CRC error checking in the networking standard ISO/IEC 8802-3:1989. The CRC checksum encoding is defined by the generating polynomial:

```
G(x) = x^32 + x^26 + x^23 + x^22 + x^16 + x^12 +
  x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1
```

Mathematically, the CRC value corresponding to a given file is defined by the following procedure:

The n bits to be evaluated are considered to be the coefficients of a mod 2 polynomial M(x) of degree $n-1$. These n bits are the bits from the file, with the most significant bit being the most significant bit of the first octet of the file and the last bit being the least significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral number of octets, followed by one or more octets representing the length of the file as a binary value, least significant octet first. The smallest number of octets capable of representing this integer are used.

M(x) is multiplied by x^32 (i.e., shifted left 32 bits) and divided by G(x) using mod 2 division, producing a remainder R(x) of degree ≤ 31.

The coefficients of R(x) are considered to be a 32-bit sequence.

The bit sequence is complemented and the result is the CRC.

The **cksum** and **sum** utilities exit 0 on success, and >0 if an error occurs.

SEE ALSO

`openssl`(1), `mtree`(8)

The default calculation is identical to that given in pseudo-code in the following ACM article.

Dilip V. Sarwate, "Computation of Cyclic Redundancy Checks Via Table Lookup", *Communications of the ACM*, August 1988.

R. Rivest, *The MD2 Message-Digest Algorithm*, RFC 1319.

R. Rivest, *The MD4 Message-Digest Algorithm*, RFC 1186 and RFC 1320.

R. Rivest, *The MD5 Message-Digest Algorithm*, RFC 1321.

U.S. DOC/NIST, *Secure Hash Standard*, FIPS PUB 180-1.

STANDARDS

The **cksum** utility is expected to conform to IEEE Std 1003.2-1992 (“POSIX.2”).

HISTORY

The **cksum** utility appeared in 4.4BSD. The functionality for **md2**, **md4**, **sha1**, and **rmd160** was added in NetBSD 1.6. Support for **sha2** was added in NetBSD 3.0. The functionality to verify checksum stored in a file (**-c**) first appeared in NetBSD 4.0.

NAME

cmp − compare two files byte by byte

SYNOPSIS

cmp [*OPTION*]... *FILE1* [*FILE2* [*SKIP1* [*SKIP2*]]]

DESCRIPTION

Compare two files byte by byte.

−b −−print−bytes
Print differing bytes.

−i SKIP **−−ignore−initial**=*SKIP*
Skip the first SKIP bytes of input.

−i SKIP1:SKIP2 **−−ignore−initial**=*SKIP1*:SKIP2

Skip the first SKIP1 bytes of FILE1 and the first SKIP2 bytes of FILE2.

−l −−verbose
Output byte numbers and values of all differing bytes.

−n LIMIT **−−bytes**=*LIMIT*
Compare at most LIMIT bytes.

−s −−quiet −−silent
Output nothing; yield exit status only.

−v −−version
Output version info.

−−help Output this help.

SKIP1 and SKIP2 are the number of bytes to skip in each file. SKIP values may be followed by the following multiplicative suffixes: kB 1000, K 1024, MB 1,000,000, M 1,048,576, GB 1,000,000,000, G 1,073,741,824, and so on for T, P, E, Z, Y.

If a FILE is ‘-’ or missing, read standard input.

AUTHOR

Written by Torbjorn Granlund and David MacKenzie.

REPORTING BUGS

Report bugs to <bug-gnu-utils@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **cmp** is maintained as a Texinfo manual. If the **info** and **cmp** programs are properly installed at your site, the command

info diff

should give you access to the complete manual.

NAME

cmp — compare two files

SYNOPSIS

cmp [**-l** | **-s**] *file1 file2* [*skip1* [*skip2*]]

DESCRIPTION

The cmp utility compares two files of any type and writes the results to the standard output. By default, **cmp** is silent if the files are the same; if they differ, the byte and line number at which the first difference occurred is reported.

Bytes and lines are numbered beginning with one.

The following options are available:

-l Print the byte number (decimal) and the differing byte values (octal) for each difference.

-s Print nothing for differing files; return exit status only.

The optional arguments *skip1* and *skip2* are the byte offsets from the beginning of *file1* and *file2*, respectively, where the comparison will begin. The offset is decimal by default, but may be expressed as an hexadecimal or octal value by preceding it with a leading ‘‘0x’’ or ‘‘0’’.

The **cmp** utility exits with one of the following values:

0 The files are identical.

1 The files are different; this includes the case where one file is identical to the first part of the other. In the latter case, if the **-s** option has not been specified, **cmp** writes to standard output that EOF was reached in the shorter file (before any differences were found).

>1 An error occurred.

SEE ALSO

diff(1), diff3(1)

STANDARDS

The **cmp** utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

co − check out RCS revisions

SYNOPSIS

co [*options*] *file* . . .

DESCRIPTION

co retrieves a revision from each RCS file and stores it into the corresponding working file.

Pathnames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

Revisions of an RCS file can be checked out locked or unlocked. Locking a revision prevents overlapping updates. A revision checked out for reading or processing (e.g., compiling) need not be locked. A revision checked out for editing and later checkin must normally be locked. Checkout with locking fails if the revision to be checked out is currently locked by another user. (A lock can be broken with **rcs**(1).) Checkout with locking also requires the caller to be on the access list of the RCS file, unless he is the owner of the file or the superuser, or the access list is empty. Checkout without locking is not subject to access list restrictions, and is not affected by the presence of locks.

A revision is selected by options for revision or branch number, checkin date/time, author, or state. When the selection options are applied in combination, **co** retrieves the latest revision that satisfies all of them. If none of the selection options is specified, **co** retrieves the latest revision on the default branch (normally the trunk, see the **−b** option of **rcs**(1)). A revision or branch number can be attached to any of the options **−f**, **−I**, **−l**, **−M**, **−p**, **−q**, **−r**, or **−u**. The options **−d** (date), **−s** (state), and **−w** (author) retrieve from a single branch, the *selected* branch, which is either specified by one of **−f**, . . ., **−u**, or the default branch.

A **co** command applied to an RCS file with no revisions creates a zero-length working file. **co** always performs keyword substitution (see below).

OPTIONS

−r[*rev*] retrieves the latest revision whose number is less than or equal to *rev*. If *rev* indicates a branch rather than a revision, the latest revision on that branch is retrieved. If *rev* is omitted, the latest revision on the default branch (see the **−b** option of **rcs**(1)) is retrieved. If *rev* is **$**, **co** determines the revision number from keyword values in the working file. Otherwise, a revision is composed of one or more numeric or symbolic fields separated by periods. If *rev* begins with a period, then the default branch (normally the trunk) is prepended to it. If *rev* is a branch number followed by a period, then the latest revision on that branch is used. The numeric equivalent of a symbolic field is specified with the **−n** option of the commands **ci**(1) and **rcs**(1).

−l[*rev*] same as **−r**, except that it also locks the retrieved revision for the caller.

−u[*rev*] same as **−r**, except that it unlocks the retrieved revision if it was locked by the caller. If *rev* is omitted, **−u** retrieves the revision locked by the caller, if there is one; otherwise, it retrieves the latest revision on the default branch.

−f[*rev*] forces the overwriting of the working file; useful in connection with **−q**. See also FILE MODES below.

−kkv Generate keyword strings using the default form, e.g. **$Revision: $** for the **Revision** keyword. A locker's name is inserted in the value of the **Header**, **Id**, and **Locker** keyword strings only as a file is being locked, i.e. by **ci −l** and **co −l**. This is the default.

−kkvl Like **−kkv**, except that a locker's name is always inserted if the given revision is currently locked.

−kk Generate only keyword names in keyword strings; omit their values. See KEYWORD SUBSTITUTION below. For example, for the **Revision** keyword, generate the string **$Revision$** instead of **$Revision: $**. This option is useful to ignore differences due to keyword substitution when comparing different revisions of a file. Log messages are inserted after **Log** keywords even if **−kk** is specified, since this tends to be more useful when merging changes.

−ko Generate the old keyword string, present in the working file just before it was checked in. For example, for the **Revision** keyword, generate the string **$Revision: 1.1 $** instead of **$Revision: $**

if that is how the string appeared when the file was checked in. This can be useful for file formats that cannot tolerate any changes to substrings that happen to take the form of keyword strings.

−kb Generate a binary image of the old keyword string. This acts like **−ko**, except it performs all working file input and output in binary mode. This makes little difference on POSIX and UNIX hosts, but on DOS-like hosts one should use **rcs −i −kb** to initialize an RCS file intended to be used for binary files. Also, on all hosts, **rcsmerge**(1) normally refuses to merge files when **−kb** is in effect.

−kv Generate only keyword values for keyword strings. For example, for the **Revision** keyword, generate the string **instead of $Revision: $**. This can help generate files in programming languages where it is hard to strip keyword delimiters like **$Revision: $** from a string. However, further keyword substitution cannot be performed once the keyword names are removed, so this option should be used with care. Because of this danger of losing keywords, this option cannot be combined with **−l**, and the owner write permission of the working file is turned off; to edit the file later, check it out again without **−kv**.

−p[*rev*] prints the retrieved revision on the standard output rather than storing it in the working file. This option is useful when **co** is part of a pipe.

−q[*rev*] quiet mode; diagnostics are not printed.

−I[*rev*] interactive mode; the user is prompted and questioned even if the standard input is not a terminal.

−d*date* retrieves the latest revision on the selected branch whose checkin date/time is less than or equal to *date*. The date and time can be given in free format. The time zone **LT** stands for local time; other common time zone names are understood. For example, the following *date*s are equivalent if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of Coordinated Universal Time (UTC):

8:00 pm lt	
4:00 AM, Jan. 12, 1990	default is UTC
1990-01-12 04:00:00+00	ISO 8601 (UTC)
1990-01-11 20:00:00−08	ISO 8601 (local time)
1990/01/12 04:00:00	traditional RCS format
Thu Jan 11 20:00:00 1990 LT	output of **ctime**(3) + **LT**
Thu Jan 11 20:00:00 PST 1990	output of **date**(1)
Fri Jan 12 04:00:00 GMT 1990	
Thu, 11 Jan 1990 20:00:00 −0800	Internet RFC 822
12-January-1990, 04:00 WET	

Most fields in the date and time can be defaulted. The default time zone is normally UTC, but this can be overridden by the **−z** option. The other defaults are determined in the order year, month, day, hour, minute, and second (most to least significant). At least one of these fields must be provided. For omitted fields that are of higher significance than the highest provided field, the time zone's current values are assumed. For all other omitted fields, the lowest possible values are assumed. For example, without **−z**, the date **20, 10:30** defaults to 10:30:00 UTC of the 20th of the UTC time zone's current month and year. The date/time must be quoted if it contains spaces.

−M[*rev*]
Set the modification time on the new working file to be the date of the retrieved revision. Use this option with care; it can confuse **make**(1).

−s*state* retrieves the latest revision on the selected branch whose state is set to *state*.

−T Preserve the modification time on the RCS file even if the RCS file changes because a lock is added or removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the change of lock would mean a change to keyword strings in the other working file.

−w[*login*]
retrieves the latest revision on the selected branch which was checked in by the user with login name *login*. If the argument *login* is omitted, the caller's login is assumed.

−j *joinlist*
generates a new revision which is the join of the revisions on *joinlist*. This option is largely obsoleted by **rcsmerge**(1) but is retained for backwards compatibility.

The *joinlist* is a comma-separated list of pairs of the form *rev2***:***rev3*, where *rev2* and *rev3* are (symbolic or numeric) revision numbers. For the initial such pair, *rev1* denotes the revision selected by the above options **−f**, . . ., **−w**. For all other pairs, *rev1* denotes the revision generated by the previous pair. (Thus, the output of one join becomes the input to the next.)

For each pair, **co** joins revisions *rev1* and *rev3* with respect to *rev2*. This means that all changes that transform *rev2* into *rev1* are applied to a copy of *rev3*. This is particularly useful if *rev1* and *rev3* are the ends of two branches that have *rev2* as a common ancestor. If *rev1*<*rev2*<*rev3* on the same branch, joining generates a new revision which is like *rev3*, but with all changes that lead from *rev1* to *rev2* undone. If changes from *rev2* to *rev1* overlap with changes from *rev2* to *rev3*, **co** reports overlaps as described in **merge**(1).

For the initial pair, *rev2* can be omitted. The default is the common ancestor. If any of the arguments indicate branches, the latest revisions on those branches are assumed. The options **−l** and **−u** lock or unlock *rev1*.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*, where *n* can be **3**, **4**, or **5**. This can be useful when interchanging RCS files with others who are running older versions of RCS. To see which version of RCS your correspondents are running, have them invoke **rcs −V**; this works with newer versions of RCS. If it doesn't work, have them invoke **rlog** on an RCS file; if none of the first few lines of output contain the string **branch:** it is version 3; if the dates' years have just two digits, it is version 4; otherwise, it is version 5. An RCS file generated while emulating version 3 loses its default branch. An RCS revision generated while emulating version 4 or earlier has a time stamp that is off by up to 13 hours. A revision extracted while emulating version 4 or earlier contains abbreviated dates of the form *yy*/*mm*/*dd* and can also contain different white space and line prefixes in the substitution for **Log**.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* specifies the date output format in keyword substitution, and specifies the default time zone for *date* in the **−d***date* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

The **−z** option does not affect dates stored in RCS files, which are always UTC.

KEYWORD SUBSTITUTION

Strings of the form **$***keyword***$** and **$***keyword***:**. . .**$** embedded in the text are replaced with strings of the form **$***keyword***:***value***$** where *keyword* and *value* are pairs listed below. Keywords can be embedded in literal strings or comments to identify a revision.

Initially, the user enters strings of the form **$***keyword***$**. On checkout, **co** replaces these strings with strings of the form **$***keyword***:***value***$**. If a revision containing strings of the latter form is checked back in, the value fields will be replaced during the next checkout. Thus, the keyword values are automatically updated

on checkout. This automatic substitution can be modified by the **−k** options.

Keywords and their corresponding values:

$Author$
The login name of the user who checked in the revision.

$Date$ The date and time the revision was checked in. With **−z***zone* a numeric time zone offset is appended; otherwise, the date is UTC.

$Header$
A standard header containing the full pathname of the RCS file, the revision number, the date and time, the author, the state, and the locker (if locked). With **−z***zone* a numeric time zone offset is appended to the date; otherwise, the date is UTC.

Id Same as **$Header$**, except that the RCS filename is without a path.

$Locker$
The login name of the user who locked the revision (empty if not locked).

Log The log message supplied during checkin, preceded by a header containing the RCS filename, the revision number, the author, and the date and time. With **−z***zone* a numeric time zone offset is appended; otherwise, the date is UTC. Existing log messages are *not* replaced. Instead, the new log message is inserted after **$Log:**...**$**. This is useful for accumulating a complete change log in a source file.

Each inserted line is prefixed by the string that prefixes the **Log** line. For example, if the **Log** line is "// **$Log: tan.cc $**", RCS prefixes each line of the log with "// ". This is useful for languages with comments that go to the end of the line. The convention for other languages is to use a " * " prefix inside a multiline comment. For example, the initial log comment of a C program conventionally is of the following form:

```
/*
 * $Log$
 */
```

For backwards compatibility with older versions of RCS, if the log prefix is /* or **(*** surrounded by optional white space, inserted log lines contain a space instead of / or **(**; however, this usage is obsolescent and should not be relied on.

$Name$
The symbolic name used to check out the revision, if any. For example, **co −rJoe** generates **$Name: Joe $**. Plain **co** generates just **$Name: $**.

$RCSfile$
The name of the RCS file without a path.

$Revision$
The revision number assigned to the revision.

$Source$
The full pathname of the RCS file.

$State$
The state assigned to the revision with the **−s** option of **rcs**(1) or **ci**(1).

The following characters in keyword values are represented by escape sequences to keep keyword strings well-formed.

char	*escape sequence*
tab	**\t**
newline	**\n**
space	**\040**
$	**\044**
\	\\

FILE MODES

The working file inherits the read and execute permissions from the RCS file. In addition, the owner write permission is turned on, unless **−kv** is set or the file is checked out unlocked and locking is set to strict (see **rcs**(1)).

If a file with the name of the working file exists already and has write permission, **co** aborts the checkout, asking beforehand if possible. If the existing working file is not writable or **−f** is given, the working file is deleted without asking.

FILES

co accesses files much as **ci**(1) does, except that it does not need to read the working file unless a revision number of **$** is specified.

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. See **ci**(1) for details.

DIAGNOSTICS

The RCS pathname, the working pathname, and the revision number retrieved are written to the diagnostic output. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

rcsintro(1), ci(1), ctime(3), date(1), ident(1), make(1), rcs(1), rcsclean(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

LIMITS

Links to the RCS and working files are not preserved.

There is no way to selectively suppress the expansion of keywords, except by writing them differently. In nroff and troff, this is done by embedding the null-character **\&** into the keyword.

NAME

col — filter reverse line feeds from input

SYNOPSIS

col [**-bfpx**] [**-l** *num*]

DESCRIPTION

col filters out reverse (and half reverse) line feeds so that the output is in the correct order with only forward and half forward line feeds, and replaces white-space characters with tabs where possible. This can be useful in processing the output of nroff(1) and tbl(1).

col reads from the standard input and writes to the standard output.

The options are as follows:

-b Do not output any backspaces, printing only the last character written to each column position.

-f Forward half line feeds are permitted (''fine'' mode). Normally characters printed on a half line boundary are printed on the following line.

-p Force unknown control sequences to be passed through unchanged. Normally, **col** will filter out any control sequences from the input other than those recognized and interpreted by itself, which are listed below.

-x Output multiple spaces instead of tabs.

-l *num* Buffer at least *num* lines in memory. By default, 128 lines are buffered.

The control sequences for carriage motion that **col** understands and their decimal values are listed in the following table:

ESC−7	reverse line feed (escape then 7)
ESC−8	half reverse line feed (escape then 8)
ESC−9	half forward line feed (escape then 9)
backspace	moves back one column (8); ignored in the first column
carriage return	(13)
newline	forward line feed (10); also does carriage return
shift in	shift to normal character set (15)
shift out	shift to alternative character set (14)
space	moves forward one column (32)
tab	moves forward to next tab stop (9)
vertical tab	reverse line feed (11)

All unrecognized control characters and escape sequences are discarded.

col keeps track of the character set as characters are read and makes sure the character set is correct when they are output.

If the input attempts to back up to the last flushed line, **col** will display a warning message.

SEE ALSO

expand(1), nroff(1), tbl(1)

STANDARDS

The **col** utility conforms to X/Open Portability Guide Issue 4, Version 2 ("XPG4.2"). The **-l** option is an extension to the standard.

HISTORY

A **col** command appeared in Version 6 AT&T UNIX.

NAME

colcrt — filter nroff output for CRT previewing

SYNOPSIS

colcrt [**-**] [**-2**] [*file . . .*]

DESCRIPTION

colcrt provides virtual half-line and reverse line feed sequences for terminals without such capability, and on which overstriking is destructive. Half-line characters and underlining (changed to dashing '−') are placed on new lines in between the normal output lines.

Available options:

- Suppress all underlining. This option is especially useful for previewing *allboxed* tables from tbl(1).

-2 Causes all half-lines to be printed, effectively double spacing the output. Normally, a minimal space output format is used which will suppress empty lines. The program never suppresses two consecutive empty lines, however. The **-2** option is useful for sending output to the line printer when the output contains superscripts and subscripts which would otherwise be invisible.

EXAMPLES

A typical use of **colcrt** would be

```
tbl exum2.n | nroff -ms | colcrt - | more
```

SEE ALSO

col(1), more(1), nroff(1), troff(1), ul(1)

HISTORY

The **colcrt** command appeared in 3.0BSD.

BUGS

Should fold underlines onto blanks even with the '**-**' option so that a true underline character would show.

Can't back up more than 102 lines.

General overstriking is lost; as a special case '|' overstruck with '−' or underline becomes '+'.

Lines are trimmed to 132 characters.

Some provision should be made for processing superscripts and subscripts in documents which are already double-spaced.

NAME

colrm — remove columns from a file

SYNOPSIS

colrm [*start* [*stop*]]

DESCRIPTION

colrm removes selected columns from the lines of a file. A column is defined as a single character in a line. Input is read from the standard input. Output is written to the standard output.

If only the *start* column is specified, columns numbered less than the *start* column will be written. If both *start* and *stop* columns are specified, columns numbered less than the *start* column or greater than the *stop* column will be written. Column numbering starts with one, not zero.

Tab characters increment the column count to the next multiple of eight. Backspace characters decrement the column count by one.

SEE ALSO

awk(1), column(1), cut(1), paste(1)

HISTORY

The **colrm** command appeared in 3.0 BSD.

NAME

column — columnate lists

SYNOPSIS

column [**-tx**] [**-c** *columns*] [**-s** *sep*] [*file . . .*]

DESCRIPTION

The **column** utility formats its input into multiple columns. Rows are filled before columns. Input is taken from *file* operands, or, by default, from the standard input. Empty lines are ignored.

The options are as follows:

-c Output is formatted for a display *columns* wide.

-s Specify a set of characters to be used to delimit columns for the **-t** option.

-t Determine the number of columns the input contains and create a table. Columns are delimited with whitespace, by default, or with the characters supplied using the **-s** option. Useful for pretty-printing displays.

-x Fill columns before filling rows.

column exits 0 on success, >0 if an error occurred.

ENVIRONMENT

COLUMNS The environment variable COLUMNS is used to determine the size of the screen if no other information is available.

EXAMPLES

```
(echo "PERM LINKS OWNER GROUP SIZE MONTH DAY HH:MM/YEAR NAME"; \
ls -l | sed 1d) | column -t
```

SEE ALSO

colrm(1), ls(1), paste(1), sort(1)

HISTORY

The **column** command appeared in 4.3BSD−Reno.

NAME

comm — select or reject lines common to two files

SYNOPSIS

comm [**-123f**] *file1 file2*

DESCRIPTION

The **comm** utility reads *file1* and *file2*, which should be sorted lexically, and produces three text columns as output: lines only in *file1*; lines only in *file2*; and lines in both files.

The filename ‘‘-’’ means the standard input.

The following options are available:

-1 Suppress printing of column 1.

-2 Suppress printing of column 2.

-3 Suppress printing of column 3.

-f Fold case in line comparisons.

Each column will have a number of tab characters prepended to it equal to the number of lower numbered columns that are being printed. For example, if column number two is being suppressed, lines printed in column number one will not have any tabs preceding them, and lines printed in column number three will have one.

comm assumes that the files are lexically sorted; all characters participate in line comparisons.

EXIT STATUS

comm exits 0 on success, >0 if an error occurred.

SEE ALSO

cmp(1), diff(1), sort(1), uniq(1)

STANDARDS

The **comm** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

NAME

compile_et — error table compiler

SYNOPSIS

compile_et [*--version*] [*--help*] *file*

DESCRIPTION

The **compile_et** utility reads the table describing error-code names and their associated messages in the file *file* and generates a C source file suitable for use with the com_err(3) library. The source file *file* must end with a suffix of ".et" and **compile_et** writes a C header file file.h which contains definitions of the numerical values of the error codes defined in the error table and a C source file file.c which should be compiled and linked with the executable.

The source file is a plain ASCII text file. A "#" in the source file is treated as a comment character, and all remaining text to the end of the source line will be ignored. The source file consists of the following declarations:

id [*base*] *string*
: Defines an identification string (such a a version string) which is recorded in the generated files. It is mandatory and must be the first declaration in the source file.

et *name*
: Specifies the name of the error table to be *name*. It is mandatory and must be declared after the id declaration and before all other declarations. The name of table is limited to four ASCII characters. The optional argument *base* specifies the base value of error codes the table.

  The name of the table is used to construct the name of a function **initialize_<name>_error_table**() which must be called to register the error table the the com_err(3) library. A re-entrant (thread-safe) version called **initialize_<name>_error_table_r**() is also defined.

prefix [*string*]
: Defines a prefix to be applied to all error code names. If no string is specified, the prefix is not defined. It is an optional declaration and can appear more than once.

index *val*
: Specifies the index *val* in the error table for the following error code declaration. Subsequent error codes are allocated sequentially from the same value. It is an optional declaration and can appear more than once.

ec *cname*, *msg*
: Defines an error code with the name *cname* and its associated error message *msg*. The error codes are assigned sequentially increasing numbers. The name is placed into the C header file as an enumerated type.

end
: Indicates the end of the error table.

To maintain compatibility, new codes should be added to the end of an existing table, and codes should not be removed from tables.

EXAMPLES

A short sample error table might be test_err.et:

```
# example error table source file
id      "\$Id\$"
et      test
prefix  TEST
```

```
ec      PERM,           "Operation not permitted"
ec      IO,             "I/O error"
ec      NOMEM,          "Out of memory"
ec      INVAL,          "Invalid argument"
end
```

Compiling the source file `test_err.et` with **`compile_et`** will create a C header file `test_err.h` containing the enumerated type *test_error_number* with values TEST_PERM, TEST_IO, TEST_NOMEM and TEST_INVAL, and a C source file `test_err.c` containing the `com_err`(3) initialisation function **`initialize_test_error_table`**().

SEE ALSO

`com_err`(3)

NAME

compress, **uncompress** — compress and expand data

SYNOPSIS

compress [**-cdfv**] [**-b** *bits*] [*file ...*]
uncompress [**-cdfv**] [*file ...*]

DESCRIPTION

compress reduces the size of the named files using adaptive Lempel-Ziv coding. Each *file* is renamed to the same name plus the extension ".Z". As many of the modification time, access time, file flags, file mode, user ID, and group ID as allowed by permissions are retained in the new file. If compression would not reduce the size of a *file*, the file is ignored.

uncompress restores the compressed files to their original form, renaming the files by deleting the ".Z" extension.

If renaming the files would cause files to be overwritten and the standard input device is a terminal, the user is prompted (on the standard error output) for confirmation. If prompting is not possible or confirmation is not received, the files are not overwritten.

If no files are specified, the standard input is compressed or uncompressed to the standard output. If either the input and output files are not regular files, the checks for reduction in size and file overwriting are not performed, the input file is not removed, and the attributes of the input file are not retained.

The options are as follows:

-b Specify the *bits* code limit (see below).

-c Compressed or uncompressed output is written to the standard output. No files are modified.

-d Force decompression.

-f Force compression of *file*, even if it is not actually reduced in size. Additionally, files are overwritten without prompting for confirmation.

-v Print the percentage reduction of each file.

compress uses a modified Lempel-Ziv algorithm. Common substrings in the file are first replaced by 9-bit codes 257 and up. When code 512 is reached, the algorithm switches to 10-bit codes and continues to use more bits until the limit specified by the **-b** flag is reached (the default is 16). *Bits* must be between 9 and 16.

After the *bits* limit is reached, **compress** periodically checks the compression ratio. If it is increasing, **compress** continues to use the existing code dictionary. However, if the compression ratio decreases, **compress** discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next "block" of the file.

The **-b** flag is omitted for *uncompress* since the *bits* parameter specified during compression is encoded within the output, along with a magic number to ensure that neither decompression of random data nor recompression of compressed data is attempted.

The amount of compression obtained depends on the size of the input, the number of *bits* per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50−60%. Compression is generally much better than that achieved by Huffman coding (as used in the historical command pack), or adaptive Huffman coding (as used in the historical command compact), and takes less time to compute.

The **compress** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

zcat(1)

Welch, Terry A., "A Technique for High Performance Data Compression", *IEEE Computer*, 17:6, pp. 8-19, June, 1984.

HISTORY

The **compress** command appeared in 4.3BSD.

NAME

config — build kernel compilation directories

SYNOPSIS

config [**-Ppv**] [**-b** *builddir*] [**-s** *srcdir*] [*config-file*]
config -x [*kernel-file*]
config -L [**-v**] [**-s** *srcdir*] [*config-file*]

DESCRIPTION

In its first synopsis form, **config** creates a kernel build directory from the machine description file *config-file*, which describes the system to configure. Refer to section KERNEL BUILD CONFIGURATION for the details of that use of **config**.

In its second synopsis form, **config** takes the binary kernel *kernel-file* as its single argument (aside from the mandatory **-x** flag), then extracts the embedded configuration file (if any) and writes it to standard output. If *kernel-file* is not given, `/netbsd` is used. Configuration data will be available if the given kernel was compiled with either *INCLUDE_CONFIG_FILE* or *INCLUDE_JUST_CONFIG* options.

In its third synopsis form, **config** is a tool for the kernel developer and generates a "lint" configuration file to be used during regression testing. Refer to section LINT CONFIGURATION for the details of that use of **config**.

config accepts the following parameters:

-b *builddir*
: Use *builddir* as the kernel build directory, instead of computing and creating one automatically.

-L
: Generate a lint configuration.

-P
: Pack locators to save space in the resulting kernel binary. The amount of space saved that way is so small that this option should be considered historical, and of no actual use.

-p
: Generate a build directory suited for kernel profiling. However, this options should be avoided in favor of the relevant options inside the configuration file as described in section KERNEL BUILD CONFIGURATION.

-s *srcdir*
: Point to the top of the kernel source tree. It must be an absolute path when **config** is used to prepare a kernel build directory, but can be relative when it is used in combination with the **-L** flag.

-v
: Increase verbosity by enabling some more warnings.

-x
: Extract the configuration embedded in a kernel binary.

KERNEL BUILD CONFIGURATION

There are several different ways to run the **config** program. The traditional way is to run **config** from the `conf` subdirectory of the machine-specific directory of the system source (usually `/sys/arch/MACHINE/conf`, where `MACHINE` is one of `vax`, `hp300`, and so forth), and to specify as the *config-file* the name of a machine description file located in that directory. **config** will by default create files in the directory `../compile/SYSTEMNAME`, where `SYSTEMNAME` is the last path component of *config-file*. **config** will assume that the top-level kernel source directory is located four directories above the build directory.

Another way is to create the build directory yourself, place the machine description file in the build directory with the name `CONFIG`, and run **config** from within the build directory without specifying a *config-file*. **config** will then by default create files in the current directory. If you run **config** this way, you must specify the location of the top-level kernel source directory using the **-s** option or by using

the "`source`" directive at the beginning of the machine description file.

Finally, you can specify the build directory for **`config`** and run it from anywhere. You can specify a build directory with the **`-b`** option or by using the "`build`" directive at the beginning of the machine description file. You must specify the location of the top-level kernel source directory if you specify a build directory.

If *`config-file`* is a binary kernel, **`config`** will try to extract the configuration file embedded into it, which will be present if that kernel was built either with *INCLUDE_CONFIG_FILE* or *INCLUDE_JUST_CONFIG* options. This work mode requires you to manually specify a build directory with the **`-b`** option, which implies the need to provide a source tree too.

If the **`-p`** option is supplied, `.PROF` is appended to the default compilation directory name, and **`config`** acts as if the lines "`makeoptions PROF="-pg"`" and "`options GPROF`" appeared in the machine description file. This will build a system that includes profiling code; see `kgmon`(8) and `gprof`(1). The **`-p`** flag is expected to be used for "one-shot" profiles of existing systems; for regular profiling, it is probably wiser to create a separate machine description file containing the `makeoptions` line.

The old undocumented **`-g`** flag is no longer supported. Instead, use "`makeoptions DEBUG="-g"`" and (typically) "`options KGDB`".

The output of **`config`** consists of a number of files, principally `ioconf.c`, a description of I/O devices that may be attached to the system; and a `Makefile`, used by `make`(1) in building the kernel.

After running **`config`**, it is wise to run "`make depend`" in the directory where the new makefile was created. **`config`** prints a reminder of this when it completes.

If **`config`** stops due to errors, the problems reported should be corrected and **`config`** should be run again. **`config`** attempts to avoid changing the compilation directory if there are configuration errors, but this code is not well-tested, and some problems (such as running out of disk space) are unrecoverable.

LINT CONFIGURATION

A so-called "lint" configuration should include everything from the kernel that can possibly be selected. The rationale is to provide a way to reach all the code a user might select, in order to make sure all options and drivers compile without error for a given source tree.

When used with the **`-L`** flag, **`config`** takes the regular configuration file *`config-file`* and prints on the standard output a configuration file that includes *`config-file`*, selects all options and file-systems the user can possibly select, and defines an instance of every possible attachment as described by the kernel option definition files used by *`config-file`*.

The resulting configuration file is meant as a way to select all possible features in order to test that each of them compiles. It is not meant to result in a kernel binary that can run on any hardware.

Unlike the first synopsis form, the provided *`srcdir`* is relative to the current working directory. In the first synopsis form, it is relative to the build directory.

SEE ALSO

The SYNOPSIS portion of each device in section 4.

`options`(4), `config`(5), `config`(9)

HISTORY

The **`config`** command appeared in 4.1BSD. It was completely revised in 4.4BSD. The **`-x`** option appeared in NetBSD 2.0. The **`-L`** option appeared in NetBSD 5.0.

NAME

config.guess − guess the build system triplet

SYNOPSIS

config.guess [*OPTION*]

DESCRIPTION

The GNU build system distinguishes three types of machines, the ‘build’ machine on which the compilers are run, the ‘host’ machine on which the package being built will run, and, exclusively when you build a compiler, assembler etc., the ‘target’ machine, for which the compiler being built will produce code.

This script will guess the type of the ‘build’ machine.

Output the configuration name of the system ‘config.guess’ is run on.

Operation modes:

−h, **−−help**
print this help, then exit

−t, **−−time−stamp**
print date of last modification, then exit

−v, **−−version**
print version number, then exit

ENVIRONMENT VARIABLES

config.guess might need to compile and run C code, hence it needs a compiler for the ‘build’ machine: use the environment variable ‘CC_FOR_BUILD’ to specify the compiler for the build machine. If ‘CC_FOR_BUILD’ is not specified, ‘CC’ will be used. Be sure to specify ‘CC_FOR_BUILD’ is ‘CC’ is a cross-compiler to the ‘host’ machine.

CC_FOR_BUILD	a native C compiler, defaults to ‘cc’
CC	a native C compiler, the previous variable is preferred

REPORTING BUGS

Report bugs and patches to <config-patches@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

NAME

config.sub − validate and canonicalize a configuration triplet

SYNOPSIS

config.sub [*OPTION*] *CPU-MFR-OPSYS*
config.sub [*OPTION*] *ALIAS*

DESCRIPTION

Canonicalize a configuration name.

Operation modes:

−h, **−−help**
print this help, then exit

−t, **−−time−stamp**
print date of last modification, then exit

−v, **−−version**
print version number, then exit

REPORTING BUGS

Report bugs and patches to <config-patches@gnu.org>.

COPYRIGHT

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

NAME

copy_cred_cache — copy credentials from one cache to another

SYNOPSIS

copy_cred_cache [**--krbtgt-only**] [**--service=***principal*] [**--enctype=***enctype*]
[**--flags=***ticketflags*] [**--valid-for=***time*]
[**--fcache-version=***integer*] [⟨*from-cache*⟩] ⟨*to-cache*⟩

DESCRIPTION

copy_cred_cache copies credentials from ⟨*from-cache*⟩ (or the default cache) to ⟨*to-cache*⟩.

Supported options:

--krbtgt-only
: Copies only krbtgt credentials for the client's realm. This is equivalent to **--service=**krbtgt/⟨*CLIENTREALM*⟩@⟨*CLIENTREALM*⟩.

--service=*principal*
: Copies only credentials matching this service principal.

--enctype=*enctype*
: Copies only credentials a matching enctype.

--flags=*ticketflags*
: Copies only credentials with these ticket flags set.

--valid-for=*time*
: Copies only credentials that are valid for at least this long. This does not take renewable creds into account.

--fcache-version=*integer*
: The created cache, If a standard FILE cache is created, it will have this file format version.

EXAMPLES

To copy only credentials that are valid for at least one day and with the initial flag set, try something like:

```
$ copy_cred_cache --valid-for=1d --flags=initial FILE:/some/cache
```

DIAGNOSTICS

The **copy_cred_cache** utility exits 0 on success, and >0 if an error occurs, or of no credentials where actually copied.

NAME

cp — copy files

SYNOPSIS

cp [**-R** [**-H** | **-L** | **-P**]] [**-f** | **-i**] [**-Npv**] *source_file target_file*
cp [**-R** [**-H** | **-L** | **-P**]] [**-f** | **-i**] [**-Npv**] *source_file ... target_directory*

DESCRIPTION

In the first synopsis form, the **cp** utility copies the contents of the *source_file* to the *target_file*. In the second synopsis form, the contents of each named *source_file* is copied to the destination *target_directory*. The names of the files themselves are not changed. If **cp** detects an attempt to copy a file to itself, the copy will fail.

The following options are available:

-f For each existing destination pathname, attempt to overwrite it. If permissions do not allow copy to succeed, remove it and create a new file, without prompting for confirmation. (The **-i** option is ignored if the **-f** option is specified.)

-H If the **-R** option is specified, symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.)

-i Causes **cp** to write a prompt to the standard error output before copying a file that would overwrite an existing file. If the response from the standard input begins with the character 'y', the file copy is attempted.

-L If the **-R** option is specified, all symbolic links are followed.

-N When used with **-p**, don't copy file flags.

-P No symbolic links are followed.

-p Causes **cp** to preserve in the copy as many of the modification time, access time, file flags, file mode, user ID, and group ID as allowed by permissions.

If the user ID and group ID cannot be preserved, no error message is displayed and the exit value is not altered.

If the source file has its set user ID bit on and the user ID cannot be preserved, the set user ID bit is not preserved in the copy's permissions. If the source file has its set group ID bit on and the group ID cannot be preserved, the set group ID bit is not preserved in the copy's permissions. If the source file has both its set user ID and set group ID bits on, and either the user ID or group ID cannot be preserved, neither the set user ID or set group ID bits are preserved in the copy's permissions.

-R If *source_file* designates a directory, **cp** copies the directory and the entire subtree connected at that point. This option also causes symbolic links to be copied, rather than indirected through, and for **cp** to create special files rather than copying them as normal files. Created directories have the same mode as the corresponding source directory, unmodified by the process's umask.

-v Cause **cp** to be verbose, showing files as they are copied.

For each destination file that already exists, its contents are overwritten if permissions allow, but its mode, user ID, and group ID are unchanged.

In the second synopsis form, *target_directory* must exist unless there is only one named *source_file* which is a directory and the **-R** flag is specified.

If the destination file does not exist, the mode of the source file is used as modified by the file mode creation mask (**umask**, see csh(1)). If the source file has its set user ID bit on, that bit is removed unless both the

source file and the destination file are owned by the same user. If the source file has its set group ID bit on, that bit is removed unless both the source file and the destination file are in the same group and the user is a member of that group. If both the set user ID and set group ID bits are set, all of the above conditions must be fulfilled or both bits are removed.

Appropriate permissions are required for file creation or overwriting.

Symbolic links are always followed unless the **-R** flag is set, in which case symbolic links are not followed, by default. The **-H** or **-L** flags (in conjunction with the **-R** flag), as well as the **-P** flag cause symbolic links to be followed as described above. The **-H** and **-L** options are ignored unless the **-R** option is specified. In addition, these options override each other and the command's actions are determined by the last one specified.

EXIT STATUS

The **cp** utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

Historic versions of the **cp** utility had a **-r** option. This implementation supports that option, however, its use is strongly discouraged, as it does not correctly copy special files, symbolic links or fifo's.

SEE ALSO

mv(1), rcp(1), umask(2), fts(3), symlink(7)

STANDARDS

The **cp** utility is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

The **-v** option is an extension to IEEE Std 1003.2 ("POSIX.2").

NAME

cpio — copy file archives in and out

SYNOPSIS

cpio **-o** [**-AaBcLvZz**] [**-C** *bytes*] [**-F** *archive*] [**-H** *format*] [**-O** *archive*] < *name-list* [> *archive*]
cpio **-i** [**-6BbcdfmrSstuvZz**] [**-C** *bytes*] [**-E** *file*] [**-F** *archive*] [**-H** *format*] [**-I** *archive*] [*pattern ...*] [< *archive*]
cpio **-p** [**-adLlmuv**] *destination-directory* < *name-list*

DESCRIPTION

The **cpio** command copies files to and from a **cpio** archive. If the archive is of the form: *[[user@]host:]file* then the archive will be processed using `rmt`(8).

The options are as follows:

-o, **--create**
Create an archive. Reads the list of files to store in the archive from standard input, and writes the archive on standard output.

-a, **--reset-access-time**
Reset the access times on files that have been copied to the archive.

-A, **--append**
Append to the specified archive.

-B Set block size of output to 5120 bytes.

-c Use ASCII format for **cpio** header for portability.

-C *bytes*
Set the block size of output to *bytes*.

-F *archive*

-O *archive*
Use the specified file name as the archive to write to.

-H *format*
Write the archive in the specified format. Recognized formats are:

bcpio	Old binary **cpio** format.
cpio	Old octal character **cpio** format.
sv4cpio	SVR4 hex **cpio** format.
tar	Old tar format.
ustar	POSIX ustar format.

-L Follow symbolic links.

-v Be verbose about operations. List filenames as they are written to the archive.

-z Compress archive using `gzip`(1) format.

-Z Compress archive using `compress`(1) format.

-i, **--extract**
Restore files from an archive. Reads the archive file from standard input and extracts files matching the *patterns* that were specified on the command line.

-b Do byte and word swapping after reading in data from the archive, for restoring archives created on systems with a different byte order.

-B Set the block size of the archive being read to 5120 bytes.

-c Expect the archive headers to be in ASCII format.

-C *bytes*
Read archive written with a block size of *bytes*.

-d, **--make-directories**
Create any intermediate directories as needed during restore.

-E *file*, **--pattern-file** *file*
Read list of file name patterns to extract or list from *file*.

-f, **--nonmatching**
Restore all files except those matching the *patterns* given on the command line.

-F *archive*, **--file** *archive*

-I *archive*
Use the specified file as the input for the archive.

-H *format*, **--format** *format*
Read an archive of the specified format. Recognized formats are:

bcpio	Old binary **cpio** format.
cpio	Old octal character **cpio** format.
sv4cpio	SVR4 hex **cpio** format.
tar	Old tar format.
ustar	POSIX ustar format.

-m Restore modification times on files.

-r, **--rename**
Rename restored files interactively.

-s Swap bytes after reading data from the archive.

-S, **--swap-halfwords**
Swap words after reading data from the archive.

-t, **--list**
Only list the contents of the archive, no files or directories will be created.

-u, **--unconditional**
Overwrite files even when the file in the archive is older than the one that will be overwritten.

-v, **--verbose**
Be verbose about operations. List filenames as they are copied in from the archive.

-z Uncompress archive using `gzip`(1) format.

-Z Uncompress archive using `compress`(1) format.

-6 Process old-style **cpio** format archives.

-p, **--pass-through**
Copy files from one location to another in a single pass. The list of files to copy are read from standard input and written out to a directory relative to the specified *directory* argument.

-a Reset the access times on files that have been copied.

-d Create any intermediate directories as needed to write the files at the new location.

-l, **--link**
When possible, link files rather than creating an extra copy.

-L, **--dereference**
Follow symbolic links.

-m, **--preserve-modification-time**
Restore modification times on files.

-u, **--unconditional**
Overwrite files even when the original file being copied is older than the one that will be overwritten.

-v, **--verbose**
Be verbose about operations. List filenames as they are copied.

--force-local
Do not interpret filenames that contain a ':' as remote files.

--insecure
Normally **cpio** ignores filenames that contain ".." as a path component. With this option, files that contain ".." can be processed.

EXIT STATUS

cpio will exit with one of the following values:

0 All files were processed successfully.

1 An error occurred.

Whenever **cpio** cannot create a file or a link when extracting an archive or cannot find a file while writing an archive, or cannot preserve the user ID, group ID, file mode, or access and modification times when the **-p** option is specified, a diagnostic message is written to standard error and a non-zero exit value will be returned, but processing will continue. In the case where **cpio** cannot create a link to a file, **cpio** will not create a second copy of the file.

If the extraction of a file from an archive is prematurely terminated by a signal or error, **cpio** may have only partially extracted the file the user wanted. Additionally, the file modes of extracted files and directories may have incorrect file bits, and the modification and access times may be wrong.

If the creation of an archive is prematurely terminated by a signal or error, **cpio** may have only partially created the archive which may violate the specific archive format specification.

SEE ALSO

pax(1), tar(1)

AUTHORS

Keith Muller at the University of California, San Diego.

BUGS

The **-s** and **-S** options are currently not implemented.

NAME

cpp — C preprocessor

SYNOPSIS

cpp [**-CdMtVv**] [**-D** *macro[=value]*] [**-I** *path*] [**-i** *file*] [**-S** *path*] [**-U** *macro*] [*infile* | -] [*outfile*]

DESCRIPTION

The **cpp** utility is a macro preprocessor used by the pcc(1) compiler. It is used to include header files, expand macro definitions, and perform conditional compilation.

The *infile* input file is optional. If not provided or the file name is "-" (dash), **cpp** reads its initial file from standard input. The *outfile* output file is also optional. It writes by default to standard output.

The options are as follows:

-C Do not discard comments.

-D *macro[=value]*
Fake a definition at the beginning by using "#define *macro=value*". If *value* is not set on command-line, then defines as 1.

-dM Print list of "#define" statements to standard output for all defined macros other than builtin macros (see below). The normal results of preprocessing are not outputted.

-I *path*
Add *path* to the list of directories containing needed header files. This may be used to override system include directories (see **-S** option). **-I** may be specified multiple times.

-i *file*
Include a file at the beginning by using "#include *file*".

-M Generate dependencies for make(1).

-S *path*
Add *path* to the list of system directories containing needed header files. **-S** may be specified multiple times. Note: **cpp** does not have a default include directory defined.

-t Traditional cpp syntax. Do not define the __TIME__, __DATE__, __STDC__, and __STDC_VERSION__, macros.

-U *macro*
Undefine a macro at the beginning by using "#undef *macro*".

-v Display version.

-V Verbose debugging output. **-V** can be repeated for further details.

-? Show command line usage for **cpp**.

Builtin Macros

A few macros are interpreted inside the **cpp** program:

__DATE__ Expands to the date in abbreviated month, day, and year format from ctime(3) in quotes.

__FILE__ Expands to the name of the current input file in quotes. When read from standard input, it expands to "⟨stdin⟩".

__LINE__ Expands to the line number of the current line containing the macro.

__STDC__ Expands to the constant 1. This means the compiler conforms to ISO Standard C, ISO/IEC 9899:1990 (also known as "C90").

__STDC_VERSION__ Expands to "199901L" which indicates it supports ISO/IEC 9899:1999 standard (commonly referred to as "C99").

__TIME__ Expands to the time in hour, minutes, and seconds from `ctime`(3) in quotes.

Also see the **`-t`** option.

EXIT STATUS

The **`cpp`** utility exits with one of the following values:

0 Successfully finished.

1 An error occurred.

SEE ALSO

`as`(1), `ccom`(1), `pcc`(1)

HISTORY

The **`cpp`** command comes from the original Portable C Compiler by S. C. Johnson, written in the late 70's. The code originates from the V6 preprocessor with some additions from V7 cpp and ansi/c99 support.

A lot of the PCC code was rewritten by Anders Magnusson.

This product includes software developed or owned by Caldera International, Inc.

NAME

cpp − The C Preprocessor

SYNOPSIS

cpp [**−D***macro*[=*defn*]...] [**−U***macro*]
[**−I***dir*...] [**−iquote***dir*...]
[**−W***warn*...]
[**−M**|**−MM**] [**−MG**] [**−MF** *filename*]
[**−MP**] [**−MQ** *target*...]
[**−MT** *target*...]
[**−P**] [**−fno−working−directory**]
[**−x** *language*] [**−std=***standard*]
infile outfile

Only the most useful options are listed here; see below for the remainder.

DESCRIPTION

The C preprocessor, often known as *cpp*, is a *macro processor* that is used automatically by the C compiler to transform your program before compilation. It is called a macro processor because it allows you to define *macros*, which are brief abbreviations for longer constructs.

The C preprocessor is intended to be used only with C, C++, and Objective-C source code. In the past, it has been abused as a general text processor. It will choke on input which does not obey C's lexical rules. For example, apostrophes will be interpreted as the beginning of character constants, and cause errors. Also, you cannot rely on it preserving characteristics of the input which are not significant to C−family languages. If a Makefile is preprocessed, all the hard tabs will be removed, and the Makefile will not work.

Having said that, you can often get away with using cpp on things which are not C. Other Algol-ish programming languages are often safe (Pascal, Ada, etc.) So is assembly, with caution. **−traditional−cpp** mode preserves more white space, and is otherwise more permissive. Many of the problems can be avoided by writing C or C++ style comments instead of native language comments, and keeping macros simple.

Wherever possible, you should use a preprocessor geared to the language you are writing in. Modern versions of the GNU assembler have macro facilities. Most high level programming languages have their own conditional compilation and inclusion mechanism. If all else fails, try a true general text processor, such as GNU M4.

C preprocessors vary in some details. This manual discusses the GNU C preprocessor, which provides a small superset of the features of ISO Standard C. In its default mode, the GNU C preprocessor does not do a few things required by the standard. These are features which are rarely, if ever, used, and may cause surprising changes to the meaning of a program which does not expect them. To get strict ISO Standard C, you should use the **−std=c89** or **−std=c99** options, depending on which version of the standard you want. To get all the mandatory diagnostics, you must also use **−pedantic**.

This manual describes the behavior of the ISO preprocessor. To minimize gratuitous differences, where the ISO preprocessor's behavior does not conflict with traditional semantics, the traditional preprocessor should behave the same way. The various differences that do exist are detailed in the section **Traditional Mode**.

For clarity, unless noted otherwise, references to **CPP** in this manual refer to GNU CPP.

OPTIONS

The C preprocessor expects two file names as arguments, *infile* and *outfile*. The preprocessor reads *infile* together with any other files it specifies with **#include**. All the output generated by the combined input files is written in *outfile*.

Either *infile* or *outfile* may be **−**, which as *infile* means to read from standard input and as *outfile* means to write to standard output. Also, if either file is omitted, it means the same as if **−** had been specified for that file.

Unless otherwise noted, or the option ends in **=**, all options which take an argument may have that argument appear either immediately after the option, or with a space between option and argument: **−Ifoo** and **−I foo** have the same effect.

Many options have multi-letter names; therefore multiple single-letter options may *not* be grouped: **−dM** is very different from **−d −M**.

−D *name*
Predefine *name* as a macro, with definition `1`.

−D *name*=*definition*
The contents of *definition* are tokenized and processed as if they appeared during translation phase three in a **#define** directive. In particular, the definition will be truncated by embedded newline characters.

If you are invoking the preprocessor from a shell or shell-like program you may need to use the shell's quoting syntax to protect characters such as spaces that have a meaning in the shell syntax.

If you wish to define a function-like macro on the command line, write its argument list with surrounding parentheses before the equals sign (if any). Parentheses are meaningful to most shells, so you will need to quote the option. With **sh** and **csh**, **−D'***name***(***args...***)=***definition***'** works.

−D and **−U** options are processed in the order they are given on the command line. All **−imacros** *file* and **−include** *file* options are processed after all **−D** and **−U** options.

−U *name*
Cancel any previous definition of *name*, either built in or provided with a **−D** option.

−undef
Do not predefine any system-specific or GCC-specific macros. The standard predefined macros remain defined.

−I *dir*
Add the directory *dir* to the list of directories to be searched for header files.

Directories named by **−I** are searched before the standard system include directories. If the directory *dir* is a standard system include directory, the option is ignored to ensure that the default search order for system directories and the special treatment of system headers are not defeated .

−o *file*
Write output to *file*. This is the same as specifying *file* as the second non-option argument to **cpp**. **gcc** has a different interpretation of a second non-option argument, so you must use **−o** to specify the output file.

−Wall
Turns on all optional warnings which are desirable for normal code. At present this is **−Wcomment**, **−Wtrigraphs**, **−Wmultichar** and a warning about integer promotion causing a change of sign in `#if` expressions. Note that many of the preprocessor's warnings are on by default and have no options to control them.

−Wcomment
−Wcomments
Warn whenever a comment-start sequence **/*** appears in a **/*** comment, or whenever a backslash-newline appears in a // comment. (Both forms have the same effect.)

−Wtrigraphs
@anchor{Wtrigraphs} Most trigraphs in comments cannot affect the meaning of the program. However, a trigraph that would form an escaped newline (**??/** at the end of a line) can, by changing where the comment begins or ends. Therefore, only trigraphs that would form escaped newlines produce warnings inside a comment.

This option is implied by **−Wall**. If **−Wall** is not given, this option is still enabled unless trigraphs are enabled. To get trigraph conversion without warnings, but get the other **−Wall** warnings, use **−trigraphs −Wall −Wno−trigraphs**.

−Wtraditional
Warn about certain constructs that behave differently in traditional and ISO C. Also warn about ISO C constructs that have no traditional C equivalent, and problematic constructs which should be avoided.

−Wimport
Warn the first time **#import** is used.

−Wundef
Warn whenever an identifier which is not a macro is encountered in an **#if** directive, outside of **defined**. Such identifiers are replaced with zero.

−Wunused−macros
Warn about macros defined in the main file that are unused. A macro is *used* if it is expanded or tested for existence at least once. The preprocessor will also warn if the macro has not been used at the time it is redefined or undefined.

Built-in macros, macros defined on the command line, and macros defined in include files are not warned about.

Note: If a macro is actually used, but only used in skipped conditional blocks, then CPP will report it as unused. To avoid the warning in such a case, you might improve the scope of the macro's definition by, for example, moving it into the first skipped block. Alternatively, you could provide a dummy use with something like:

```
#if defined the_macro_causing_the_warning
#endif
```

−Wendif−labels
Warn whenever an **#else** or an **#endif** are followed by text. This usually happens in code of the form

```
#if FOO
...
#else FOO
...
#endif FOO
```

The second and third `FOO` should be in comments, but often are not in older programs. This warning is on by default.

−Werror
Make all warnings into hard errors. Source code which triggers warnings will be rejected.

−Wsystem−headers
Issue warnings for code in system headers. These are normally unhelpful in finding bugs in your own code, therefore suppressed. If you are responsible for the system library, you may want to see them.

−w Suppress all warnings, including those which GNU CPP issues by default.

−pedantic
Issue all the mandatory diagnostics listed in the C standard. Some of them are left out by default, since they trigger frequently on harmless code.

−pedantic−errors
Issue all the mandatory diagnostics, and make all mandatory diagnostics into errors. This includes mandatory diagnostics that GCC issues without **−pedantic** but treats as warnings.

−M
Instead of outputting the result of preprocessing, output a rule suitable for **make** describing the dependencies of the main source file. The preprocessor outputs one **make** rule containing the object file name for that source file, a colon, and the names of all the included files, including those coming from **−include** or **−imacros** command line options.

Unless specified explicitly (with **−MT** or **−MQ**), the object file name consists of the basename of the

source file with any suffix replaced with object file suffix. If there are many included files then the rule is split into several lines using \−newline. The rule has no commands.

This option does not suppress the preprocessor's debug output, such as **−dM**. To avoid mixing such debug output with the dependency rules you should explicitly specify the dependency output file with **−MF**, or use an environment variable like **DEPENDENCIES_OUTPUT**. Debug output will still be sent to the regular output stream as normal.

Passing **−M** to the driver implies **−E**, and suppresses warnings with an implicit **−w**.

−MM
Like **−M** but do not mention header files that are found in system header directories, nor header files that are included, directly or indirectly, from such a header.

This implies that the choice of angle brackets or double quotes in an **#include** directive does not in itself determine whether that header will appear in **−MM** dependency output. This is a slight change in semantics from GCC versions 3.0 and earlier.

@anchor{dashMF}

−MF *file*
When used with **−M** or **−MM**, specifies a file to write the dependencies to. If no **−MF** switch is given the preprocessor sends the rules to the same place it would have sent preprocessed output.

When used with the driver options **−MD** or **−MMD**, **−MF** overrides the default dependency output file.

−MG
In conjunction with an option such as **−M** requesting dependency generation, **−MG** assumes missing header files are generated files and adds them to the dependency list without raising an error. The dependency filename is taken directly from the `#include` directive without prepending any path. **−MG** also suppresses preprocessed output, as a missing header file renders this useless.

This feature is used in automatic updating of makefiles.

−MP
This option instructs CPP to add a phony target for each dependency other than the main file, causing each to depend on nothing. These dummy rules work around errors **make** gives if you remove header files without updating the *Makefile* to match.

This is typical output:

```
test.o: test.c test.h

test.h:
```

−MT *target*
Change the target of the rule emitted by dependency generation. By default CPP takes the name of the main input file, including any path, deletes any file suffix such as **.c**, and appends the platform's usual object suffix. The result is the target.

An **−MT** option will set the target to be exactly the string you specify. If you want multiple targets, you can specify them as a single argument to **−MT**, or use multiple **−MT** options.

For example, **−MT '$(objpfx)foo.o'** might give

```
$(objpfx)foo.o: foo.c
```

−MQ *target*
Same as **−MT**, but it quotes any characters which are special to Make. **−MQ '$(objpfx)foo.o'** gives

```
$$(objpfx)foo.o: foo.c
```

The default target is automatically quoted, as if it were given with **−MQ**.

−MD

−MD is equivalent to **−M −MF** *file*, except that **−E** is not implied. The driver determines *file* based on whether an **−o** option is given. If it is, the driver uses its argument but with a suffix of *.d*, otherwise it take the basename of the input file and applies a *.d* suffix.

If **−MD** is used in conjunction with **−E**, any **−o** switch is understood to specify the dependency output file (but @pxref{dashMF,,−MF}), but if used without **−E**, each **−o** is understood to specify a target object file.

Since **−E** is not implied, **−MD** can be used to generate a dependency output file as a side-effect of the compilation process.

−MMD

Like **−MD** except mention only user header files, not system header files.

−x c
−x c++
−x objective-c
−x assembler-with-cpp

Specify the source language: C, C++, Objective−C, or assembly. This has nothing to do with standards conformance or extensions; it merely selects which base syntax to expect. If you give none of these options, cpp will deduce the language from the extension of the source file: **.c**, **.cc**, **.m**, or **.S**. Some other common extensions for C++ and assembly are also recognized. If cpp does not recognize the extension, it will treat the file as C; this is the most generic mode.

Note: Previous versions of cpp accepted a **−lang** option which selected both the language and the standards conformance level. This option has been removed, because it conflicts with the **−l** option.

−std=*standard*
−ansi

Specify the standard to which the code should conform. Currently CPP knows about C and C++ standards; others may be added in the future.

standard may be one of:

`iso9899:1990`
`c89`

The ISO C standard from 1990. **c89** is the customary shorthand for this version of the standard.

The **−ansi** option is equivalent to **−std=c89**.

`iso9899:199409`

The 1990 C standard, as amended in 1994.

`iso9899:1999`
`c99`
`iso9899:199x`
`c9x`

The revised ISO C standard, published in December 1999. Before publication, this was known as C9X.

`gnu89`

The 1990 C standard plus GNU extensions. This is the default.

`gnu99`
`gnu9x`

The 1999 C standard plus GNU extensions.

`c++98`

The 1998 ISO C++ standard plus amendments.

`gnu++98`
The same as **−std=c++98** plus GNU extensions. This is the default for C++ code.

−I−
Split the include path. Any directories specified with **−I** options before **−I−** are searched only for headers requested with `#include "file"`; they are not searched for `#include <file>`. If additional directories are specified with **−I** options after the **−I−**, those directories are searched for all **#include** directives.

In addition, **−I−** inhibits the use of the directory of the current file directory as the first search directory for `#include "file"`.

This option has been deprecated.

−nostdinc
Do not search the standard system directories for header files. Only the directories you have specified with **−I** options (and the directory of the current file, if appropriate) are searched.

−nostdinc++
Do not search for header files in the C++−specific standard directories, but do still search the other standard directories. (This option is used when building the C++ library.)

−include *file*
Process *file* as if `#include "file"` appeared as the first line of the primary source file. However, the first directory searched for *file* is the preprocessor's working directory *instead of* the directory containing the main source file. If not found there, it is searched for in the remainder of the `#include "..."` search chain as normal.

If multiple **−include** options are given, the files are included in the order they appear on the command line.

−imacros *file*
Exactly like **−include**, except that any output produced by scanning *file* is thrown away. Macros it defines remain defined. This allows you to acquire all the macros from a header without also processing its declarations.

All files specified by **−imacros** are processed before all files specified by **−include**.

−idirafter *dir*
Search *dir* for header files, but do it *after* all directories specified with **−I** and the standard system directories have been exhausted. *dir* is treated as a system include directory.

−iprefix *prefix*
Specify *prefix* as the prefix for subsequent **−iwithprefix** options. If the prefix represents a directory, you should include the final /.

−iwithprefix *dir*
−iwithprefixbefore *dir*
Append *dir* to the prefix specified previously with **−iprefix**, and add the resulting directory to the include search path. **−iwithprefixbefore** puts it in the same place **−I** would; **−iwithprefix** puts it where **−idirafter** would.

−isysroot *dir*
This option is like the **−−sysroot** option, but applies only to header files. See the **−−sysroot** option for more information.

−isystem *dir*
Search *dir* for header files, after all directories specified by **−I** but before the standard system directories. Mark it as a system directory, so that it gets the same special treatment as is applied to the standard system directories.

−iquote *dir*
Search *dir* only for header files requested with `#include "file"`; they are not searched for `#include <file>`, before all directories specified by **−I** and before the standard system directories.

−fdollars−in−identifiers
@anchor{fdollars−in−identifiers} Accept **$** in identifiers.

−fextended−identifiers
Accept universal character names in identifiers. This option is experimental; in a future version of GCC, it will be enabled by default for C99 and C++.

−fpreprocessed
Indicate to the preprocessor that the input file has already been preprocessed. This suppresses things like macro expansion, trigraph conversion, escaped newline splicing, and processing of most directives. The preprocessor still recognizes and removes comments, so that you can pass a file preprocessed with **−C** to the compiler without problems. In this mode the integrated preprocessor is little more than a tokenizer for the front ends.

−fpreprocessed is implicit if the input file has one of the extensions **.i**, **.ii** or **.mi**. These are the extensions that GCC uses for preprocessed files created by **−save−temps**.

−ftabstop=*width*
Set the distance between tab stops. This helps the preprocessor report correct column numbers in warnings or errors, even if tabs appear on the line. If the value is less than 1 or greater than 100, the option is ignored. The default is 8.

−fexec−charset=*charset*
Set the execution character set, used for string and character constants. The default is UTF−8. *charset* can be any encoding supported by the system's `iconv` library routine.

−fwide−exec−charset=*charset*
Set the wide execution character set, used for wide string and character constants. The default is UTF−32 or UTF−16, whichever corresponds to the width of `wchar_t`. As with **−fexec−charset**, *charset* can be any encoding supported by the system's `iconv` library routine; however, you will have problems with encodings that do not fit exactly in `wchar_t`.

−finput−charset=*charset*
Set the input character set, used for translation from the character set of the input file to the source character set used by GCC. If the locale does not specify, or GCC cannot get this information from the locale, the default is UTF−8. This can be overridden by either the locale or this command line option. Currently the command line option takes precedence if there's a conflict. *charset* can be any encoding supported by the system's `iconv` library routine.

−fworking−directory
Enable generation of linemarkers in the preprocessor output that will let the compiler know the current working directory at the time of preprocessing. When this option is enabled, the preprocessor will emit, after the initial linemarker, a second linemarker with the current working directory followed by two slashes. GCC will use this directory, when it's present in the preprocessed input, as the directory emitted as the current working directory in some debugging information formats. This option is implicitly enabled if debugging information is enabled, but this can be inhibited with the negated form **−fno−working−directory**. If the **−P** flag is present in the command line, this option has no effect, since no `#line` directives are emitted whatsoever.

−fno−show−column
Do not print column numbers in diagnostics. This may be necessary if diagnostics are being scanned by a program that does not understand the column numbers, such as **dejagnu**.

−A *predicate=answer*
Make an assertion with the predicate *predicate* and answer *answer*. This form is preferred to the older form **−A** *predicate***(***answer***)**, which is still supported, because it does not use shell special characters.

−A *−predicate=answer*
Cancel an assertion with the predicate *predicate* and answer *answer*.

−dCHARS
CHARS is a sequence of one or more of the following characters, and must not be preceded by a space. Other characters are interpreted by the compiler proper, or reserved for future versions of GCC, and so are silently ignored. If you specify characters whose behavior conflicts, the result is undefined.

M Instead of the normal output, generate a list of **#define** directives for all the macros defined during the execution of the preprocessor, including predefined macros. This gives you a way of finding out what is predefined in your version of the preprocessor. Assuming you have no file *foo.h*, the command

```
touch foo.h; cpp -dM foo.h
```

will show all the predefined macros.

D Like **M** except in two respects: it does *not* include the predefined macros, and it outputs *both* the **#define** directives and the result of preprocessing. Both kinds of output go to the standard output file.

N Like **D**, but emit only the macro names, not their expansions.

I Output **#include** directives in addition to the result of preprocessing.

−P Inhibit generation of linemarkers in the output from the preprocessor. This might be useful when running the preprocessor on something that is not C code, and will be sent to a program which might be confused by the linemarkers.

−C Do not discard comments. All comments are passed through to the output file, except for comments in processed directives, which are deleted along with the directive.

You should be prepared for side effects when using **−C**; it causes the preprocessor to treat comments as tokens in their own right. For example, comments appearing at the start of what would be a directive line have the effect of turning that line into an ordinary source line, since the first token on the line is no longer a **#**.

−CC
Do not discard comments, including during macro expansion. This is like **−C**, except that comments contained within macros are also passed through to the output file where the macro is expanded.

In addition to the side-effects of the **−C** option, the **−CC** option causes all C++−style comments inside a macro to be converted to C−style comments. This is to prevent later use of that macro from inadvertently commenting out the remainder of the source line.

The **−CC** option is generally used to support lint comments.

−traditional−cpp
Try to imitate the behavior of old-fashioned C preprocessors, as opposed to ISO C preprocessors.

−trigraphs
Process trigraph sequences.

−remap
Enable special code to work around file systems which only permit very short file names, such as MS−DOS.

−−help
−−target−help
Print text describing all the command line options instead of preprocessing anything.

−v Verbose mode. Print out GNU CPP's version number at the beginning of execution, and report the final form of the include path.

−H Print the name of each header file used, in addition to other normal activities. Each name is indented to show how deep in the **#include** stack it is. Precompiled header files are also printed, even if they are found to be invalid; an invalid precompiled header file is printed with **...x** and a valid one with **...!** .

−version
−−version
Print out GNU CPP's version number. With one dash, proceed to preprocess as normal. With two dashes, exit immediately.

ENVIRONMENT

This section describes the environment variables that affect how CPP operates. You can use them to specify directories or prefixes to use when searching for include files, or to control dependency output.

Note that you can also specify places to search using options such as **−I**, and control dependency output with options like **−M**. These take precedence over environment variables, which in turn take precedence over the configuration of GCC.

CPATH
C_INCLUDE_PATH
CPLUS_INCLUDE_PATH
OBJC_INCLUDE_PATH
Each variable's value is a list of directories separated by a special character, much like **PATH**, in which to look for header files. The special character, `PATH_SEPARATOR`, is target-dependent and determined at GCC build time. For Microsoft Windows-based targets it is a semicolon, and for almost all other targets it is a colon.

CPATH specifies a list of directories to be searched as if specified with **−I**, but after any paths given with **−I** options on the command line. This environment variable is used regardless of which language is being preprocessed.

The remaining environment variables apply only when preprocessing the particular language indicated. Each specifies a list of directories to be searched as if specified with **−isystem**, but after any paths given with **−isystem** options on the command line.

In all these variables, an empty element instructs the compiler to search its current working directory. Empty elements can appear at the beginning or end of a path. For instance, if the value of **CPATH** is `:/special/include`, that has the same effect as **−I. −I/special/include**.

DEPENDENCIES_OUTPUT
If this variable is set, its value specifies how to output dependencies for Make based on the non-system header files processed by the compiler. System header files are ignored in the dependency output.

The value of **DEPENDENCIES_OUTPUT** can be just a file name, in which case the Make rules are written to that file, guessing the target name from the source file name. Or the value can have the form *file target*, in which case the rules are written to file *file* using *target* as the target name.

In other words, this environment variable is equivalent to combining the options **−MM** and **−MF**, with an optional **−MT** switch too.

SUNPRO_DEPENDENCIES
This variable is the same as **DEPENDENCIES_OUTPUT** (see above), except that system header files are not ignored, so it implies **−M** rather than **−MM**. However, the dependence on the main input file is omitted.

SEE ALSO

gpl (7), *gfdl* (7), *fsf−funding* (7), *gcc* (1), *as* (1), *ld* (1), and the Info entries for *cpp*, *gcc*, and *binutils*.

COPYRIGHT

NAME

crontab − maintain crontab files for individual users (V3)

SYNOPSIS

crontab [-u user] file
crontab [-u user] [-l | -r | -e]

DESCRIPTION

Crontab is the program used to install, deinstall or list the tables used to drive the *cron*(8) daemon in Vixie Cron. Each user can have their own crontab, and though these are files in /var, they are not intended to be edited directly.

If the *allow* file exists, then you must be listed therein in order to be allowed to use this command. If the *allow* file does not exist but the *deny* file does exist, then you must **not** be listed in the *deny* file in order to use this command. If neither of these files exists, then depending on site-dependent configuration parameters, only the super user will be allowed to use this command, or all users will be able to use this command.

The default maximum size for a crontab is 256 kilobytes, but this may be changed for all users on the system by putting the desired maximum size (in bytes) in the *maxtabsize* file.

If the *-u* option is given, it specifies the name of the user whose crontab is to be tweaked. If this option is not given, *crontab* examines "your" crontab, i.e., the crontab of the person executing the command. Note that *su*(1) can confuse *crontab* and that if you are running inside of *su*(1) you should always use the *-u* option for safety's sake.

The first form of this command is used to install a new crontab from some named file or standard input if the pseudo-filename ‘‘-’’ is given.

The *-l* option causes the current crontab to be displayed on standard output.

The *-r* option causes the current crontab to be removed.

The *-e* option is used to edit the current crontab using the editor specified by the VISUAL or EDITOR environment variables. After you exit from the editor, the modified crontab will be installed automatically.

SEE ALSO

crontab(5), cron(8)

FILES

/var/cron/allow
/var/cron/deny
/var/cron/maxtabsize

STANDARDS

The *crontab* command conforms to IEEE Std1003.2-1992 (‘‘POSIX’’). This new command syntax differs from previous versions of Vixie Cron, as well as from the classic SVR3 syntax.

DIAGNOSTICS

A fairly informative usage message appears if you run it with a bad command line.

AUTHOR

Paul Vixie <paul@vix.com>

NAME

crunchgen — generates build environment for a crunched binary

SYNOPSIS

crunchgen [**-fOoq**] [**-c** *c-file-name*] [**-D** *src-root*] [**-d** *build-options*] [**-e** *exec-file-name*] [**-L** *lib-dir*] [**-m** *makefile-name*] [**-v** *var-spec*] *conf-file*

DESCRIPTION

A crunched binary is a program made up of many other programs linked together into a single executable. The crunched binary **main**() function determines which component program to run by the contents of argv[0]. The main reason to crunch programs together is for fitting as many programs as possible onto an installation or system recovery floppy.

crunchgen reads in the specifications in *conf-file* for a crunched binary, and generates a Makefile and accompanying top-level C source file that when built create the crunched executable file from the component programs. For each component program, **crunchgen** can optionally attempt to determine the object (.o) files that make up the program from its source directory Makefile. This information is cached between runs. **crunchgen** uses the companion program *crunchide* to eliminate link-time conflicts between the component programs by hiding all unnecessary symbols.

After **crunchgen** is run, the crunched binary can be built by running "make -f ⟨conf-name⟩.mk". The component programs' object files must already be built. An "objs" target, included in the output makefile, will run make in each component program's source dir to build the object files for the user. This is not done automatically since in release engineering circumstances it is generally not desirable to be modifying objects in other directories.

The options are as follows:

-c *c-file-name*
: Set output C file name to *c-file-name*. The default name is "⟨confname⟩.c".

-D *src-root*
: Assume that relative source directory specifications begin with *src-root*.

-d *build-options*
: Set the DBG variable in the generated makefile to *build-options*. The default flags are -Os.

-e *exec-file-name*
: Set crunched binary executable file name to *exec-file-name*. The default name is "⟨conf-name⟩".

-f
: Flush cache. Forces the recalculation of cached parameters.

-L *lib-dir*
: Try to obtain libraries from *lib-dir*.

-m *makefile-name*
: Set output Makefile name to *makefile-name*. The default name is "⟨conf-name⟩.mk".

-O
: Force **crunchgen** to parse the program's Makefile in determine the list of .o files. Without this option **crunchgen** expects the program's Makefile to have a program.ro target that links all the program objects into a single relocatable.

-o
: Use existing object files. Rather than rebuilding object files via reach-over makefiles, instead search for and use existing object files.

-q Quiet operation. Status messages are suppressed.

-v *varspec*
Append a variable specification to the on-the fly generated Makefile.

CRUNCHGEN CONFIGURATION FILE COMMANDS

crunchgen reads specifications from the *conf-file* that describe the components of the crunched binary. In its simplest use, the component program names are merely listed along with the top-level source directories in which their sources can be found. **crunchgen** then calculates (via the source makefiles) and caches the list of object files and their locations. For more specialized situations, the user can specify by hand all the parameters that **crunchgen** needs.

The *conf-file* commands are as follows:

srcdirs *dirname ...*
A list of source trees in which the source directories of the component programs can be found. These dirs are searched using the BSD "⟨source-dir⟩/⟨progname⟩/" convention. Multiple *srcdirs* lines can be specified. The directories are searched in the order they are given.

progs *progname ...*
A list of programs that make up the crunched binary. Multiple *progs* lines can be specified.

libs *libspec ...*
A list of library specifications to be included in the crunched binary link. Multiple *libs* lines can be specified.

ln *progname linkname*
Causes the crunched binary to invoke *progname* whenever *linkname* appears in argv[0]. This allows programs that change their behavior when run under different names to operate correctly.

To handle specialized situations, such as when the source is not available or not built via a conventional Makefile, the following *special* commands can be used to set **crunchgen** parameters for a component program.

special *progname* **keepsymbols** *symbols ...*
Don't hide the specified symbols for *progname*. Normally all externally visible symbols for a program is hidden to avoid interference. Multiple *keepsymbols* lines can be specified for given *progname*.

special *progname* **srcdir** *pathname*
Set the source directory for *progname*. This is normally calculated by searching the specified *srcdirs* for a directory named *progname*.

special *progname* **objdir** *pathname*
Set the obj directory for *progname*. This is normally calculated by looking for a directory named "obj" under the *srcdir*, and if that is not found, the *srcdir* itself becomes the *objdir*.
Note: This option only takes effect if the -o option to use existing object files is also specified.

special *progname* **objs** *object-file-name ...*
Set the list of object files for program *progname*. This is normally calculated by constructing a temporary makefile that includes "**srcdir / Makefile**" and outputs the value of $(OBJS). Multiple *objs* lines can be specified for given *progname*.

special *progname* **objpaths** *full-pathname-to-object-file ...*
Sets the pathnames of the object files for program *progname*. This is normally calculated by prepending the *objdir* pathname to each file in the **objs** list. Multiple *objpaths* lines can be specified for given *progname*.

Only the *objpaths* parameter is actually needed by **crunchgen** but it is calculated from *objdir* and *objs*, which are in turn calculated from *srcdir*, so is sometimes convenient to specify the earlier parameters and let **crunchgen** calculate forward from there if it can.

The makefile produced by **crunchgen** contains an optional *objs* target that will build the object files for each component program by running make inside that program's source directory. For this to work the *srcdir* and *objs* parameters must also be valid. If they are not valid for a particular program, that program is skipped in the *objs* target.

ENVIRONMENT

MAKEOBJDIRPREFIX If the environment variable MAKEOBJDIRPREFIX is set, the object directory will be prefixed with the path contained in this environment variable.
Note: This variable is only used if the -o option to use existing object files is also specified.

MACHINE If the environment variable MACHINE is set, it is used as the name of the machine type, when accessing object directories of the form obj.MACHINE. If it is not set, it defaults to the machine type returned by uname(3).
Note: This option is only used if the -o option to use existing object files is also specified.

MAKE If the environment variable MAKE is set, it is used as the name of the make(1) executable to be called. If this environment variable is not set, **crunchgen** defaults to "make".

EXAMPLES

Here is an example *crunchgen* input conf file, named "kcopy.conf":

```
srcdirs /usr/src/bin /usr/src/sbin

progs test cp echo sh fsck halt init mount umount myinstall
ln test [       # test can be invoked via [
ln sh -sh       # init invokes the shell with "-sh" in argv[0]

special myprog objpaths /homes/leroy/src/myinstall.o # no sources

libs -lutil -lcrypt
```

This conf file specifies a small crunched binary consisting of some basic system utilities plus a home-grown install program "myinstall", for which no source directory is specified, but its object file is specified directly with the *special* line.

The crunched binary "kcopy" can be built as follows:

```
% crunchgen -m Makefile kcopy.conf    # gen Makefile and kcopy.c
% make objs             # build the component programs' .o files
% make                  # build the crunched binary kcopy
% kcopy sh              # test that this invokes a sh shell
$                      # it works!
```

At this point the binary "kcopy" can be copied onto an install floppy and hard-linked to the names of the component programs.

SEE ALSO

`crunchide`(1), `make`(1)

AUTHORS

`crunchgen` was written by James da Silva ⟨jds@cs.umd.edu⟩.

BUGS

While **`crunchgen`** takes care to eliminate link conflicts between the component programs of a crunched binary, conflicts are still possible between the libraries that are linked in. Some shuffling in the order of libraries may be required, and in some rare cases two libraries may have an unresolvable conflict and thus cannot be crunched together.

Some versions of the BSD build environment do not by default build the intermediate object file for single-source file programs. The “make objs” target must then be used to get those object files built, or some other arrangements made.

If a program directory being searched for is found, but contains no objects, other directories are not searched. This causes the following directive to fail:

```
srcdirs /usr/src/usr.bin /usr/src/usr.bin/less
progs less gzip
```

as the `/usr/src/usr.bin/less` directory will be found with the `/usr/src/usr.bin` *srcdirs* entry, and as it does not contain the require objects, **`crunchgen`** fails to find objects for the *less* program. To avoid this problem, list specific srcdirs first, and the more general ones later, for e.g.:

```
srcdirs /usr/src/usr.bin/less /usr/src/usr.bin
progs less gzip
```

will not have the above problem.

NAME

crunchide — hides symbol names from ld, for crunching programs together

SYNOPSIS

crunchide [**-f** *keep-list-file*] [**-k** *keep-symbol*] *object-file*
[*object-file ...*]

DESCRIPTION

crunchide hides the global symbols of *object-file* such that they are ignored by subsequent runs of the linker, ld(1). Some symbols may be left visible via the **-k** *keep-symbol* and **-f** *keep-list-file* options. The *keep-list-file* must contain a list of symbols to keep visible, one symbol per line. The names given by *keep-symbol* or in *keep-list-file* should be C names. For example, to keep the C function “foo” visible, the option “-k foo” should be used.

crunchide is designed as a companion program for crunchgen(1), which automates the process of creating crunched binaries from multiple component programs.

SEE ALSO

crunchgen(1), ld(1)

AUTHORS

crunchide was written by James da Silva ⟨jds@cs.umd.edu⟩.

Chris Demetriou ⟨cgd@NetBSD.org⟩ reorganized **crunchide** so that it supported multiple object formats, and added ELF object support and ECOFF object recognition.

NAME

csh — a shell (command interpreter) with C-like syntax

SYNOPSIS

csh [**-bcefinstvVxX**] [arg ...]
csh [**-l**]

DESCRIPTION

The **csh** is a command language interpreter incorporating a history mechanism (see **History Substitutions**), job control facilities (see **Jobs**), interactive file name and user name completion (see **File Name Completion**), and a C-like syntax. It is used both as an interactive login shell and a shell script command processor.

Argument list processing

If the first argument (argument 0) to the shell is **-**, then this is a login shell. A login shell also can be specified by invoking the shell with the **-l** flag as the only argument.

The rest of the flag arguments are interpreted as follows:

-b This flag forces a ''break'' from option processing, causing any further shell arguments to be treated as non-option arguments. The remaining arguments will not be interpreted as shell options. This may be used to pass options to a shell script without confusion or possible subterfuge. The shell will not run a set-user ID script without this option.

-c Commands are read from the (single) following argument which must be present. Any remaining arguments are placed in *argv*.

-e The shell exits if any invoked command terminates abnormally or yields a non-zero exit status.

-f The shell will start faster, because it will neither search for nor execute commands from the file `.cshrc` in the invoker's home directory.

-i The shell is interactive and prompts for its top-level input, even if it appears not to be a terminal. Shells are interactive without this option if their inputs and outputs are terminals.

-l The shell is a login shell (only applicable if **-l** is the only flag specified).

-m Read `.cshrc` even if not owned by the user. This flag is normally given only by `su`(1).

-n Commands are parsed, but not executed. This aids in syntactic checking of shell scripts.

-s Command input is taken from the standard input.

-t A single line of input is read and executed. A '\' may be used to escape the newline at the end of this line and continue onto another line.

-v Causes the *verbose* variable to be set, with the effect that command input is echoed after history substitution.

-x Causes the *echo* variable to be set, so that commands are echoed immediately before execution.

-V Causes the *verbose* variable to be set even before `.cshrc` is executed.

-X Is to **-x** as **-V** is to **-v**.

After processing of flag arguments, if arguments remain but none of the **-c**, **-i**, **-s**, or **-t** options were given, the first argument is taken as the name of a file of commands to be executed. The shell opens this file, and saves its name for possible resubstitution by '$0'. Since many systems use either the standard version 6 or version 7 shells whose shell scripts are not compatible with this shell, the shell will execute such a 'standard' shell if the first character of a script is not a '#', i.e., if the script does not start with a comment.

Remaining arguments initialize the variable *argv*.

An instance of **csh** begins by executing commands from the file `/etc/csh.cshrc` and, if this is a login shell, `/etc/csh.login`. It then executes commands from `.cshrc` in the *home* directory of the invoker, and, if this is a login shell, the file `.login` in the same location. It is typical for users on crt's to put the command ''stty crt'' in their `.login` file, and to also invoke `tset`(1) there.

In the normal case, the shell will begin reading commands from the terminal, prompting with '% '. Processing of arguments and the use of the shell to process files containing command scripts will be described later.

The shell repeatedly performs the following actions: a line of command input is read and broken into *words*. This sequence of words is placed on the command history list and parsed. Finally each command in the current line is executed.

When a login shell terminates it executes commands from the files `.logout` in the user's *home* directory and `/etc/csh.logout`.

Lexical structure

The shell splits input lines into words at blanks and tabs with the following exceptions. The characters '&' '|' ';' '<' '>' '(' ')' form separate words. If doubled in '&&', '||', '<<' or '>>' these pairs form single words. These parser metacharacters may be made part of other words, or prevented their special meaning, by preceding them with '\'. A newline preceded by a '\' is equivalent to a blank.

Strings enclosed in matched pairs of quotations, ''', '`' or '"', form parts of a word; metacharacters in these strings, including blanks and tabs, do not form separate words. These quotations have semantics to be described later. Within pairs of ''' or '"' characters, a newline preceded by a '\' gives a true newline character.

When the shell's input is not a terminal, the character '#' introduces a comment that continues to the end of the input line. It is prevented this special meaning when preceded by '\' and in quotations using '`', ''', and '"'.

Commands

A simple command is a sequence of words, the first of which specifies the command to be executed. A simple command or a sequence of simple commands separated by '|' characters forms a pipeline. The output of each command in a pipeline is connected to the input of the next. Sequences of pipelines may be separated by ';', and are then executed sequentially. A sequence of pipelines may be executed without immediately waiting for it to terminate by following it with an '&'.

Any of the above may be placed in '(' ')' to form a simple command (that may be a component of a pipeline, etc.). It is also possible to separate pipelines with '||' or '&&' showing, as in the C language, that the second is to be executed only if the first fails or succeeds respectively. (See **Expressions**.)

Jobs

The shell associates a *job* with each pipeline. It keeps a table of current jobs, printed by the *jobs* command, and assigns them small integer numbers. When a job is started asynchronously with '&', the shell prints a line that looks like:

[1] 1234

showing that the job which was started asynchronously was job number 1 and had one (top-level) process, whose process id was 1234.

If you are running a job and wish to do something else you may hit the key **^Z** (control-Z) which sends a STOP signal to the current job. The shell will then normally show that the job has been 'Stopped', and print another prompt. You can then manipulate the state of this job, putting it in the *background* with the *bg* command, or run some other commands and eventually bring the job back into the foreground with the

foreground command *fg*. A **^Z** takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed. There is another special key **^Y** that does not generate a STOP signal until a program attempts to read(2) it. This request can usefully be typed ahead when you have prepared some commands for a job that you wish to stop after it has read them.

A job being run in the background will stop if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by giving the command ''stty tostop''. If you set this tty option, then background jobs will stop when they try to produce output like they do when they try to read input.

There are several ways to refer to jobs in the shell. The character '%' introduces a job name. If you wish to refer to job number 1, you can name it as '%1'. Just naming a job brings it to the foreground; thus '%1' is a synonym for 'fg %1', bringing job number 1 back into the foreground. Similarly saying '%1 &' resumes job number 1 in the background. Jobs can also be named by prefixes of the string typed in to start them, if these prefixes are unambiguous, thus '%ex' would normally restart a suspended ex(1) job, if there were only one suspended job whose name began with the string 'ex'. It is also possible to say '%?string' which specifies a job whose text contains *string*, if there is only one such job.

The shell maintains a notion of the current and previous jobs. In output about jobs, the current job is marked with a '+' and the previous job with a '−'. The abbreviation '%+' refers to the current job and '%−' refers to the previous job. For close analogy with the syntax of the *history* mechanism (described below), '%%' is also a synonym for the current job.

The job control mechanism requires that the stty(1) option **new** be set. It is an artifact from a *new* implementation of the tty driver that allows generation of interrupt characters from the keyboard to tell jobs to stop. See stty(1) for details on setting options in the new tty driver.

Status reporting

The shell learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work. If, however, you set the shell variable *notify*, the shell will notify you immediately of changes of status in background jobs. There is also a shell command *notify* that marks a single process so that its status changes will be immediately reported. By default *notify* marks the current process; simply say 'notify' after starting a background job to mark it.

When you try to leave the shell while jobs are stopped, you will be warned that 'You have stopped jobs.' You may use the *jobs* command to see what they are. If you try to exit again immediately, the shell will not warn you a second time, and the suspended jobs will be terminated.

File Name Completion

When the file name completion feature is enabled by setting the shell variable *filec* (see **set**), **csh** will interactively complete file names and user names from unique prefixes, when they are input from the terminal followed by the escape character (the escape key, or control-[) For example, if the current directory looks like

```
DSC.OLD  bin      cmd      lib      xmpl.c
DSC.NEW  chaosnet cmtest   mail     xmpl.o
bench    class    dev      mbox     xmpl.out
```

and the input is

```
% vi ch<escape>
```

csh will complete the prefix ''ch'' to the only matching file name ''chaosnet'', changing the input line to

```
% vi chaosnet
```

However, given

```
% vi D<escape>
```

csh will only expand the input to

```
% vi DSC.
```

and will sound the terminal bell to indicate that the expansion is incomplete, since there are two file names matching the prefix "D".

If a partial file name is followed by the end-of-file character (usually control-D), then, instead of completing the name, **csh** will list all file names matching the prefix. For example, the input

```
% vi D<control-D>
```

causes all files beginning with "D" to be listed:

```
DSC.NEW     DSC.OLD
```

while the input line remains unchanged.

The same system of escape and end-of-file can also be used to expand partial user names, if the word to be completed (or listed) begins with the character "~". For example, typing

```
cd ~ro<escape>
```

may produce the expansion

```
cd ~root
```

The use of the terminal bell to signal errors or multiple matches can be inhibited by setting the variable *nobeep*.

Normally, all files in the particular directory are candidates for name completion. Files with certain suffixes can be excluded from consideration by setting the variable *fignore* to the list of suffixes to be ignored. Thus, if *fignore* is set by the command

```
% set fignore = (.o .out)
```

then typing

```
% vi x<escape>
```

would result in the completion to

```
% vi xmpl.c
```

ignoring the files "xmpl.o" and "xmpl.out". However, if the only completion possible requires not ignoring these suffixes, then they are not ignored. In addition, *fignore* does not affect the listing of file names by control-D. All files are listed regardless of their suffixes.

Substitutions

We now describe the various transformations the shell performs on the input in the order in which they occur.

History substitutions

History substitutions place words from previous command input as portions of new commands, making it easy to repeat commands, repeat arguments of a previous command in the current command, or fix spelling mistakes in the previous command with little typing and a high degree of confidence. History substitutions begin with the character '!' and may begin *anywhere* in the input stream (with the proviso that they *do not* nest.) This '!' may be preceded by a '\' to prevent its special meaning; for convenience, an '!' is passed

unchanged when it is followed by a blank, tab, newline, '=' or '('. (History substitutions also occur when an input line begins with '↑'. This special abbreviation will be described later.) Any input line that contains history substitution is echoed on the terminal before it is executed as it would have been typed without history substitution.

Commands input from the terminal that consist of one or more words are saved on the history list. The history substitutions reintroduce sequences of words from these saved commands into the input stream. The size of the history list is controlled by the *history* variable; the previous command is always retained, regardless of the value of the history variable. Commands are numbered sequentially from 1.

For example, consider the following output from the *history* command:

```
 9  write michael
10  ex write.c
11  cat oldwrite.c
12  diff *write.c
```

The commands are shown with their event numbers. It is not usually necessary to use event numbers, but the current event number can be made part of the *prompt* by placing an '!' in the prompt string.

With the current event 13 we can refer to previous events by event number '!11', relatively as in '!−2' (referring to the same event), by a prefix of a command word as in '!d' for event 12 or '!wri' for event 9, or by a string contained in a word in the command as in '!?mic?' also referring to event 9. These forms, without further change, simply reintroduce the words of the specified events, each separated by a single blank. As a special case, '!!' refers to the previous command; thus '!!' alone is a *redo*.

To select words from an event we can follow the event specification by a ':' and a designator for the desired words. The words of an input line are numbered from 0, the first (usually command) word being 0, the second word (first argument) being 1, etc. The basic word designators are:

0	first (command) word
n	*n*'th argument
↑	first argument, i.e., '1'
$	last argument
%	word matched by (immediately preceding) ?*s*? search
x−*y*	range of words
−*y*	abbreviates *'0−y´*
*	abbreviates '↑−$', or nothing if only 1 word in event
*x**	abbreviates *'x−$´*
x−	like *'x*´ but omitting word '$'

The ':' separating the event specification from the word designator can be omitted if the argument selector begins with a '↑', '$', '*', '−' or '%'. After the optional word designator can be placed a sequence of modifiers, each preceded by a ':'. The following modifiers are defined:

h	Remove a trailing pathname component, leaving the head.
r	Remove a trailing '.xxx' component, leaving the root name.
e	Remove all but the extension '.xxx' part.
s*/l/r/*	Substitute *l* for *r*
t	Remove all leading pathname components, leaving the tail.
&	Repeat the previous substitution.
g	Apply the change once on each word, prefixing the above, e.g., 'g&'.
a	Apply the change as many times as possible on a single word, prefixing the above. It can be used together with 'g' to apply a substitution globally.

p Print the new command line but do not execute it.
q Quote the substituted words, preventing further substitutions.
x Like q, but break into words at blanks, tabs and newlines.

Unless preceded by a ‘g’ the change is applied only to the first modifiable word. With substitutions, it is an error for no word to be applicable.

The left hand side of substitutions are not regular expressions in the sense of the editors, but instead strings. Any character may be used as the delimiter in place of ‘/’; a ‘\’ quotes the delimiter into the *l* and *r* strings. The character ‘&’ in the right hand side is replaced by the text from the left. A ‘\’ also quotes ‘&’. A null *l* (‘//’) uses the previous string either from an *l* or from a contextual scan string *s* in ‘!?*s*\?’. The trailing delimiter in the substitution may be omitted if a newline follows immediately as may the trailing ‘?’ in a contextual scan.

A history reference may be given without an event specification, e.g., ‘!$’. Here, the reference is to the previous command unless a previous history reference occurred on the same line in which case this form repeats the previous reference. Thus ‘!?foo?↑ !$’ gives the first and last arguments from the command matching ‘?foo?’.

A special abbreviation of a history reference occurs when the first non-blank character of an input line is a ‘↑’. This is equivalent to ‘!:s↑’ providing a convenient shorthand for substitutions on the text of the previous line. Thus ‘↑lb↑lib’ fixes the spelling of ‘lib’ in the previous command. Finally, a history substitution may be surrounded with ‘{’ and ‘}’ if necessary to insulate it from the characters that follow. Thus, after ‘ls −ld ~paul’ we might do ‘!{l}a’ to do ‘ls −ld ~paula’, while ‘!la’ would look for a command starting with ‘la’.

Quotations with ´ and "

The quotation of strings by ‘´’ and ‘"’ can be used to prevent all or some of the remaining substitutions. Strings enclosed in ‘´’ are prevented any further interpretation. Strings enclosed in ‘"’ may be expanded as described below.

In both cases the resulting text becomes (all or part of) a single word; only in one special case (see *Command Substitution* below) does a ‘"’ quoted string yield parts of more than one word; ‘´’ quoted strings never do.

Alias substitution

The shell maintains a list of aliases that can be established, displayed and modified by the *alias* and *unalias* commands. After a command line is scanned, it is parsed into distinct commands and the first word of each command, left-to-right, is checked to see if it has an alias. If it does, then the text that is the alias for that command is reread with the history mechanism available as though that command were the previous input line. The resulting words replace the command and argument list. If no reference is made to the history list, then the argument list is left unchanged.

Thus if the alias for ‘ls’ is ‘ls −l’ the command ‘ls /usr’ would map to ‘ls −l /usr’, the argument list here being undisturbed. Similarly if the alias for ‘lookup’ was ‘grep !↑ /etc/passwd’ then ‘lookup bill’ would map to ‘grep bill /etc/passwd’.

If an alias is found, the word transformation of the input text is performed and the aliasing process begins again on the reformed input line. Looping is prevented if the first word of the new text is the same as the old by flagging it to prevent further aliasing. Other loops are detected and cause an error.

Note that the mechanism allows aliases to introduce parser metasyntax. Thus, we can ‘alias print ´pr \!* | lpr´’ to make a command that *pr*’s its arguments to the line printer.

Variable substitution

The shell maintains a set of variables, each of which has as value a list of zero or more words. Some of these variables are set by the shell or referred to by it. For instance, the *argv* variable is an image of the shell’s argument list, and words of this variable’s value are referred to in special ways.

The values of variables may be displayed and changed by using the *set* and *unset* commands. Of the variables referred to by the shell a number are toggles; the shell does not care what their value is, only whether they are set or not. For instance, the *verbose* variable is a toggle that causes command input to be echoed. The setting of this variable results from the **-v** command line option.

Other operations treat variables numerically. The '@' command permits numeric calculations to be performed and the result assigned to a variable. Variable values are, however, always represented as (zero or more) strings. For the purposes of numeric operations, the null string is considered to be zero, and the second and additional words of multiword values are ignored.

After the input line is aliased and parsed, and before each command is executed, variable substitution is performed keyed by '$' characters. This expansion can be prevented by preceding the '$' with a '\' except within '"'s where it *always* occurs, and within '´'s where it *never* occurs. Strings quoted by '`' are interpreted later (see **Command substitution** below), so '$' substitution does not occur there until later, if at all. A '$' is passed unchanged if followed by a blank, tab, or end-of-line.

Input/output redirections are recognized before variable expansion, and are variable expanded separately. Otherwise, the command name and entire argument list are expanded together. It is thus possible for the first (command) word (to this point) to generate more than one word, the first of which becomes the command name, and the rest of which become arguments.

Unless enclosed in '"' or given the ':q' modifier the results of variable substitution may eventually be command and filename substituted. Within '"', a variable whose value consists of multiple words expands to a (portion of) a single word, with the words of the variable's value separated by blanks. When the ':q' modifier is applied to a substitution the variable will expand to multiple words with each word separated by a blank and quoted to prevent later command or filename substitution.

The following metasequences are provided for introducing variable values into the shell input. Except as noted, it is an error to reference a variable that is not set.

$name
${name}
: Are replaced by the words of the value of variable *name*, each separated by a blank. Braces insulate *name* from following characters that would otherwise be part of it. Shell variables have names consisting of up to 20 letters and digits starting with a letter. The underscore character is considered a letter. If *name* is not a shell variable, but is set in the environment, then that value is returned (but ':' modifiers and the other forms given below are not available here).

$name[selector]
${name[selector]}
: May be used to select only some of the words from the value of *name*. The selector is subjected to '$' substitution and may consist of a single number or two numbers separated by a '-'. The first word of a variable's value is numbered '1'. If the first number of a range is omitted it defaults to '1'. If the last number of a range is omitted it defaults to '$#name'. The selector '*' selects all words. It is not an error for a range to be empty if the second argument is omitted or in range.

$#name
${#name}
: Gives the number of words in the variable. This is useful for later use in a '$argv[selector]'.

$0
: Substitutes the name of the file from which command input is being read. An error occurs if the name is not known.

$number

${number}
Equivalent to '$argv[number]'.
$* Equivalent to '$argv[*]'.

The modifiers ':e', ':h', ':t', ':r', ':q' and ':x' may be applied to the substitutions above as may ':gh', ':gt' and ':gr'. If braces '{' '}' appear in the command form then the modifiers must appear within the braces. The current implementation allows only one ':' modifier on each '$' expansion.

The following substitutions may not be modified with ':' modifiers.

$?name
${?name}
Substitutes the string '1' if name is set, '0' if it is not.
$?0 Substitutes '1' if the current input filename is known, '0' if it is not.
$$ Substitute the (decimal) process number of the (parent) shell.
$! Substitute the (decimal) process number of the last background process started by this shell.
$< Substitutes a line from the standard input, with no further interpretation. It can be used to read from the keyboard in a shell script.

Command and filename substitution

The remaining substitutions, command and filename substitution, are applied selectively to the arguments of builtin commands. By selectively, we mean that portions of expressions which are not evaluated are not subjected to these expansions. For commands that are not internal to the shell, the command name is substituted separately from the argument list. This occurs very late, after input-output redirection is performed, and in a child of the main shell.

Command substitution

Command substitution is shown by a command enclosed in '`'. The output from such a command is normally broken into separate words at blanks, tabs and newlines, with null words being discarded; this text then replaces the original string. Within '"'s, only newlines force new words; blanks and tabs are preserved.

In any case, the single final newline does not force a new word. Note that it is thus possible for a command substitution to yield only part of a word, even if the command outputs a complete line.

Filename substitution

If a word contains any of the characters '*', '?', '[' or '{' or begins with the character '~', then that word is a candidate for filename substitution, also known as 'globbing'. This word is then regarded as a pattern, and replaced with an alphabetically sorted list of file names that match the pattern. In a list of words specifying filename substitution it is an error for no pattern to match an existing file name, but it is not required for each pattern to match. Only the metacharacters '*', '?' and '[' imply pattern matching, the characters '~' and '{' being more akin to abbreviations.

In matching filenames, the character '.' at the beginning of a filename or immediately following a '/', as well as the character '/' must be matched explicitly. The character '*' matches any string of characters, including the null string. The character '?' matches any single character. The sequence '[...]' matches any one of the characters enclosed. Within '[...]', a pair of characters separated by '-' matches any character lexically between the two (inclusive).

The character '~' at the beginning of a filename refers to home directories. Standing alone, i.e., '~' it expands to the invoker's home directory as reflected in the value of the variable *home*. When followed by a name consisting of letters, digits and '-' characters, the shell searches for a user with that name and substitutes their home directory; thus '~ken' might expand to '/usr/ken' and '~ken/chmach' to '/usr/ken/chmach'. If the character '~' is followed by a character other than a letter or '/' or does not appear at the beginning of a word, it is left undisturbed.

The metanotation ‘a{b,c,d}e’ is a shorthand for ‘abe ace ade’. Left to right order is preserved, with results of matches being sorted separately at a low level to preserve this order. This construct may be nested. Thus, ‘~source/s1/{oldls,ls}.c’ expands to ‘/usr/source/s1/oldls.c /usr/source/s1/ls.c’ without chance of error if the home directory for ‘source’ is ‘/usr/source’. Similarly ‘../{memo,*box}’ might expand to ‘../memo ../box ../mbox’. (Note that ‘memo’ was not sorted with the results of the match to ‘*box’.) As a special case ‘{’, ‘}’ and ‘{}’ are passed undisturbed.

Input/output

The standard input and the standard output of a command may be redirected with the following syntax:

< name Open file *name* (which is first variable, command and filename expanded) as the standard input.

<< word

Read the shell input up to a line that is identical to *word*. *Word* is not subjected to variable, filename or command substitution, and each input line is compared to *word* before any substitutions are done on the input line. Unless a quoting ‘\’, ‘"’, ‘´’ or ‘`’ appears in *word*, variable and command substitution is performed on the intervening lines, allowing ‘\’ to quote ‘$’, ‘\’ and ‘`’. Commands that are substituted have all blanks, tabs, and newlines preserved, except for the final newline which is dropped. The resultant text is placed in an anonymous temporary file that is given to the command as its standard input.

> name

>! name

>& name

>&! name

The file *name* is used as the standard output. If the file does not exist then it is created; if the file exists, it is truncated; its previous contents are lost.

If the variable *noclobber* is set, then the file must not exist or be a character special file (e.g., a terminal or ‘/dev/null’) or an error results. This helps prevent accidental destruction of files. Here, the ‘!’ forms can be used to suppress this check.

The forms involving ‘&’ route the standard error output into the specified file as well as the standard output. *Name* is expanded in the same way as ‘<’ input filenames are.

>> name

>>& name

>>! name

>>&! name

Uses file *name* as the standard output; like ‘>’ but places output at the end of the file. If the variable *noclobber* is set, then it is an error for the file not to exist unless one of the ‘!’ forms is given. Otherwise similar to ‘>’.

A command receives the environment in which the shell was invoked as modified by the input-output parameters and the presence of the command in a pipeline. Thus, unlike some previous shells, commands run from a file of shell commands have no access to the text of the commands by default; instead they receive the original standard input of the shell. The ‘<<’ mechanism should be used to present inline data. This permits shell command scripts to function as components of pipelines and allows the shell to block read its input. Note that the default standard input for a command run detached is *not* modified to be the empty file `/dev/null`; instead the standard input remains as the original standard input of the shell. If this is a terminal and if the process attempts to read from the terminal, then the process will block and the user will be notified (see **Jobs** above).

The standard error output may be directed through a pipe with the standard output. Simply use the form ‘|&’ instead of just ‘|’.

Expressions

Several of the builtin commands (to be described later) take expressions, in which the operators are similar to those of C, with the same precedence, but with the *opposite grouping*: right to left. These expressions appear in the *@*, *exit*, *if*, and *while* commands. The following operators are available:

|| && | ↑ & == != =~ !~ ≤ ≥ < > << >> + − * / % ! ~ ( )

Here the precedence increases to the right, '==' '!=' '=~' and '!~', '≤' '≥' '<' and '>', '<<' and '>>', '+' and '−', '*' '/' and '%' being, in groups, at the same level. The '==' '!=' '=~' and '!~' operators compare their arguments as strings; all others operate on numbers. The operators '=~' and '!~' are like '!=' and '==' except that the right hand side is a *pattern* (containing, e.g., '*'s, '?'s and instances of '[...]') against which the left hand operand is matched. This reduces the need for use of the *switch* statement in shell scripts when all that is really needed is pattern matching.

Strings that begin with '0' are considered octal numbers. Null or missing arguments are considered '0'. The result of all expressions are strings, which represent decimal numbers. It is important to note that no two components of an expression can appear in the same word; except when adjacent to components of expressions that are syntactically significant to the parser ('&' '|' '<' '>' '(' ')'), they should be surrounded by spaces.

Also available in expressions as primitive operands are command executions enclosed in '{' and '}' and file enquiries of the form **-l** *name* where **l** is one of:

```
r       read access
w       write access
x       execute access
e       existence
o       ownership
z       zero size
f       plain file
d       directory
```

The specified name is command and filename expanded and then tested to see if it has the specified relationship to the real user. If the file does not exist or is inaccessible then all enquiries return false, i.e., '0'. Command executions succeed, returning true, i.e., '1', if the command exits with status 0, otherwise they fail, returning false, i.e., '0'. If more detailed status information is required then the command should be executed outside an expression and the variable *status* examined.

Control flow

The shell contains several commands that can be used to regulate the flow of control in command files (shell scripts) and (in limited but useful ways) from terminal input. These commands all operate by forcing the shell to reread or skip in its input and, because of the implementation, restrict the placement of some of the commands.

The **foreach**, **switch**, and **while** statements, as well as the **if-then-else** form of the **if** statement require that the major keywords appear in a single simple command on an input line as shown below.

If the shell's input is not seekable, the shell buffers up input whenever a loop is being read and performs seeks in this internal buffer to accomplish the rereading implied by the loop. (To the extent that this allows, backward goto's will succeed on non-seekable inputs.)

Builtin commands

Builtin commands are executed within the shell. If a builtin command occurs as any component of a pipeline except the last then it is executed in a subshell.

alias
alias *name*
alias *name wordlist*
The first form prints all aliases. The second form prints the alias for name. The final form assigns the specified *wordlist* as the alias of *name*; *wordlist* is command and filename substituted. *Name* is not allowed to be *alias* or *unalias*.

alloc Shows the amount of dynamic memory acquired, broken down into used and free memory. With an argument shows the number of free and used blocks in each size category. The categories start at size 8 and double at each step. This command's output may vary across system types, since systems other than the VAX may use a different memory allocator.

bg
bg %*job ...*
Puts the current or specified jobs into the background, continuing them if they were stopped.

break Causes execution to resume after the **end** of the nearest enclosing **foreach** or **while**. The remaining commands on the current line are executed. Multi-level breaks are thus possible by writing them all on one line.

breaksw
Causes a break from a **switch**, resuming after the **endsw**.

case *label*:
A label in a **switch** statement as discussed below.

cd
cd *name*
chdir
chdir *name*
Change the shell's working directory to directory *name*. If no argument is given then change to the home directory of the user. If *name* is not found as a subdirectory of the current directory (and does not begin with '/', './' or '../'), then each component of the variable **cdpath** is checked to see if it has a subdirectory *name*. Finally, if all else fails but *name* is a shell variable whose value begins with '/', then this is tried to see if it is a directory.

continue
Continue execution of the nearest enclosing **while** or **foreach**. The rest of the commands on the current line are executed.

default:
Labels the default case in a **switch** statement. The default should come after all **case** labels.

dirs Prints the directory stack; the top of the stack is at the left, the first directory in the stack being the current directory.

echo *wordlist*
echo **-n** *wordlist*
The specified words are written to the shell's standard output, separated by spaces, and terminated with a newline unless the **-n** option is specified.

else
end
endif

endsw See the description of the **foreach**, **if**, **switch**, and **while** statements below.

eval *arg ...*
(As in sh(1).) The arguments are read as input to the shell and the resulting command(s) executed in the context of the current shell. This is usually used to execute commands generated as the result of command or variable substitution, since parsing occurs before these substitutions. See tset(1) for an example of using **eval**.

exec *command*
The specified command is executed in place of the current shell.

exit
exit (*expr*)
The shell exits either with the value of the **status** variable (first form) or with the value of the specified **expr** (second form).

fg
fg *%job ...*
Brings the current or specified jobs into the foreground, continuing them if they were stopped.

foreach *name* (*wordlist*)
...
end The variable **name** is successively set to each member of **wordlist** and the sequence of commands between this command and the matching **end** are executed. (Both **foreach** and **end** must appear alone on separate lines.) The builtin command **continue** may be used to continue the loop prematurely and the builtin command **break** to terminate it prematurely. When this command is read from the terminal, the loop is read once prompting with '?' before any statements in the loop are executed. If you make a mistake typing in a loop at the terminal you can rub it out.

glob *wordlist*
Like **echo** but no '\' escapes are recognized and words are delimited by null characters in the output. Useful for programs that wish to use the shell to filename expand a list of words.

goto *word*
The specified **word** is filename and command expanded to yield a string of the form 'label'. The shell rewinds its input as much as possible and searches for a line of the form 'label:' possibly preceded by blanks or tabs. Execution continues after the specified line.

hashstat
Print a statistics line showing how effective the internal hash table has been at locating commands (and avoiding **exec**´s). An **exec** is attempted for each component of the *path* where the hash function indicates a possible hit, and in each component that does not begin with a '/'.

history
history *n*
history **−r** *n*
history **−h** *n*
Displays the history event list; if *n* is given only the *n* most recent events are printed. The **−r** option reverses the order of printout to be most recent first instead of oldest first. The **−h** option causes the history list to be printed without leading numbers. This format produces files suitable for sourcing using the −h option to **source**.

if (*expr*) command
: If the specified expression evaluates true, then the single *command* with arguments is executed. Variable substitution on *command* happens early, at the same time it does for the rest of the **if** command. *Command* must be a simple command, not a pipeline, a command list, or a parenthesized command list. Input/output redirection occurs even if *expr* is false, i.e., when command is *not* executed (this is a bug).

if (*expr*) **then**
...
else if (*expr2*) **then**
...
else
...
endif
: If the specified *expr* is true then the commands up to the first **else** are executed; otherwise if *expr2* is true then the commands up to the second **else** are executed, etc. Any number of **else-if** pairs are possible; only one **endif** is needed. The **else** part is likewise optional. (The words **else** and **endif** must appear at the beginning of input lines; the **if** must appear alone on its input line or after an **else**.)

jobs
jobs -l
: Lists the active jobs; the **-l** option lists process id's in addition to the normal information.

kill %*job*
kill *pid ...*
kill -l [*exit_status*]
kill -s *signal_name pid ...*
kill -*signal_name pid ...*
kill -*signal_number pid ...*
: Sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in ⟨`signal.h`⟩, stripped of the prefix "SIG"). The signal names are listed by "kill −l"; if an *exit_status* is specified, only the corresponding signal name will be written. There is no default, just saying 'kill' does not send a signal to the current job. If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process will be sent a CONT (continue) signal as well.

limit
limit *resource*
limit *resource maximum-use*
limit -h
limit -h *resource*
limit -h *resource maximum-use*
: Manipulates per-process system resource limits via the `getrlimit`(2) and `setrlimit`(2) system calls; this limits the consumption by the current process and each process it creates to not individually exceed *maximum-use* on the specified *resource*. If no *maximum-use* is given, then the current limit is printed; if no *resource* is given, then all limitations are given.

  If the **-h** flag is given, the hard limits are used instead of the current limits. The hard limits impose a ceiling on the values of the current limits. Only the super-user may raise the hard limits, but a user may lower or raise the current limits within the legal range.

Resources controllable currently include:

cputime	The maximum number of CPU-seconds to be used by each process.
filesize	The largest single file (in bytes) that can be created.
datasize	The maximum growth of the data+stack region via sbrk(2) beyond the end of the program text.
stacksize	The maximum size of the automatically-extended stack region.
coredumpsize	The size of the largest core dump (in bytes) that will be created.
memoryuse	The maximum size (in bytes) to which a process's resident set size (RSS) may grow.
memorylocked	The maximum size (in bytes) which a process may lock into memory using the mlock(2) function.
maxproc	The maximum number of simultaneous processes for this user id.
openfiles	The maximum number of simultaneous open files for this user id.
sbsize	The maximum socket buffer size of a process (in bytes).

The *maximum-use* may be given as a (floating point or integer) number followed by a scale factor. For all limits other than *cputime* the default scale is 'k' or 'kilobytes' (1024 bytes); a scale factor of 'm' or 'megabytes' may also be used. For *cputime* the default scale is 'seconds'; a scale factor of 'm' for minutes or 'h' for hours, or a time of the form 'mm:ss' giving minutes and seconds also may be used.

For both *resource* names and scale factors, unambiguous prefixes of the names suffice.

Limits of an arbitrary process can be displayed or set using the sysctl(8) utility. See the getrlimit(2) and setrlimit(2) man pages for an additional description of system resource limits.

login Terminate a login shell, replacing it with an instance of /usr/bin/login. This is one way to log off, included for compatibility with sh(1).

logout
Terminate a login shell. Especially useful if **ignoreeof** is set.

nice
nice *+number*
nice *command*
nice *+number command*
The first form sets the scheduling priority for this shell to 4. The second form sets the priority to the given *number*. The final two forms run command at priority 4 and *number* respectively. The greater the number, the less CPU the process will get. The super-user may specify negative priority by using 'nice −number ...'. *Command* is always executed in a subshell, and the restrictions placed on commands in simple **if** statements apply.

nohup
nohup *command*
The first form can be used in shell scripts to cause hangups to be ignored for the remainder of the script. The second form causes the specified command to be run with hangups ignored. All processes detached with '&' are effectively **nohup**´ed.

notify
notify *%job ...*

Causes the shell to notify the user asynchronously when the status of the current or specified jobs change; normally notification is presented before a prompt. This is automatic if the shell variable **notify** is set.

onintr
onintr -
onintr *label*

Control the action of the shell on interrupts. The first form restores the default action of the shell on interrupts which is to terminate shell scripts or to return to the terminal command input level. The second form 'onintr −' causes all interrupts to be ignored. The final form causes the shell to execute a 'goto label' when an interrupt is received or a child process terminates because it was interrupted.

In any case, if the shell is running detached and interrupts are being ignored, all forms of **onintr** have no meaning and interrupts continue to be ignored by the shell and all invoked commands. Finally **onintr** statements are ignored in the system startup files where interrupts are disabled (/etc/csh.cshrc, /etc/csh.login).

popd
popd *+n*

Pops the directory stack, returning to the new top directory. With an argument `+ *n*´ discards the *n*´th entry in the stack. The members of the directory stack are numbered from the top starting at 0.

pushd
pushd *name*
pushd *+n*

With no arguments, **pushd** exchanges the top two elements of the directory stack. Given a *name* argument, **pushd** changes to the new directory (ala **cd**) and pushes the old current working directory (as in **cwd**) onto the directory stack. With a numeric argument, **pushd** rotates the *n*´th argument of the directory stack around to be the top element and changes to it. The members of the directory stack are numbered from the top starting at 0.

rehash

Causes the internal hash table of the contents of the directories in the **path** variable to be recomputed. This is needed if new commands are added to directories in the **path** while you are logged in. This should only be necessary if you add commands to one of your own directories, or if a systems programmer changes the contents of a system directory.

repeat *count command*

The specified *command*, which is subject to the same restrictions as the *command* in the one line **if** statement above, is executed *count* times. I/O redirections occur exactly once, even if *count* is 0.

set
set *name*
set *name*=word
set *name[index]*=word
set *name*=(wordlist)

The first form of the command shows the value of all shell variables. Variables that have other than a single word as their value print as a parenthesized word list. The second form sets *name* to the null string. The third form sets *name* to the single *word*. The fourth form sets the *index*'th component of *name* to *word*; this component must already exist. The

final form sets *name* to the list of words in *wordlist*. The value is always command and filename expanded.

These arguments may be repeated to set multiple values in a single set command. Note however, that variable expansion happens for all arguments before any setting occurs.

setenv
setenv *name*
setenv *name value*
The first form lists all current environment variables. It is equivalent to `printenv`(1). The last form sets the value of environment variable *name* to be *value*, a single string. The second form sets *name* to an empty string. The most commonly used environment variables `USER`, `TERM`, and `PATH` are automatically imported to and exported from the **csh** variables *user*, *term*, and *path*; there is no need to use **setenv** for these.

shift
shift *variable*
The members of **argv** are shifted to the left, discarding **argv**[1]. It is an error for **argv** not to be set or to have less than one word as value. The second form performs the same function on the specified variable.

source *name*
source −h *name*
The shell reads commands from *name*. **Source** commands may be nested; if they are nested too deeply the shell may run out of file descriptors. An error in a **source** at any level terminates all nested **source** commands. Normally input during **source** commands is not placed on the history list; the −h option causes the commands to be placed on the history list without being executed.

stop
stop *%job ...*
Stops the current or specified jobs that are executing in the background.

suspend
Causes the shell to stop in its tracks, much as if it had been sent a stop signal with **^Z**. This is most often used to stop shells started by `su`(1).

switch (*string*)
case *str1*:
...
breaksw
...
default:
...
breaksw
endsw Each case label is successively matched against the specified *string* which is first command and filename expanded. The file metacharacters '∗', '?' and '[...]' may be used in the case labels, which are variable expanded. If none of the labels match before the 'default' label is found, then the execution begins after the default label. Each case label and the default label must appear at the beginning of a line. The command **breaksw** causes execution to continue after the **endsw**. Otherwise control may fall through case labels and the default label as in C. If no label matches and there is no default, execution continues after the **endsw**.

time
time *command*
With no argument, a summary of time used by this shell and its children is printed. If arguments are given the specified simple command is timed and a time summary as described under the **time** variable is printed. If necessary, an extra shell is created to print the time statistic when the command completes.

umask
umask *value*
The file creation mask is displayed (first form) or set to the specified value (second form). The mask is given in octal. Common values for the mask are 002 giving all access to the group and read and execute access to others or 022 giving all access except write access for users in the group or others.

unalias *pattern*
All aliases whose names match the specified pattern are discarded. Thus all aliases are removed by 'unalias *'. It is not an error for nothing to be **unaliased**.

unhash
Use of the internal hash table to speed location of executed programs is disabled.

unlimit
unlimit *resource*
unlimit **-h**
unlimit **-h** *resource*
Removes the limitation on *resource*. If no *resource* is specified, then all *resource* limitations are removed. If **-h** is given, the corresponding hard limits are removed. Only the super-user may do this.

unset *pattern*
All variables whose names match the specified pattern are removed. Thus all variables are removed by 'unset *'; this has noticeably distasteful side-effects. It is not an error for nothing to be **unset**.

unsetenv *pattern*
Removes all variables whose name match the specified pattern from the environment. See also the **setenv** command above and printenv(1).

wait Wait for all background jobs. If the shell is interactive, then an interrupt can disrupt the wait. After the interrupt, the shell prints names and job numbers of all jobs known to be outstanding.

which *command*
Displays the resolved command that will be executed by the shell.

while (*expr*)
...
end While the specified expression evaluates non-zero, the commands between the **while** and the matching **end** are evaluated. **Break** and **continue** may be used to terminate or continue the loop prematurely. (The **while** and **end** must appear alone on their input lines.) Prompting occurs here the first time through the loop as for the **foreach** statement if the input is a terminal.

%*job* Brings the specified job into the foreground.

%job **&**
Continues the specified job in the background.

@
@ *name*= expr
@ *name[index]*= expr
The first form prints the values of all the shell variables. The second form sets the specified *name* to the value of *expr*. If the expression contains '<', '>', '&' or '|' then at least this part of the expression must be placed within '(' ')'. The third form assigns the value of *expr* to the *index*'th argument of *name*. Both *name* and its *index*'th component must already exist.

The operators '*=', '+=', etc are available as in C. The space separating the name from the assignment operator is optional. Spaces are, however, mandatory in separating components of *expr* which would otherwise be single words.

Special postfix '++' and '--' operators increment and decrement *name* respectively, i.e., '@ i++'.

Pre-defined and environment variables

The following variables have special meaning to the shell. Of these, *argv*, *cwd*, *home*, *path*, *prompt*, *shell* and *status* are always set by the shell. Except for *cwd* and *status*, this setting occurs only at initialization; these variables will not then be modified unless done explicitly by the user.

The shell copies the environment variable USER into the variable *user*, TERM into *term*, and HOME into *home*, and copies these back into the environment whenever the normal shell variables are reset. The environment variable PATH is likewise handled; it is not necessary to worry about its setting other than in the file *.cshrc* as inferior **csh** processes will import the definition of *path* from the environment, and re-export it if you then change it.

argv Set to the arguments to the shell, it is from this variable that positional parameters are substituted, i.e., '$1' is replaced by '$argv[1]', etc.

cdpath Gives a list of alternative directories searched to find subdirectories in *chdir* commands.

cwd The full pathname of the current directory.

echo Set when the **-x** command line option is given. Causes each command and its arguments to be echoed just before it is executed. For non-builtin commands all expansions occur before echoing. Builtin commands are echoed before command and filename substitution, since these substitutions are then done selectively.

filec Enable file name completion.

histchars Can be given a string value to change the characters used in history substitution. The first character of its value is used as the history substitution character, replacing the default character '!'. The second character of its value replaces the character '^' in quick substitutions.

histfile Can be set to the pathname where history is going to be saved/restored.

history Can be given a numeric value to control the size of the history list. Any command that has been referenced in this many events will not be discarded. Too large values of *history* may run the shell out of memory. The last executed command is always saved on the history list.

home The home directory of the invoker, initialized from the environment. The filename expansion of '~' refers to this variable.

ignoreeof If set the shell ignores end-of-file from input devices which are terminals. This prevents shells from accidentally being killed by control-D's.

mail The files where the shell checks for mail. This checking is done after each command completion that will result in a prompt, if a specified interval has elapsed. The shell says 'You have new mail.' if the file exists with an access time not greater than its modify time.

If the first word of the value of *mail* is numeric it specifies a different mail checking interval, in seconds, than the default, which is 10 minutes.

If multiple mail files are specified, then the shell says 'New mail in *name*' when there is mail in the file *name*.

noclobber As described in the section on **input/output**, restrictions are placed on output redirection to ensure that files are not accidentally destroyed, and that '>>' redirections refer to existing files.

noglob If set, filename expansion is inhibited. This inhibition is most useful in shell scripts that are not dealing with filenames, or after a list of filenames has been obtained and further expansions are not desirable.

nonomatch If set, it is not an error for a filename expansion to not match any existing files; instead the primitive pattern is returned. It is still an error for the primitive pattern to be malformed, i.e., 'echo [' still gives an error.

notify If set, the shell notifies asynchronously of job completions; the default is to present job completions just before printing a prompt.

path Each word of the path variable specifies a directory in which commands are to be sought for execution. A null word specifies the current directory. If there is no *path* variable then only full path names will execute. The usual search path is '.', '/bin' and '/usr/bin', but this may vary from system to system. For the super-user the default search path is '/etc', '/bin' and '/usr/bin'. A shell that is given neither the **-c** nor the **-t** option will normally hash the contents of the directories in the *path* variable after reading *.cshrc*, and each time the *path* variable is reset. If new commands are added to these directories while the shell is active, it may be necessary to do a **rehash** or the commands may not be found.

prompt The string that is printed before each command is read from an interactive terminal input. If a '!' appears in the string it will be replaced by the current event number unless a preceding '\' is given. Default is '% ', or '# ' for the super-user.

savehist Is given a numeric value to control the number of entries of the history list that are saved in ~/.history when the user logs out. Any command that has been referenced in this many events will be saved. During start up the shell sources ~/.history into the history list enabling history to be saved across logins. Too large values of *savehist* will slow down the shell during start up. If *savehist* is just set, the shell will use the value of *history*.

shell The file in which the shell resides. This variable is used in forking shells to interpret files that have execute bits set, but which are not executable by the system. (See the description of **Non-builtin Command Execution** below.) Initialized to the (system-dependent) home of the shell.

status The status returned by the last command. If it terminated abnormally, then 0200 is added to the status. Builtin commands that fail return exit status '1', all other builtin commands set status to '0'.

time Controls automatic timing of commands. This setting allows two parameters. The first specifies the CPU time threshold at which reporting should be done for a process, and the optional second specifies the output format. The following format strings are available:

%c	Number of involuntary context switches.
%D	Average unshared data size.
%E	Elapsed (wall−clock) time.
%F	Page faults.
%I	Filesystem blocks in.
%K	Average total data memory used.
%k	Number of signals received.
%M	Maximum Resident Set Size.
%O	Filesystem blocks out.
%P	Total percent time spent running.
%R	Page reclaims.
%r	Socket messages received.
%S	Total system CPU time used.
%s	Socket messages sent.
%U	Total user CPU time used.
%W	Number of swaps.
%w	Number of voluntary context switches (waits).
%X	Average shared text size.

The default summary is "%Uu %Ss %E %P %X+%Dk %I+%Oio %Fpf+%Ww"

verbose Set by the **−v** command line option, causes the words of each command to be printed after history substitution.

Non-builtin command execution

When a command to be executed is found to not be a builtin command the shell attempts to execute the command via execve(2). Each word in the variable *path* names a directory from which the shell will attempt to execute the command. If it is given neither a **−c** nor a **−t** option, the shell will hash the names in these directories into an internal table so that it will only try an **exec** in a directory if there is a possibility that the command resides there. This shortcut greatly speeds command location when many directories are present in the search path. If this mechanism has been turned off (via **unhash**), or if the shell was given a **−c** or **−t** argument, and in any case for each directory component of *path* that does not begin with a '/', the shell concatenates with the given command name to form a path name of a file which it then attempts to execute.

Parenthesized commands are always executed in a subshell. Thus

```
(cd; pwd); pwd
```

prints the *home* directory; leaving you where you were (printing this after the home directory), while

```
cd; pwd
```

leaves you in the *home* directory. Parenthesized commands are most often used to prevent **chdir** from affecting the current shell.

If the file has execute permissions but is not an executable binary to the system, then it is assumed to be a file containing shell commands and a new shell is spawned to read it.

If there is an **alias** for **shell** then the words of the alias will be prepended to the argument list to form the shell command. The first word of the **alias** should be the full path name of the shell (e.g., '$shell'). Note that this is a special, late occurring, case of **alias** substitution, and only allows words to be prepended to the argument list without change.

Signal handling

The shell normally ignores *quit* signals. Jobs running detached (either by **&** or the **bg** or **%... &** commands) are immune to signals generated from the keyboard, including hangups. Other signals have the val-

ues which the shell inherited from its parent. The shell's handling of interrupts and terminate signals in shell scripts can be controlled by **onintr**. Login shells catch the *terminate* signal; otherwise this signal is passed on to children from the state in the shell's parent. Interrupts are not allowed when a login shell is reading the file `.logout`.

FILES

`~/.cshrc`	Read at beginning of execution by each shell.
`~/.login`	Read by login shell, after '.cshrc' at login.
`~/.logout`	Read by login shell, at logout.
`/bin/sh`	Standard shell, for shell scripts not starting with a '#'.
`/tmp/sh*`	Temporary file for '<<'.
`/etc/passwd`	Source of home directories for '~name'.

LIMITATIONS

Word lengths − Words can be no longer than 1024 characters. The system limits argument lists to 10240 characters. The number of arguments to a command that involves filename expansion is limited to 1/6'th the number of characters allowed in an argument list. Command substitutions may substitute no more characters than are allowed in an argument list. To detect looping, the shell restricts the number of **alias** substitutions on a single line to 20.

SEE ALSO

`sh`(1), `access`(2), `execve`(2), `fork`(2), `pipe`(2), `setrlimit`(2), `sigaction`(2), `umask`(2), `wait`(2), `killpg`(3), `tty`(4), `a.out`(5), `environ`(7), `sysctl`(8)
An introduction to the C shell

HISTORY

csh appeared in 3BSD. It was a first implementation of a command language interpreter incorporating a history mechanism (see **History Substitutions**), job control facilities (see **Jobs**), interactive file name and user name completion (see **File Name Completion**), and a C-like syntax. There are now many shells that also have these mechanisms, plus a few more (and maybe some bugs too), which are available through the usenet.

AUTHORS

William Joy. Job control and directory stack features first implemented by J.E. Kulp of IIASA, Laxenburg, Austria, with different syntax than that used now. File name completion code written by Ken Greer, HP Labs. Eight-bit implementation Christos S. Zoulas, Cornell University.

BUGS

When a command is restarted from a stop, the shell prints the directory it started in if this is different from the current directory; this can be misleading (i.e., wrong) as the job may have changed directories internally.

Shell builtin functions are not stoppable/restartable. Command sequences of the form 'a ; b ; c' are also not handled gracefully when stopping is attempted. If you suspend 'b', the shell will immediately execute 'c'. This is especially noticeable if this expansion results from an *alias*. It suffices to place the sequence of commands in ()'s to force it to a subshell, i.e., '(a ; b ; c)'.

Control over tty output after processes are started is primitive; perhaps this will inspire someone to work on a good virtual terminal interface. In a virtual terminal interface much more interesting things could be done with output control.

Alias substitution is most often used to clumsily simulate shell procedures; shell procedures should be provided instead of aliases.

Commands within loops, prompted for by '?', are not placed on the **history** list. Control structure should be parsed instead of being recognized as built-in commands. This would allow control commands to be placed anywhere, to be combined with '|', and to be used with '&' and ';' metasyntax.

It should be possible to use the ':' modifiers on the output of command substitutions.

The way the **filec** facility is implemented is ugly and expensive.

NAME

csplit — split files based on context

SYNOPSIS

csplit [**-ks**] [**-f** *prefix*] [**-n** *number*] *file args ...*

DESCRIPTION

The **csplit** utility splits *file* into pieces using the patterns *args*. If *file* is a dash (‘-’), **csplit** reads from standard input.

The options are as follows:

-f *prefix*
: Give created files names beginning with *prefix*. The default is “xx”.

-k
: Do not remove output files if an error occurs or a HUP, INT, or TERM signal is received.

-n *number*
: Use *number* of decimal digits after the *prefix* to form the file name. The default is 2.

-s
: Do not write the size of each output file to standard output as it is created.

The *args* operands may be a combination of the following patterns:

/*regexp*/[[**+**|**-**]*offset*]
: Create a file containing the input from the current line to (but not including) the next line matching the given basic regular expression. An optional *offset* from the line that matched may be specified.

%*regexp*%[[**+**|**-**]*offset*]
: Same as above but a file is not created for the output.

line_no
: Create containing the input from the current line to (but not including) the specified line number.

{*num*}
: Repeat the previous pattern the specified number of times. If it follows a line number pattern, a new file will be created for each *line_no* lines, *num* times. The first line of the file is line number 1 for historic reasons.

After all the patterns have been processed, the remaining input data (if there is any) will be written to a new file.

Requesting to split at a line before the current line number or past the end of the file will result in an error.

The **csplit** utility exits 0 on success, and >0 if an error occurs.

ENVIRONMENT

The LANG, LC_ALL, LC_COLLATE, and LC_CTYPE environment variables affect the execution of **csplit** as described in environ(7).

EXAMPLES

Split the mdoc(7) file foo.1 into one file for each section (up to 20):

```
$ csplit -k foo.1 '%^\.Sh%' '/^\.Sh/' '{20}'
```

Split standard input after the first 99 lines and every 100 lines thereafter:

```
$ csplit -k - 100 '{19}'
```

SEE ALSO

sed(1), split(1), re_format(7)

STANDARDS

The **csplit** utility conforms to IEEE Std 1003.1-2004 " ("POSIX.1").

HISTORY

A **csplit** command appeared in PWB UNIX.

BUGS

Input lines are limited to LINE_MAX (2048) bytes in length.

NAME

ctags — create a tags file

SYNOPSIS

ctags [**-BFadtuwvx**] [**-f** *tagsfile*] *name ...*

DESCRIPTION

ctags makes a tags file for ex(1) from the specified C, Pascal, Fortran, YACC, lex, and lisp sources. A tags file gives the locations of specified objects in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and a search pattern for the object definition, separated by white-space. Using the *tags* file, ex(1) can quickly locate these object definitions. Depending upon the options provided to **ctags**, objects will consist of subroutines, typedefs, defines, structs, enums and unions.

-B use backward searching patterns (`?...?`).

-F use forward searching patterns (`/.../`) (the default).

-a append to *tags* file.

-d create tags for `#defines` that don't take arguments; `#defines` that take arguments are tagged automatically.

-f Places the tag descriptions in a file called *tagsfile*. The default behaviour is to place them in a file called *tags*.

-t create tags for typedefs, structs, unions, and enums.

-u update the specified files in the *tags* file, that is, all references to them are deleted, and the new values are appended to the file. (Beware: this option is implemented in a way which is rather slow; it is usually faster to simply rebuild the *tags* file.)

-v An index of the form expected by vgrind(1) is produced on the standard output. This listing contains the object name, file name, and page number (assuming 64 line pages). Since the output will be sorted into lexicographic order, it may be desired to run the output through sort(1). Sample use:

```
ctags −v files | sort −f > index
vgrind −x index
```

-w suppress warning diagnostics.

-x **ctags** produces a list of object names, the line number and file name on which each is defined, as well as the text of that line and prints this on the standard output. This is a simple index which can be printed out as an off-line readable function index.

Files whose names end in '.c' or '.h' are assumed to be C source files and are searched for C style routine and macro definitions. Files whose names end in '.y' are assumed to be YACC source files. Files whose names end in '.l' are assumed to be lisp files if their first non-blank character is ';', '(', or '[', otherwise, they are treated as lex files. Other files are first examined to see if they contain any Pascal or Fortran routine definitions, and, if not, are searched for C style definitions.

The tag `main` is treated specially in C programs. The tag formed is created by prepending *M* to the name of the file, with the trailing '.c' and any leading pathname components removed. This makes use of **ctags** practical in directories with more than one program.

Yacc and lex files each have a special tag. *Yyparse* is the start of the second section of the yacc file, and *yylex* is the start of the second section of the lex file.

EXIT STATUS

ctags exits with a value of 1 if an error occurred, 0 otherwise. Duplicate objects are not considered errors.

FILES

`tags` default output tags file

SEE ALSO

`ex`(1), `vi`(1)

HISTORY

The **ctags** command appeared in 3.0BSD.

BUGS

Recognition of *functions*, *subroutines* and *procedures* for FORTRAN and Pascal is done in a very simple-minded way. No attempt is made to deal with block structure; if you have two Pascal procedures in different blocks with the same name you lose. **ctags** doesn't understand about Pascal types.

The method of deciding whether to look for C, Pascal or FORTRAN functions is a hack.

ctags relies on the input being well formed, and any syntactical errors will completely confuse it. It also finds some legal syntax confusing; for example, since it doesn't understand `#ifdef`'s (incidentally, that's a feature, not a bug), any code with unbalanced braces inside `#ifdef`'s will cause it to become somewhat disoriented. In a similar fashion, multiple line changes within a definition will cause it to enter the last line of the object, rather than the first, as the searching pattern. The last line of multiple line `typedef`'s will similarly be noted.

NAME

cut — select portions of each line of a file

SYNOPSIS

cut −b *list* [**−n**] [*file ...*]
cut −c *list* [*file ...*]
cut −f *list* [**−d** *delim*] [**−s**] [*file ...*]

DESCRIPTION

The **cut** utility selects portions of each line (as specified by *list*) from each *file* and writes them to the standard output. If the *file* argument is a single dash (‘-’) or no *file* arguments were specified, lines are read from the standard input. The items specified by *list* can be in terms of column position or in terms of fields delimited by a special character. Column numbering starts from 1.

List is a comma or whitespace separated set of increasing numbers and/or number ranges. Number ranges consist of a number, a dash (−), and a second number and select the fields or columns from the first number to the second, inclusive. Numbers or number ranges may be preceded by a dash, which selects all fields or columns from 1 to the first number. Numbers or number ranges may be followed by a dash, which selects all fields or columns from the last number to the end of the line. Numbers and number ranges may be repeated, overlapping, and in any order. It is not an error to select fields or columns not present in the input line.

The options are as follows:

−b *list* The *list* specifies byte positions.

−c *list* The *list* specifies character positions.

−d *string*
Use the first character of *string* as the field delimiter character instead of the tab character.

−f *list* The *list* specifies fields, delimited in the input by a single tab character. Output fields are separated by a single tab character.

−n Do not split multi-byte characters.

−s Suppresses lines with no field delimiter characters. Unless specified, lines with no delimiters are passed through unmodified.

EXIT STATUS

cut exits 0 on success, 1 if an error occurred.

SEE ALSO

paste(1)

STANDARDS

The **cut** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

NAME

cvs − Concurrent Versions System

SYNOPSIS

cvs [*cvs_options*]
cvs_command [*command_options*] [*command_args*]

NOTE

This manpage is a summary of some of the features of **cvs**. It is auto-generated from an appendix of the CVS manual. For more in-depth documentation, please consult the Cederqvist manual (via the **info CVS** command or otherwise, as described in the SEE ALSO section of this manpage). Cross-references in this man page refer to nodes in the same.

CVS commands

Guide to CVS commands

This appendix describes the overall structure of **cvs** commands, and describes some commands in detail (others are described elsewhere; for a quick reference to **cvs** commands, see node ‘Invoking CVS' in the CVS manual).

Structure

Overall structure of CVS commands

The overall format of all **cvs** commands is:

cvs [cvs_options] cvs_command [command_options] [command_args]

cvs

The name of the **cvs** program.

cvs_options

Some options that affect all sub-commands of **cvs**. These are described below.

cvs_command

One of several different sub-commands. Some of the commands have aliases that can be used instead; those aliases are noted in the reference manual for that command. There are only two situations where you may omit **cvs_command**: **cvs -H** elicits a list of available commands, and **cvs -v** displays version information on **cvs** itself.

command_options

Options that are specific for the command.

command_args

Arguments to the commands.

There is unfortunately some confusion between **cvs_options** and **command_options**. When given as a **cvs_option**, some options only affect some of the commands. When given as a **command_option** it may have a different meaning, and be accepted by more commands. In other words, do not take the above categorization too seriously. Look at the documentation instead.

Exit status

CVS's exit status

cvs can indicate to the calling environment whether it succeeded or failed by setting its *exit status*. The exact way of testing the exit status will vary from one operating system to another. For example in a unix shell script the **$?** variable will be 0 if the last command returned a successful exit status, or greater than 0 if the exit status indicated failure.

If **cvs** is successful, it returns a successful status; if there is an error, it prints an error message and returns a failure status. The one exception to this is the **cvs diff** command. It will return a successful status if it found no differences, or a failure status if there were differences or if there was an error. Because this behavior provides no good way to detect errors, in the future it is possible that **cvs diff** will be changed to behave like the other **cvs** commands.

~/.cvsrc

Default options and the ~/.cvsrc file

There are some **command_options** that are used so often that you might have set up an alias or some other means to make sure you always specify that option. One example (the one that drove the implementation of the **.cvsrc** support, actually) is that many people find the default output of the **diff** command to be very hard to read, and that either context diffs or unidiffs are much easier to understand.

The **~/.cvsrc** file is a way that you can add default options to **cvs_commands** within cvs, instead of relying on aliases or other shell scripts.

The format of the **~/.cvsrc** file is simple. The file is searched for a line that begins with the same name as the **cvs_command** being executed. If a match is found, then the remainder of the line is split up (at white-space characters) into separate options and added to the command arguments *before* any options from the command line.

If a command has two names (e.g., **checkout** and **co**), the official name, not necessarily the one used on the command line, will be used to match against the file. So if this is the contents of the user's **~/.cvsrc** file:

```
log -N
diff -uN
rdiff -u
update -Pd
checkout -P
release -d
```

the command **cvs checkout foo** would have the **-P** option added to the arguments, as well as **cvs co foo**.

With the example file above, the output from **cvs diff foobar** will be in unidiff format. **cvs diff -c foobar** will provide context diffs, as usual. Getting "old" format diffs would be slightly more complicated, because **diff** doesn't have an option to specify use of the "old" format, so you would need **cvs -f diff foobar**.

In place of the command name you can use **cvs** to specify global options (see node ‘Global options' in the CVS manual). For example the following line in **.cvsrc**

```
cvs -z6
```

causes **cvs** to use compression level 6.

Global options

The available **cvs_options** (that are given to the left of **cvs_command**) are:

--allow-root=*rootdir*

Specify legal **cvsroot** directory. See see node ‘Password authentication server' in the CVS manual.

-a

Authenticate all communication between the client and the server. Only has an effect on the **cvs** client. As of this writing, this is only implemented when using a GSSAPI connection (see node ‘GSSAPI authenticated' in the CVS manual). Authentication prevents certain sorts of attacks involving hijacking the active **tcp** connection. Enabling authentication does not enable encryption.

-b *bindir*

In **cvs** 1.9.18 and older, this specified that **rcs** programs are in the *bindir* directory. Current versions of **cvs** do not run **rcs** programs; for compatibility this option is accepted, but it does nothing.

-T *tempdir*

Use *tempdir* as the directory where temporary files are located. Overrides the setting of the **$TMPDIR** environment variable and any precompiled directory. This parameter should be specified as an absolute pathname. (When running client/server, **-T** affects only the local process; specifying **-T** for the client has no effect on the server and vice versa.)

-d *cvs_root_directory*

Use *cvs_root_directory* as the root directory pathname of the repository. Overrides the setting of the **$CVSROOT** environment variable. see node ‘Repository' in the CVS manual.

-e *editor*

Use *editor* to enter revision log information. Overrides the setting of the **$CVSEDITOR** and **$EDITOR** environment variables. For more information, see see node ‘Committing your changes' in the CVS manual.

-f

Do not read the **~/.cvsrc** file. This option is most often used because of the non-orthogonality of the **cvs** option set. For example, the **cvs log** option **-N** (turn off display of tag names) does not have a corresponding option to turn the display on. So if you have **-N** in the **~/.cvsrc** entry for **log**, you may need to use **-f** to show the tag names.

-H

--help

Display usage information about the specified **cvs_command** (but do not actually execute the command). If you don't specify a command name, **cvs -H** displays overall help for **cvs**, including a list of other help options.

-n

Do not change any files. Attempt to execute the **cvs_command**, but only to issue reports; do not remove, update, or merge any existing files, or create any new files.

Note that **cvs** will not necessarily produce exactly the same output as without **-n**. In some cases the output will be the same, but in other cases **cvs** will skip some of the processing that would have been required to produce the exact same output.

-Q

Cause the command to be really quiet; the command will only generate output for serious problems.

-q

Cause the command to be somewhat quiet; informational messages, such as reports of recursion through subdirectories, are suppressed.

-r

Make new working files read-only. Same effect as if the **$CVSREAD** environment variable is set (see node ‘Environment variables' in the CVS manual). The default is to make working files writable, unless watches are on (see node ‘Watches' in the CVS manual).

-s *variable=value*

Set a user variable (see node ‘Variables' in the CVS manual).

-t

Trace program execution; display messages showing the steps of **cvs** activity. Particularly useful with **-n** to explore the potential impact of an unfamiliar command.

-v

--version

Display version and copyright information for **cvs**.

-w

Make new working files read-write. Overrides the setting of the **$CVSREAD** environment variable. Files are created read-write by default, unless **$CVSREAD** is set or **-r** is given.

-x

Encrypt all communication between the client and the server. Only has an effect on the **cvs** client. As of this writing, this is only implemented when using a GSSAPI connection (see node ‘GSSAPI authenticated' in the CVS manual) or a Kerberos connection (see node ‘Kerberos authenticated' in the CVS manual). Enabling encryption implies that message traffic is also authenticated. Encryption support is not

available by default; it must be enabled using a special configure option, **--enable-encryption**, when you build **cvs**.

-z *gzip-level*

Set the compression level. Valid levels are 1 (high speed, low compression) to 9 (low speed, high compression), or 0 to disable compression (the default). Only has an effect on the **cvs** client.

Common options

Common command options

This section describes the **command_options** that are available across several **cvs** commands. These options are always given to the right of **cvs_command**. Not all commands support all of these options; each option is only supported for commands where it makes sense. However, when a command has one of these options you can almost always count on the same behavior of the option as in other commands. (Other command options, which are listed with the individual commands, may have different behavior from one **cvs** command to the other).

The history command is an exception; it supports many options that conflict even with these standard options.

-D *date_spec*

Use the most recent revision no later than *date_spec*. *date_spec* is a single argument, a date description specifying a date in the past.

The specification is *sticky* when you use it to make a private copy of a source file; that is, when you get a working file using **-D**, **cvs** records the date you specified, so that further updates in the same directory will use the same date (for more information on sticky tags/dates, see node ‘Sticky tags' in the CVS manual).

-D is available with the **annotate**, **checkout**, **diff**, **export**, **history**, **rdiff**, **rtag**, and **update** commands. (The **history** command uses this option in a slightly different way; see node ‘history options' in the CVS manual).

A wide variety of date formats are supported by **cvs**. The most standard ones are ISO8601 (from the International Standards Organization) and the Internet e-mail standard (specified in RFC822 as amended by RFC1123).

ISO8601 dates have many variants but a few examples are:

```
1972-09-24
1972-09-24 20:05
```

There are a lot more ISO8601 date formats, and **cvs** accepts many of them, but you probably don't want to hear the *whole* long story :-).

In addition to the dates allowed in Internet e-mail itself, **cvs** also allows some of the fields to be omitted. For example:

```
24 Sep 1972 20:05
24 Sep
```

The date is interpreted as being in the local timezone, unless a specific timezone is specified.

These two date formats are preferred. However, **cvs** currently accepts a wide variety of other date formats. They are intentionally not documented here in any detail, and future versions of **cvs** might not accept all of them.

One such format is *month/day/year*. This may confuse people who are accustomed to having the month and day in the other order; **1/4/96** is January 4, not April 1.

Remember to quote the argument to the **-D** flag so that your shell doesn't interpret spaces as argument separators. A command using the **-D** flag can look like this:

```
$ cvs diff -D "1 hour ago" cvs.texinfo
```

-f

When you specify a particular date or tag to **cvs** commands, they normally ignore files that do not contain the tag (or did not exist prior to the date) that you specified. Use the **-f** option if you want files retrieved even when there is no match for the tag or date. (The most recent revision of the file will be used).

Note that even with **-f**, a tag that you specify must exist (that is, in some file, not necessary in every file). This is so that **cvs** will continue to give an error if you mistype a tag name.

-f is available with these commands: **annotate**, **checkout**, **export**, **rdiff**, **rtag**, and **update**.

WARNING: The commit and remove commands also have a -f option, but it has a different behavior for those commands. See see node ‘commit options' in the CVS manual, and see node ‘Removing files' in the CVS manual.

-k *kflag*

Alter the default processing of keywords. see node ‘Keyword substitution' in the CVS manual, for the meaning of *kflag*. Your *kflag* specification is *sticky* when you use it to create a private copy of a source file; that is, when you use this option with the **checkout** or **update** commands, **cvs** associates your selected *kflag* with the file, and continues to use it with future update commands on the same file until you specify otherwise.

The **-k** option is available with the **add**, **checkout**, **diff**, **rdiff**, **import** and **update** commands.

-l

Local; run only in current working directory, rather than recursing through subdirectories.

Available with the following commands: **annotate**, **checkout**, **commit**, **diff**, **edit**, **editors**, **export**, **log**, **rdiff**, **remove**, **rtag**, **status**, **tag**, **unedit**, **update**, **watch**, and **watchers**.

-m *message*

Use *message* as log information, instead of invoking an editor.

Available with the following commands: **add**, **commit** and **import**.

-n

Do not run any tag program. (A program can be specified to run in the modules database (see node ‘modules' in the CVS manual); this option bypasses it).

This is not the same as the cvs -n program option, which you can specify to the left of a cvs command!

Available with the **checkout**, **export**, and **rtag** commands.

-P

Prune empty directories. See see node ‘Removing directories' in the CVS manual.

-p

Pipe the files retrieved from the repository to standard output, rather than writing them in the current directory. Available with the **checkout** and **update** commands.

-R

Process directories recursively. This is on by default.

Available with the following commands: **annotate**, **checkout**, **commit**, **diff**, **edit**, **editors**, **export**, **rdiff**, **remove**, **rtag**, **status**, **tag**, **unedit**, **update**, **watch**, and **watchers**.

-r *tag*

Use the revision specified by the *tag* argument instead of the default *head* revision. As well as arbitrary tags defined with the **tag** or **rtag** command, two special tags are always available: **HEAD** refers to the most recent version available in the repository, and **BASE** refers to the revision you last checked out into the current working directory.

The tag specification is sticky when you use this with **checkout** or **update** to make your own copy of a file: **cvs** remembers the tag and continues to use it on future update commands, until you specify otherwise (for more information on sticky tags/dates, see node ‘Sticky tags' in the CVS manual).

The tag can be either a symbolic or numeric tag, as described in see node ‘Tags' in the CVS manual, or the name of a branch, as described in see node ‘Branching and merging' in the CVS manual. When a command expects a specific revision, the name of a branch is interpreted as the most recent revision on that branch.

Specifying the **-q** global option along with the **-r** command option is often useful, to suppress the warning messages when the **rcs** file does not contain the specified tag.

This is not the same as the overall cvs -r option, which you can specify to the left of a cvs command!

-r is available with the **annotate**, **checkout**, **commit**, **diff**, **history**, **export**, **rdiff**, **rtag**, and **update** commands.

-W

Specify file names that should be filtered. You can use this option repeatedly. The spec can be a file name pattern of the same type that you can specify in the **.cvswrappers** file. Available with the following commands: **import**, and **update**.

admin

Administration

- Requires: repository, working directory.

- Changes: repository.
- Synonym: rcs

This is the **cvs** interface to assorted administrative facilities. Some of them have questionable usefulness for **cvs** but exist for historical purposes. Some of the questionable options are likely to disappear in the future. This command *does* work recursively, so extreme care should be used.

On unix, if there is a group named **cvsadmin**, only members of that group can run **cvs admin** (except for the **cvs admin -k** command, which can be run by anybody). This group should exist on the server, or any system running the non-client/server **cvs**. To disallow **cvs admin** for all users, create a group with no users in it. On NT, the **cvsadmin** feature does not exist and all users can run **cvs admin**.

admin options

Some of these options have questionable usefulness for **cvs** but exist for historical purposes. Some even make it impossible to use **cvs** until you undo the effect!

-A*oldfile*

Might not work together with **cvs**. Append the access list of *oldfile* to the access list of the **rcs** file.

-a*logins*

Might not work together with **cvs**. Append the login names appearing in the comma-separated list *logins* to the access list of the **rcs** file.

-b[*rev***]**

Set the default branch to *rev*. In **cvs**, you normally do not manipulate default branches; sticky tags (see node ‘Sticky tags' in the CVS manual) are a better way to decide which branch you want to work on. There is one reason to run **cvs admin -b**: to revert to the vendor's version when using vendor branches (see node ‘Reverting local changes' in the CVS manual). There can be no space between **-b** and its argument.

-c*string*

Sets the comment leader to *string*. The comment leader is not used by current versions of **cvs** or **rcs** 5.7. Therefore, you can almost surely not worry about it. see node ‘Keyword substitution' in the CVS manual.

-e[*logins***]**

Might not work together with **cvs**. Erase the login names appearing in the comma-separated list *logins* from the access list of the RCS file. If *logins* is omitted, erase the entire access list. There can be no space between **-e** and its argument.

-I

Run interactively, even if the standard input is not a terminal. This option does not work with the client/server **cvs** and is likely to disappear in a future release of **cvs**.

-i

Useless with **cvs**. This creates and initializes a new **rcs** file, without depositing a revision. With **cvs**, add files with the **cvs add** command (see node ‘Adding files' in the CVS manual).

-k*subst*

Set the default keyword substitution to *subst*. see node ‘Keyword substitution' in the CVS manual. Giving an explicit **-k** option to **cvs update**, **cvs export**, or **cvs checkout** overrides this default.

-l[*rev***]**

Lock the revision with number *rev*. If a branch is given, lock the latest revision on that branch. If *rev* is omitted, lock the latest revision on the default branch. There can be no space between **-l** and its argument.

This can be used in conjunction with the **rcslock.pl** script in the **contrib** directory of the **cvs** source distribution to provide reserved checkouts (where only one user can be editing a given file at a time). See the comments in that file for details (and see the **README** file in that directory for disclaimers about the unsupported nature of contrib). According to comments in that file, locking must set to strict (which is the default).

-L

Set locking to strict. Strict locking means that the owner of an RCS file is not exempt from locking for checkin. For use with **cvs**, strict locking must be set; see the discussion under the **-l** option above.

-m*rev***:***msg*

Replace the log message of revision *rev* with *msg*.

-N*name***[:[***rev***]]**

Act like **-n**, except override any previous assignment of *name*. For use with magic branches, see see node ‘Magic branch numbers' in the CVS manual.

-n*name***[:[***rev***]]**

Associate the symbolic name *name* with the branch or revision *rev*. It is normally better to use **cvs tag** or **cvs rtag** instead. Delete the symbolic name if both **:** and *rev* are omitted; otherwise, print an error message if *name* is already associated with another number. If *rev* is symbolic, it is expanded before association. A *rev* consisting of a branch number followed by a **.** stands for the current latest revision in the branch. A **:** with an empty *rev* stands for the current latest revision on the default branch, normally the trunk. For example, **cvs admin -n***name***:** associates *name* with the current latest revision of all the RCS files; this contrasts with **cvs admin -n***name***:$** which associates *name* with the revision numbers extracted from keyword strings in the corresponding working files.

-o*range*

Deletes (*outdates*) the revisions given by *range*.

Note that this command can be quite dangerous unless you know *exactly* what you are doing (for example see the warnings below about how the *rev1*:*rev2* syntax is confusing).

If you are short on disc this option might help you. But think twice before using it—there is no way short of restoring the latest backup to undo this command! If you delete different revisions than you

planned, either due to carelessness or (heaven forbid) a **cvs** bug, there is no opportunity to correct the error before the revisions are deleted. It probably would be a good idea to experiment on a copy of the repository first.

Specify *range* in one of the following ways:

*rev1***::***rev2*

Collapse all revisions between rev1 and rev2, so that **cvs** only stores the differences associated with going from rev1 to rev2, not intermediate steps. For example, after **-o 1.3::1.5** one can retrieve revision 1.3, revision 1.5, or the differences to get from 1.3 to 1.5, but not the revision 1.4, or the differences between 1.3 and 1.4. Other examples: **-o 1.3::1.4** and **-o 1.3::1.3** have no effect, because there are no intermediate revisions to remove.

::*rev*

Collapse revisions between the beginning of the branch containing *rev* and *rev* itself. The branchpoint and *rev* are left intact. For example, **-o ::1.3.2.6** deletes revision 1.3.2.1, revision 1.3.2.5, and everything in between, but leaves 1.3 and 1.3.2.6 intact.

*rev***::**

Collapse revisions between *rev* and the end of the branch containing *rev*. Revision *rev* is left intact but the head revision is deleted.

rev

Delete the revision *rev*. For example, **-o 1.3** is equivalent to **-o 1.2::1.4**.

*rev1***:***rev2*

Delete the revisions from *rev1* to *rev2*, inclusive, on the same branch. One will not be able to retrieve *rev1* or *rev2* or any of the revisions in between. For example, the command **cvs admin -oR_1_01:R_1_02 .** is rarely useful. It means to delete revisions up to, and including, the tag R_1_02. But beware! If there are files that have not changed between R_1_02 and R_1_03 the file will have *the same* numerical revision number assigned to the tags R_1_02 and R_1_03. So not only will it be impossible to retrieve R_1_02; R_1_03 will also have to be restored from the tapes! In most cases you want to specify *rev1*::*rev2* instead.

:*rev*

Delete revisions from the beginning of the branch containing *rev* up to and including *rev*.

*rev***:**

Delete revisions from revision *rev*, including *rev* itself, to the end of the branch containing *rev*.

None of the revisions to be deleted may have branches or locks.

If any of the revisions to be deleted have symbolic names, and one specifies one of the **::** syntaxes, then **cvs** will give an error and not delete any revisions. If you really want to delete both the symbolic names and the revisions, first delete the symbolic names with **cvs tag -d**, then run **cvs admin -o**. If one specifies the non-**::** syntaxes, then **cvs** will delete the revisions but leave the symbolic names pointing to nonexistent revisions. This behavior is preserved for compatibility with previous versions of **cvs**, but because it isn't very useful, in the future it may change to be like the **::** case.

Due to the way **cvs** handles branches *rev* cannot be specified symbolically if it is a branch. see node ‘Magic branch numbers' in the CVS manual, for an explanation.

Make sure that no-one has checked out a copy of the revision you outdate. Strange things will happen if he starts to edit it and tries to check it back in. For this reason, this option is not a good way to take back a bogus commit; commit a new revision undoing the bogus change instead (see node ‘Merging

two revisions' in the CVS manual).

-q

Run quietly; do not print diagnostics.

-s*state***[:***rev***]**

Useful with **cvs**. Set the state attribute of the revision *rev* to *state*. If *rev* is a branch number, assume the latest revision on that branch. If *rev* is omitted, assume the latest revision on the default branch. Any identifier is acceptable for *state*. A useful set of states is **Exp** (for experimental), **Stab** (for stable), and **Rel** (for released). By default, the state of a new revision is set to **Exp** when it is created. The state is visible in the output from *cvs log* (see node 'log' in the CVS manual), and in the **Log** and **$State$** keywords (see node 'Keyword substitution' in the CVS manual). Note that **cvs** uses the **dead** state for its own purposes (see node 'Attic' in the CVS manual); to take a file to or from the **dead** state use commands like **cvs remove** and **cvs add** (see node 'Adding and removing' in the CVS manual), not **cvs admin -s**.

-t[*file***]**

Useful with **cvs**. Write descriptive text from the contents of the named *file* into the RCS file, deleting the existing text. The *file* pathname may not begin with **-**. The descriptive text can be seen in the output from **cvs log** (see node 'log' in the CVS manual). There can be no space between **-t** and its argument.

If *file* is omitted, obtain the text from standard input, terminated by end-of-file or by a line containing **.** by itself. Prompt for the text if interaction is possible; see **-I**.

-t-*string*

Similar to **-t***file*. Write descriptive text from the *string* into the **rcs** file, deleting the existing text. There can be no space between **-t** and its argument.

-U

Set locking to non-strict. Non-strict locking means that the owner of a file need not lock a revision for checkin. For use with **cvs**, strict locking must be set; see the discussion under the **-l** option above.

-u[*rev***]**

See the option **-l** above, for a discussion of using this option with **cvs**. Unlock the revision with number *rev*. If a branch is given, unlock the latest revision on that branch. If *rev* is omitted, remove the latest lock held by the caller. Normally, only the locker of a revision may unlock it; somebody else unlocking a revision breaks the lock. This causes the original locker to be sent a **commit** notification (see node 'Getting Notified' in the CVS manual). There can be no space between **-u** and its argument.

-V*n*

In previous versions of **cvs**, this option meant to write an **rcs** file which would be acceptable to **rcs** version *n*, but it is now obsolete and specifying it will produce an error.

-x*suffixes*

In previous versions of **cvs**, this was documented as a way of specifying the names of the **rcs** files. However, **cvs** has always required that the **rcs** files used by **cvs** end in **,v**, so this option has never done anything useful.

annotate

What revision modified each line of a file?

- Synopsis: annotate [options] files...
- Requires: repository.
- Changes: nothing.

For each file in *files*, print the head revision of the trunk, together with information on the last modification for each line.

annotate options

These standard options are supported by **annotate** (see node ‘Common options' in the CVS manual, for a complete description of them):

-l

Local directory only, no recursion.

-R

Process directories recursively.

-f

Use head revision if tag/date not found.

-F

Annotate binary files.

-r *revision*

Annotate file as of specified revision/tag.

-D *date*

Annotate file as of specified date.

annotate example

For example:

```
$ cvs annotate ssfile
Annotations for ssfile
***************
1.1         (mary     27-Mar-96): ssfile line 1
1.2         (joe      28-Mar-96): ssfile line 2
```

The file **ssfile** currently contains two lines. The **ssfile line 1** line was checked in by **mary** on March 27. Then, on March 28, **joe** added a line **ssfile line 2**, without modifying the **ssfile line 1** line. This report doesn't tell you anything about lines which have been deleted or replaced; you need to use **cvs diff** for that (see node ‘diff' in the CVS manual).

The options to **cvs annotate** are listed in see node ‘Invoking CVS' in the CVS manual, and can be used to select the files and revisions to annotate. The options are described in more detail there and in see node ‘Common options' in the CVS manual.

checkout

Check out sources for editing

- Synopsis: checkout [options] modules...
- Requires: repository.
- Changes: working directory.
- Synonyms: co, get

Create or update a working directory containing copies of the source files specified by *modules*. You must execute **checkout** before using most of the other **cvs** commands, since most of them operate on your working directory.

The *modules* are either symbolic names for some collection of source directories and files, or paths to directories or files in the repository. The symbolic names are defined in the **modules** file. see node ‘modules' in the CVS manual.

Depending on the modules you specify, **checkout** may recursively create directories and populate them with the appropriate source files. You can then edit these source files at any time (regardless of whether other software developers are editing their own copies of the sources); update them to include new changes applied by others to the source repository; or commit your work as a permanent change to the source repository.

Note that **checkout** is used to create directories. The top-level directory created is always added to the directory where **checkout** is invoked, and usually has the same name as the specified module. In the case of a module alias, the created sub-directory may have a different name, but you can be sure that it will be a sub-directory, and that **checkout** will show the relative path leading to each file as it is extracted into your private work area (unless you specify the **-Q** global option).

The files created by **checkout** are created read-write, unless the **-r** option to **cvs** (see node ‘Global options' in the CVS manual) is specified, the **CVSREAD** environment variable is specified (see node ‘Environment variables' in the CVS manual), or a watch is in effect for that file (see node ‘Watches' in the CVS manual).

Note that running **checkout** on a directory that was already built by a prior **checkout** is also permitted. This is similar to specifying the **-d** option to the **update** command in the sense that new directories that have been created in the repository will appear in your work area. However, **checkout** takes a module name whereas **update** takes a directory name. Also to use **checkout** this way it must be run from the top level directory (where you originally ran **checkout** from), so before you run **checkout** to update an existing directory, don't forget to change your directory to the top level directory.

For the output produced by the **checkout** command see see node ‘update output' in the CVS manual.

checkout options

These standard options are supported by **checkout** (see node ‘Common options' in the CVS manual, for a complete description of them):

-D *date*

Use the most recent revision no later than *date*. This option is sticky, and implies **-P**. See see node ‘Sticky tags' in the CVS manual, for more information on sticky tags/dates.

-f

Only useful with the **-D** *date* or **-r** *tag* flags. If no matching revision is found, retrieve the most recent revision (instead of ignoring the file).

-k *kflag*

Process keywords according to *kflag*. See see node ‘Keyword substitution' in the CVS manual. This option is sticky; future updates of this file in this working directory will use the same *kflag*. The **status** command can be viewed to see the sticky options. See see node ‘Invoking CVS' in the CVS manual, for more information on the **status** command.

-l

Local; run only in current working directory.

-n

Do not run any checkout program (as specified with the **-o** option in the modules file; see node ‘modules' in the CVS manual).

-P

Prune empty directories. See see node ‘Moving directories' in the CVS manual.

-p

Pipe files to the standard output.

-R

Checkout directories recursively. This option is on by default.

-r *tag*

Use revision *tag*. This option is sticky, and implies **-P**. See see node ‘Sticky tags' in the CVS manual, for more information on sticky tags/dates.

In addition to those, you can use these special command options with **checkout**:

-A

Reset any sticky tags, dates, or **-k** options. Does not reset sticky **-k** options on modified files. See see node ‘Sticky tags' in the CVS manual, for more information on sticky tags/dates.

-c

Copy the module file, sorted, to the standard output, instead of creating or modifying any files or directories in your working directory.

-d *dir*

Create a directory called *dir* for the working files, instead of using the module name. In general, using this flag is equivalent to using **mkdir** *dir***; cd** *dir* followed by the checkout command without the **-d** flag.

There is an important exception, however. It is very convenient when checking out a single item to have the output appear in a directory that doesn't contain empty intermediate directories. In this case *only*, **cvs** tries to ``shorten'' pathnames to avoid those empty directories.

For example, given a module **foo** that contains the file **bar.c**, the command **cvs co -d dir foo** will create directory **dir** and place **bar.c** inside. Similarly, given a module **bar** which has subdirectory **baz** wherein there is a file **quux.c**, the command **cvs co -d dir bar/baz** will create directory **dir** and place **quux.c** inside.

Using the **-N** flag will defeat this behavior. Given the same module definitions above, **cvs co -N -d dir foo** will create directories **dir/foo** and place **bar.c** inside, while **cvs co -N -d dir bar/baz** will create directories **dir/bar/baz** and place **quux.c** inside.

-j *tag*

With two **-j** options, merge changes from the revision specified with the first **-j** option to the revision specified with the second **j** option, into the working directory.

With one **-j** option, merge changes from the ancestor revision to the revision specified with the **-j** option, into the working directory. The ancestor revision is the common ancestor of the revision which the working directory is based on, and the revision specified in the **-j** option.

In addition, each -j option can contain an optional date specification which, when used with branches, can limit the chosen revision to one within a specific date. An optional date is specified by adding a colon (:) to the tag: **-j***Symbolic_Tag***:***Date_Specifier*.

see node ‘Branching and merging' in the CVS manual.

-N

Only useful together with **-d** *dir*. With this option, **cvs** will not ``shorten'' module paths in your working directory when you check out a single module. See the **-d** flag for examples and a discussion.

-s

Like **-c**, but include the status of all modules, and sort it by the status string. see node ‘modules' in the CVS manual, for info about the **-s** option that is used inside the modules file to set the module status.

checkout examples

Get a copy of the module **tc**:

```
$ cvs checkout tc
```

Get a copy of the module **tc** as it looked one day ago:

```
$ cvs checkout -D yesterday tc
```

commit

Check files into the repository

- Synopsis: commit [-lRf] [-m 'log_message' | -F file] [-r revision] [files...]
- Requires: working directory, repository.
- Changes: repository.
- Synonym: ci

Use **commit** when you want to incorporate changes from your working source files into the source repository.

If you don't specify particular files to commit, all of the files in your working current directory are examined. **commit** is careful to change in the repository only those files that you have really changed. By default (or if you explicitly specify the **-R** option), files in subdirectories are also examined and committed if they have changed; you can use the **-l** option to limit **commit** to the current directory only.

commit verifies that the selected files are up to date with the current revisions in the source repository; it will notify you, and exit without committing, if any of the specified files must be made current first with **update** (see node ‘update' in the CVS manual). **commit** does not call the **update** command for you, but rather leaves that for you to do when the time is right.

When all is well, an editor is invoked to allow you to enter a log message that will be written to one or more logging programs (see node ‘modules' in the CVS manual, and see node ‘loginfo' in the CVS manual) and placed in the **rcs** file inside the repository. This log message can be retrieved with the **log** command; see see node ‘log' in the CVS manual. You can specify the log message on the command line with the **-m** *message* option, and thus avoid the editor invocation, or use the **-F** *file* option to specify that the argument file contains the log message.

commit options

These standard options are supported by **commit** (see node ‘Common options' in the CVS manual, for a complete description of them):

-l

Local; run only in current working directory.

-R

Commit directories recursively. This is on by default.

-r *revision*

Commit to *revision*. *revision* must be either a branch, or a revision on the main trunk that is higher than any existing revision number (see node ‘Assigning revisions' in the CVS manual). You cannot commit to a specific revision on a branch.

commit also supports these options:

-F *file*

Read the log message from *file*, instead of invoking an editor.

-f

Note that this is not the standard behavior of the **-f** option as defined in see node ‘Common options' in the CVS manual.

Force **cvs** to commit a new revision even if you haven't made any changes to the file. If the current revision of *file* is 1.7, then the following two commands are equivalent:

```
$ cvs commit -f file
$ cvs commit -r 1.8 file
```

The **-f** option disables recursion (i.e., it implies **-l**). To force **cvs** to commit a new revision for all files in all subdirectories, you must use **-f -R**.

-m *message*

Use *message* as the log message, instead of invoking an editor.

commit examples

Committing to a branch

You can commit to a branch revision (one that has an even number of dots) with the **-r** option. To create a branch revision, use the **-b** option of the **rtag** or **tag** commands (see node ‘Branching and merging' in the CVS manual). Then, either **checkout** or **update** can be used to base your sources on the newly created branch. From that point on, all **commit** changes made within these working sources will be automatically added to a branch revision, thereby not disturbing main-line development in any way. For example, if you had to create a patch to the 1.2 version of the product, even though the 2.0 version is already under development, you might do:

```
$ cvs rtag -b -r FCS1_2 FCS1_2_Patch product_module
$ cvs checkout -r FCS1_2_Patch product_module
$ cd product_module
[[ hack away ]]
$ cvs commit
```

This works automatically since the **-r** option is sticky.

Creating the branch after editing

Say you have been working on some extremely experimental software, based on whatever revision you happened to checkout last week. If others in your group would like to work on this software with you, but without disturbing main-line development, you could commit your change to a new branch. Others can then checkout your experimental stuff and utilize the full benefit of **cvs** conflict resolution. The scenario might look like:

```
[[ hacked sources are present ]]
$ cvs tag -b EXPR1
$ cvs update -r EXPR1
$ cvs commit
```

The **update** command will make the **-r EXPR1** option sticky on all files. Note that your changes to the files will never be removed by the **update** command. The **commit** will automatically commit to the correct branch, because the **-r** is sticky. You could also do like this:

```
[[ hacked sources are present ]]
$ cvs tag -b EXPR1
$ cvs commit -r EXPR1
```

but then, only those files that were changed by you will have the **-r EXPR1** sticky flag. If you hack away, and commit without specifying the **-r EXPR1** flag, some files may accidentally end up on the main trunk.

To work with you on the experimental change, others would simply do

```
$ cvs checkout -r EXPR1 whatever_module
```

diff

Show differences between revisions

- Synopsis: diff [-lR] [-k kflag] [format_options] [[-r rev1 | -D date1] [-r rev2 | -D date2]] [files...]
- Requires: working directory, repository.
- Changes: nothing.

The **diff** command is used to compare different revisions of files. The default action is to compare your working files with the revisions they were based on, and report any differences that are found.

If any file names are given, only those files are compared. If any directories are given, all files under them will be compared.

The exit status for diff is different than for other **cvs** commands; for details see node ‘Exit status' in the CVS manual.

diff options

These standard options are supported by **diff** (see node ‘Common options' in the CVS manual, for a complete description of them):

-D *date*

Use the most recent revision no later than *date*. See **-r** for how this affects the comparison.

-k *kflag*

Process keywords according to *kflag*. See see node ‘Keyword substitution' in the CVS manual.

-l

Local; run only in current working directory.

-R

Examine directories recursively. This option is on by default.

-r *tag*

Compare with revision *tag*. Zero, one or two **-r** options can be present. With no **-r** option, the working file will be compared with the revision it was based on. With one **-r**, that revision will be compared to your current working file. With two **-r** options those two revisions will be compared (and your working file will not affect the outcome in any way).

One or both **-r** options can be replaced by a **-D** *date* option, described above.

The following options specify the format of the output. They have the same meaning as in GNU diff. Most options have two equivalent names, one of which is a single letter preceded by **-**, and the other of which is a long name preceded by **--**.

-*lines*

Show *lines* (an integer) lines of context. This option does not specify an output format by itself; it has no effect unless it is combined with **-c** or **-u**. This option is obsolete. For proper operation, **patch** typically needs at least two lines of context.

-a

Treat all files as text and compare them line-by-line, even if they do not seem to be text.

-b

Ignore trailing white space and consider all other sequences of one or more white space characters to be equivalent.

-B

Ignore changes that just insert or delete blank lines.

--binary

Read and write data in binary mode.

--brief

Report only whether the files differ, not the details of the differences.

-c

Use the context output format.

-C *lines*

--context[**=***lines*]

Use the context output format, showing *lines* (an integer) lines of context, or three if *lines* is not given. For proper operation, **patch** typically needs at least two lines of context.

--changed-group-format=*format*

Use *format* to output a line group containing differing lines from both files in if-then-else format. see node ‘Line group formats' in the CVS manual.

-d

Change the algorithm to perhaps find a smaller set of changes. This makes **diff** slower (sometimes much slower).

-e

--ed

Make output that is a valid **ed** script.

--expand-tabs

Expand tabs to spaces in the output, to preserve the alignment of tabs in the input files.

-f

Make output that looks vaguely like an **ed** script but has changes in the order they appear in the file.

-F *regexp*

In context and unified format, for each hunk of differences, show some of the last preceding line that matches *regexp*.

--forward-ed

Make output that looks vaguely like an **ed** script but has changes in the order they appear in the file.

-H

Use heuristics to speed handling of large files that have numerous scattered small changes.

--horizon-lines=*lines*

Do not discard the last *lines* lines of the common prefix and the first *lines* lines of the common suffix.

-i

Ignore changes in case; consider upper- and lower-case letters equivalent.

-I *regexp*

Ignore changes that just insert or delete lines that match *regexp*.

--ifdef=*name*

Make merged if-then-else output using *name*.

--ignore-all-space

Ignore white space when comparing lines.

--ignore-blank-lines

Ignore changes that just insert or delete blank lines.

--ignore-case

Ignore changes in case; consider upper- and lower-case to be the same.

--ignore-matching-lines=*regexp*

Ignore changes that just insert or delete lines that match *regexp*.

--ignore-space-change

Ignore trailing white space and consider all other sequences of one or more white space characters to be equivalent.

--initial-tab

Output a tab rather than a space before the text of a line in normal or context format. This causes the alignment of tabs in the line to look normal.

-L *label*

Use *label* instead of the file name in the context format and unified format headers.

--label=*label*

Use *label* instead of the file name in the context format and unified format headers.

--left-column

Print only the left column of two common lines in side by side format.

--line-format=*format*

Use *format* to output all input lines in if-then-else format. see node ‘Line formats' in the CVS manual.

--minimal

Change the algorithm to perhaps find a smaller set of changes. This makes **diff** slower (sometimes much slower).

-n

Output RCS-format diffs; like **-f** except that each command specifies the number of lines affected.

-N

--new-file

In directory comparison, if a file is found in only one directory, treat it as present but empty in the other directory.

--new-group-format=*format*

Use *format* to output a group of lines taken from just the second file in if-then-else format. see node ‘Line group formats' in the CVS manual.

--new-line-format=*format*

Use *format* to output a line taken from just the second file in if-then-else format. see node ‘Line formats' in the CVS manual.

--old-group-format=*format*

Use *format* to output a group of lines taken from just the first file in if-then-else format. see node ‘Line group formats' in the CVS manual.

--old-line-format=*format*

Use *format* to output a line taken from just the first file in if-then-else format. see node ‘Line formats' in the CVS manual.

-p

Show which C function each change is in.

--rcs

Output RCS-format diffs; like **-f** except that each command specifies the number of lines affected.

--report-identical-files

-s

Report when two files are the same.

--show-c-function

Show which C function each change is in.

--show-function-line=*regexp*

In context and unified format, for each hunk of differences, show some of the last preceding line that matches *regexp*.

--side-by-side

Use the side by side output format.

--speed-large-files

Use heuristics to speed handling of large files that have numerous scattered small changes.

--suppress-common-lines

Do not print common lines in side by side format.

-t

Expand tabs to spaces in the output, to preserve the alignment of tabs in the input files.

-T

Output a tab rather than a space before the text of a line in normal or context format. This causes the alignment of tabs in the line to look normal.

--text

Treat all files as text and compare them line-by-line, even if they do not appear to be text.

-u

Use the unified output format.

--unchanged-group-format=*format*

Use *format* to output a group of common lines taken from both files in if-then-else format. see node ‘Line group formats' in the CVS manual.

--unchanged-line-format=*format*

Use *format* to output a line common to both files in if-then-else format. see node ‘Line formats' in the CVS manual.

-U *lines*

--unified[=*lines*]

Use the unified output format, showing *lines* (an integer) lines of context, or three if *lines* is not given. For proper operation, **patch** typically needs at least two lines of context.

-w

Ignore white space when comparing lines.

-W *columns*

--width=*columns*

Use an output width of *columns* in side by side format.

-y

Use the side by side output format.

Line group formats

Line group formats let you specify formats suitable for many applications that allow if-then-else input, including programming languages and text formatting languages. A line group format specifies the output format for a contiguous group of similar lines.

For example, the following command compares the TeX file **myfile** with the original version from the repository, and outputs a merged file in which old regions are surrounded by **\begin{em}**-**\end{em}** lines, and new regions are surrounded by **\begin{bf}**-**\end{bf}** lines.

```
cvs diff \
  --old-group-format='\begin{em}
%<\end{em}
' \
  --new-group-format='\begin{bf}
%>\end{bf}
' \
  myfile
```

The following command is equivalent to the above example, but it is a little more verbose, because it spells out the default line group formats.

```
cvs diff \
  --old-group-format='\begin{em}
%<\end{em}
' \
  --new-group-format='\begin{bf}
```

```
%>\end{bf}
' \
  --unchanged-group-format='%=' \
  --changed-group-format='\begin{em}
%<\end{em}
\begin{bf}
%>\end{bf}
' \
  myfile
```

Here is a more advanced example, which outputs a diff listing with headers containing line numbers in a ``plain English'' style.

```
cvs diff \
  --unchanged-group-format='' \
  --old-group-format='-------- %dn line%(n=1?:s) deleted at %df:
%<' \
  --new-group-format='-------- %dN line%(N=1?:s) added after %de:
%>' \
  --changed-group-format='-------- %dn line%(n=1?:s) changed at %df:
%<-------- to:
%>' \
  myfile
```

To specify a line group format, use one of the options listed below. You can specify up to four line group formats, one for each kind of line group. You should quote *format*, because it typically contains shell metacharacters.

--old-group-format=*format*

These line groups are hunks containing only lines from the first file. The default old group format is the same as the changed group format if it is specified; otherwise it is a format that outputs the line group as-is.

--new-group-format=*format*

These line groups are hunks containing only lines from the second file. The default new group format is same as the changed group format if it is specified; otherwise it is a format that outputs the line group as-is.

--changed-group-format=*format*

These line groups are hunks containing lines from both files. The default changed group format is the concatenation of the old and new group formats.

--unchanged-group-format=*format*

These line groups contain lines common to both files. The default unchanged group format is a format that outputs the line group as-is.

In a line group format, ordinary characters represent themselves; conversion specifications start with **%** and have one of the following forms.

%<

stands for the lines from the first file, including the trailing newline. Each line is formatted according to the old line format (see node ‘Line formats' in the CVS manual).

%>

stands for the lines from the second file, including the trailing newline. Each line is formatted according to the new line format.

%=

stands for the lines common to both files, including the trailing newline. Each line is formatted according to the unchanged line format.

%%

stands for **%**.

%c'*C*'

where *C* is a single character, stands for *C*. *C* may not be a backslash or an apostrophe. For example, **%c':'** stands for a colon, even inside the then-part of an if-then-else format, which a colon would normally terminate.

%c'*O*'

where *O* is a string of 1, 2, or 3 octal digits, stands for the character with octal code *O*. For example, **%c'\0'** stands for a null character.

Fn

where *F* is a **printf** conversion specification and *n* is one of the following letters, stands for *n*'s value formatted with *F*.

e

The line number of the line just before the group in the old file.

f

The line number of the first line in the group in the old file; equals *e* + 1.

l

The line number of the last line in the group in the old file.

m

The line number of the line just after the group in the old file; equals *l* + 1.

n

The number of lines in the group in the old file; equals *l* - *f* + 1.

E, F, L, M, N

Likewise, for lines in the new file.

The **printf** conversion specification can be **%d**, **%o**, **%x**, or **%X**, specifying decimal, octal, lower case hexadecimal, or upper case hexadecimal output respectively. After the **%** the following options can appear in sequence: a **-** specifying left-justification; an integer specifying the minimum field width; and a period followed by an optional integer specifying the minimum number of digits. For example, **%5dN** prints the number of new lines in the group in a field of width 5 characters, using the **printf** format **"%5d"**.

(*A***=***B***?***T***:***E***)**

If *A* equals *B* then *T* else *E*. *A* and *B* are each either a decimal constant or a single letter interpreted as above. This format spec is equivalent to *T* if *A*'s value equals *B*'s; otherwise it is equivalent to *E*.

For example, **%(N=0?no:%dN) line%(N=1?:s)** is equivalent to **no lines** if *N* (the number of lines in the group in the new file) is 0, to **1 line** if *N* is 1, and to **%dN lines** otherwise.

Line formats

Line formats control how each line taken from an input file is output as part of a line group in if-then-else format.

For example, the following command outputs text with a one-column change indicator to the left of the text. The first column of output is **-** for deleted lines, **|** for added lines, and a space for unchanged lines. The formats contain newline characters where newlines are desired on output.

```
cvs diff \
  --old-line-format='-%l
' \
  --new-line-format='|%l
' \
  --unchanged-line-format=' %l
' \
  myfile
```

To specify a line format, use one of the following options. You should quote *format*, since it often contains shell metacharacters.

--old-line-format=*format*

formats lines just from the first file.

--new-line-format=*format*

formats lines just from the second file.

--unchanged-line-format=*format*

formats lines common to both files.

--line-format=*format*

formats all lines; in effect, it sets all three above options simultaneously.

In a line format, ordinary characters represent themselves; conversion specifications start with **%** and have one of the following forms.

%l

stands for the contents of the line, not counting its trailing newline (if any). This format ignores whether the line is incomplete.

%L

stands for the contents of the line, including its trailing newline (if any). If a line is incomplete, this format preserves its incompleteness.

%%

stands for **%**.

%c'*C***'**

where *C* is a single character, stands for *C*. *C* may not be a backslash or an apostrophe. For example, **%c':'** stands for a colon.

%c'*O***'**

where *O* is a string of 1, 2, or 3 octal digits, stands for the character with octal code *O*. For example, **%c'\0'** stands for a null character.

***F*n**

where *F* is a **printf** conversion specification, stands for the line number formatted with *F*. For example, **%.5dn** prints the line number using the **printf** format **"%.5d"**. see node ‘Line group formats' in the CVS manual, for more about printf conversion specifications.

The default line format is **%l** followed by a newline character.

If the input contains tab characters and it is important that they line up on output, you should ensure that **%l** or **%L** in a line format is just after a tab stop (e.g. by preceding **%l** or **%L** with a tab character), or you should use the **-t** or **--expand-tabs** option.

Taken together, the line and line group formats let you specify many different formats. For example, the following command uses a format similar to **diff**'s normal format. You can tailor this command to get fine control over **diff**'s output.

```
cvs diff \
  --old-line-format='< %l
' \
  --new-line-format='> %l
```

```
' \
  --old-group-format='%df%(f=l?:,%dl)d%dE
%<' \
  --new-group-format='%dea%dF%(F=L?:,%dL)
%>' \
  --changed-group-format='%df%(f=l?:,%dl)c%dF%(F=L?:,%dL)
%<—
%>' \
  --unchanged-group-format=" \
  myfile
```

diff examples

The following line produces a Unidiff (**-u** flag) between revision 1.14 and 1.19 of **backend.c**. Due to the **-kk** flag no keywords are substituted, so differences that only depend on keyword substitution are ignored.

```
$ cvs diff -kk -u -r 1.14 -r 1.19 backend.c
```

Suppose the experimental branch EXPR1 was based on a set of files tagged RELEASE_1_0. To see what has happened on that branch, the following can be used:

```
$ cvs diff -r RELEASE_1_0 -r EXPR1
```

A command like this can be used to produce a context diff between two releases:

```
$ cvs diff -c -r RELEASE_1_0 -r RELEASE_1_1 > diffs
```

If you are maintaining ChangeLogs, a command like the following just before you commit your changes may help you write the ChangeLog entry. All local modifications that have not yet been committed will be printed.

```
$ cvs diff -u | less
```

export

Export sources from CVS, similar to checkout

- Synopsis: export [-flNnR] [-r rev|-D date] [-k subst] [-d dir] module...
- Requires: repository.
- Changes: current directory.

  This command is a variant of **checkout**; use it when you want a copy of the source for module without the **cvs** administrative directories. For example, you might use **export** to prepare source for shipment off-site. This command requires that you specify a date or tag (with **-D** or **-r**), so that you can count on reproducing the source you ship to others (and thus it always prunes empty directories).

  One often would like to use **-kv** with **cvs export**. This causes any keywords to be expanded such that an

import done at some other site will not lose the keyword revision information. But be aware that doesn't handle an export containing binary files correctly. Also be aware that after having used **-kv**, one can no longer use the **ident** command (which is part of the **rcs** suite—see ident(1)) which looks for keyword strings. If you want to be able to use **ident** you must not use **-kv**.

export options

These standard options are supported by **export** (see node ‘Common options' in the CVS manual, for a complete description of them):

-D *date*

Use the most recent revision no later than *date*.

-f

If no matching revision is found, retrieve the most recent revision (instead of ignoring the file).

-l

Local; run only in current working directory.

-n

Do not run any checkout program.

-R

Export directories recursively. This is on by default.

-r *tag*

Use revision *tag*.

In addition, these options (that are common to **checkout** and **export**) are also supported:

-d *dir*

Create a directory called *dir* for the working files, instead of using the module name. see node ‘checkout options' in the CVS manual, for complete details on how **cvs** handles this flag.

-k *subst*

Set keyword expansion mode (see node ‘Substitution modes' in the CVS manual).

-N

Only useful together with **-d** *dir*. see node ‘checkout options' in the CVS manual, for complete details on how **cvs** handles this flag.

history

Show status of files and users

- Synopsis: history [-report] [-flags] [-options args] [files...]
- Requires: the file **$CVSROOT/CVSROOT/history**
- Changes: nothing.

cvs can keep a history file that tracks each use of the **checkout**, **commit**, **rtag**, **update**, and **release** commands. You can use **history** to display this information in various formats.

Logging must be enabled by creating the file **$CVSROOT/CVSROOT/history**.

history uses -f, -l, -n, and -p in ways that conflict with the normal use inside cvs (see node 'Common options' in the CVS manual).

history options

Several options (shown above as **-report**) control what kind of report is generated:

-c

Report on each time commit was used (i.e., each time the repository was modified).

-e

Everything (all record types). Equivalent to specifying **-x** with all record types. Of course, **-e** will also include record types which are added in a future version of **cvs**; if you are writing a script which can only handle certain record types, you'll want to specify **-x**.

-m *module*

Report on a particular module. (You can meaningfully use **-m** more than once on the command line.)

-o

Report on checked-out modules. This is the default report type.

-T

Report on all tags.

-x *type*

Extract a particular set of record types *type* from the **cvs** history. The types are indicated by single letters, which you may specify in combination.

Certain commands have a single record type:

F

release

O

checkout

E

export

T

rtag

One of five record types may result from an update:

C

A merge was necessary but collisions were detected (requiring manual merging).

G

A merge was necessary and it succeeded.

U

A working file was copied from the repository.

P

A working file was patched to match the repository.

W

The working copy of a file was deleted during update (because it was gone from the repository).

One of three record types results from commit:

A

A file was added for the first time.

M

A file was modified.

R

A file was removed.

The options shown as **-flags** constrain or expand the report without requiring option arguments:

-a

Show data for all users (the default is to show data only for the user executing **history**).

-l

Show last modification only.

-w

Show only the records for modifications done from the same working directory where **history** is executing.

The options shown as **-options** *args* constrain the report based on an argument:

-b *str*

Show data back to a record containing the string *str* in either the module name, the file name, or the repository path.

-D *date*

Show data since *date*. This is slightly different from the normal use of **-D** *date*, which selects the newest revision older than *date*.

-f *file*

Show data for a particular file (you can specify several **-f** options on the same command line). This is equivalent to specifying the file on the command line.

-n *module*

Show data for a particular module (you can specify several **-n** options on the same command line).

-p *repository*

Show data for a particular source repository (you can specify several **-p** options on the same command line).

-r *rev*

Show records referring to revisions since the revision or tag named *rev* appears in individual **rcs** files. Each **rcs** file is searched for the revision or tag.

-t *tag*

Show records since tag *tag* was last added to the history file. This differs from the **-r** flag above in that it reads only the history file, not the **rcs** files, and is much faster.

-u *name*

Show records for user *name*.

-z *timezone*

Show times in the selected records using the specified time zone instead of UTC.

import

Import sources into CVS, using vendor branches

- Synopsis: import [-options] repository vendortag releasetag...
- Requires: Repository, source distribution directory.
- Changes: repository.

Use **import** to incorporate an entire source distribution from an outside source (e.g., a source vendor) into your source repository directory. You can use this command both for initial creation of a repository, and for wholesale updates to the module from the outside source. see node ‘Tracking sources' in the CVS manual, for a discussion on this subject.

The *repository* argument gives a directory name (or a path to a directory) under the **cvs** root directory for repositories; if the directory did not exist, import creates it.

When you use import for updates to source that has been modified in your source repository (since a prior import), it will notify you of any files that conflict in the two branches of development; use **checkout -j** to reconcile the differences, as import instructs you to do.

If **cvs** decides a file should be ignored (see node ‘cvsignore' in the CVS manual), it does not import it and

prints **I** followed by the filename (see node ‘import output' in the CVS manual, for a complete description of the output).

If the file **$CVSROOT/CVSROOT/cvswrappers** exists, any file whose names match the specifications in that file will be treated as packages and the appropriate filtering will be performed on the file/directory before being imported. see node ‘Wrappers' in the CVS manual.

The outside source is saved in a first-level branch, by default 1.1.1. Updates are leaves of this branch; for example, files from the first imported collection of source will be revision 1.1.1.1, then files from the first imported update will be revision 1.1.1.2, and so on.

At least three arguments are required. *repository* is needed to identify the collection of source. *vendortag* is a tag for the entire branch (e.g., for 1.1.1). You must also specify at least one *releasetag* to uniquely identify the files at the leaves created each time you execute **import**. The *releasetag* should be new, not previously existing in the repository file, and uniquely identify the imported release,

Note that **import** does *not* change the directory in which you invoke it. In particular, it does not set up that directory as a **cvs** working directory; if you want to work with the sources import them first and then check them out into a different directory (see node ‘Getting the source' in the CVS manual).

import options

This standard option is supported by **import** (see node ‘Common options' in the CVS manual, for a complete description):

-m *message*

Use *message* as log information, instead of invoking an editor.

There are the following additional special options.

-b *branch*

See see node ‘Multiple vendor branches' in the CVS manual.

-k *subst*

Indicate the keyword expansion mode desired. This setting will apply to all files created during the import, but not to any files that previously existed in the repository. See see node ‘Substitution modes' in the CVS manual, for a list of valid **-k** settings.

-I *name*

Specify file names that should be ignored during import. You can use this option repeatedly. To avoid ignoring any files at all (even those ignored by default), specify `-I !'.

name can be a file name pattern of the same type that you can specify in the **.cvsignore** file. see node ‘cvsignore' in the CVS manual.

-W *spec*

Specify file names that should be filtered during import. You can use this option repeatedly.

spec can be a file name pattern of the same type that you can specify in the **.cvswrappers** file. see node ‘Wrappers' in the CVS manual.

import output

import keeps you informed of its progress by printing a line for each file, preceded by one character indicating the status of the file:

U *file*

The file already exists in the repository and has not been locally modified; a new revision has been created (if necessary).

N *file*

The file is a new file which has been added to the repository.

C *file*

The file already exists in the repository but has been locally modified; you will have to merge the changes.

I *file*

The file is being ignored (see node ‘cvsignore' in the CVS manual).

L *file*

The file is a symbolic link; **cvs import** ignores symbolic links. People periodically suggest that this behavior should be changed, but if there is a consensus on what it should be changed to, it doesn't seem to be apparent. (Various options in the **modules** file can be used to recreate symbolic links on checkout, update, etc.; see node ‘modules' in the CVS manual.)

import examples

See see node ‘Tracking sources' in the CVS manual, and see node ‘From files' in the CVS manual.

log

Print out log information for files

- Synopsis: log [options] [files...]
- Requires: repository, working directory.
- Changes: nothing.

Display log information for files. **log** used to call the **rcs** utility **rlog**. Although this is no longer true in the current sources, this history determines the format of the output and the options, which are not quite in the style of the other **cvs** commands.

The output includes the location of the **rcs** file, the *head* revision (the latest revision on the trunk), all symbolic names (tags) and some other things. For each revision, the revision number, the author, the number of lines added/deleted and the log message are printed. All times are displayed in Coordinated Universal Time (UTC). (Other parts of **cvs** print times in the local timezone).

log uses -R in a way that conflicts with the normal use inside cvs (see node ‘Common options' in the CVS manual).

log options

By default, **log** prints all information that is available. All other options restrict the output. Note that the revision selection options (**-d**, **-r**, **-s**, and **-w**) have no effect, other than possibly causing a search for files in Attic directories, when used in conjunction with the options that restrict the output to only **log** header fields

(**-b**, **-h**, **-R**, and **-t**) unless the **-S** option is also specified.

-b

Print information about the revisions on the default branch, normally the highest branch on the trunk.

-d *dates*

Print information about revisions with a checkin date/time in the range given by the semicolon-separated list of dates. The date formats accepted are those accepted by the **-D** option to many other **cvs** commands (see node ‘Common options' in the CVS manual). Dates can be combined into ranges as follows:

d1<d2

d2>d1

Select the revisions that were deposited between *d1* and *d2*.

<d

d>

Select all revisions dated *d* or earlier.

d<

>d

Select all revisions dated *d* or later.

d

Select the single, latest revision dated *d* or earlier.

The > or < characters may be followed by = to indicate an inclusive range rather than an exclusive one.

Note that the separator is a semicolon (;).

-h

Print only the name of the **rcs** file, name of the file in the working directory, head, default branch, access list, locks, symbolic names, and suffix.

-l

Local; run only in current working directory. (Default is to run recursively).

-N

Do not print the list of tags for this file. This option can be very useful when your site uses a lot of tags, so rather than "more"'ing over 3 pages of tag information, the log information is presented without tags at all.

-R

Print only the name of the **rcs** file.

-r*revisions*

Print information about revisions given in the comma-separated list *revisions* of revisions and ranges. The following table explains the available range formats:

*rev1***:***rev2*

Revisions *rev1* to *rev2* (which must be on the same branch).

*rev1***::***rev2*

The same, but excluding *rev1*.

:*rev*

::*rev*

Revisions from the beginning of the branch up to and including *rev*.

*rev***:**

Revisions starting with *rev* to the end of the branch containing *rev*.

*rev***::**

Revisions starting just after *rev* to the end of the branch containing *rev*.

branch

An argument that is a branch means all revisions on that branch.

*branch1***:***branch2*

*branch1***::***branch2*

A range of branches means all revisions on the branches in that range.

*branch***.**

The latest revision in *branch*.

A bare **-r** with no revisions means the latest revision on the default branch, normally the trunk. There can be no space between the **-r** option and its argument.

-S

Suppress the header if no revisions are selected.

-s *states*

Print information about revisions whose state attributes match one of the states given in the comma-separated list *states*. Individual states may be any text string, though **cvs** commonly only uses two states, **Exp** and **dead**. See see node ‘admin options' in the CVS manual for more information.

-t

Print the same as **-h**, plus the descriptive text.

-w*logins*

Print information about revisions checked in by users with login names appearing in the comma-separated list *logins*. If *logins* is omitted, the user's login is assumed. There can be no space between the **-w** option and its argument.

log prints the intersection of the revisions selected with the options **-d**, **-s**, and **-w**, intersected with the union of the revisions selected by **-b** and **-r**.

log examples

Contributed examples are gratefully accepted.

rdiff

'patch' format diffs between releases

- rdiff [-flags] [-V vn] [-r t|-D d [-r t2|-D d2]] modules...
- Requires: repository.
- Changes: nothing.
- Synonym: patch

Builds a Larry Wall format patch(1) file between two releases, that can be fed directly into the **patch** program to bring an old release up-to-date with the new release. (This is one of the few **cvs** commands that operates directly from the repository, and doesn't require a prior checkout.) The diff output is sent to the standard output device.

You can specify (using the standard **-r** and **-D** options) any combination of one or two revisions or dates. If only one revision or date is specified, the patch file reflects differences between that revision or date and the current head revisions in the **rcs** file.

Note that if the software release affected is contained in more than one directory, then it may be necessary to specify the **-p** option to the **patch** command when patching the old sources, so that **patch** is able to find the files that are located in other directories.

rdiff options

These standard options are supported by **rdiff** (see node ‘Common options' in the CVS manual, for a complete description of them):

-D *date*

Use the most recent revision no later than *date*.

-f

If no matching revision is found, retrieve the most recent revision (instead of ignoring the file).

-k *kflag*

Process keywords according to *kflag*. See see node ‘Keyword substitution' in the CVS manual.

-l

Local; don't descend subdirectories.

-R

Examine directories recursively. This option is on by default.

-r *tag*

Use revision *tag*.

In addition to the above, these options are available:

-c

Use the context diff format. This is the default format.

-s

Create a summary change report instead of a patch. The summary includes information about files that were changed or added between the releases. It is sent to the standard output device. This is useful for finding out, for example, which files have changed between two dates or revisions.

-t

A diff of the top two revisions is sent to the standard output device. This is most useful for seeing what the last change to a file was.

-u

Use the unidiff format for the context diffs. Remember that old versions of the **patch** program can't handle the unidiff format, so if you plan to post this patch to the net you should probably not use **-u**.

-V *vn*

Expand keywords according to the rules current in **rcs** version *vn* (the expansion format changed with **rcs** version 5). Note that this option is no longer accepted. **cvs** will always expand keywords the way that **rcs** version 5 does.

rdiff examples

Suppose you receive mail from foo@example.net asking for an update from release 1.2 to 1.4 of the tc compiler. You have no such patches on hand, but with **cvs** that can easily be fixed with a command such as this:

```
$ cvs rdiff -c -r FOO1_2 -r FOO1_4 tc | \
$$ Mail -s 'The patches you asked for' foo@example.net
```

Suppose you have made release 1.3, and forked a branch called **R_1_3fix** for bug fixes. **R_1_3_1** corresponds to release 1.3.1, which was made some time ago. Now, you want to see how much development has been done on the branch. This command can be used:

```
$ cvs patch -s -r R_1_3_1 -r R_1_3fix module-name
cvs rdiff: Diffing module-name
File ChangeLog,v changed from revision 1.52.2.5 to 1.52.2.6
File foo.c,v changed from revision 1.52.2.3 to 1.52.2.4
File bar.h,v changed from revision 1.29.2.1 to 1.2
```

release

Indicate that a Module is no longer in use

- release [-d] directories...
- Requires: Working directory.
- Changes: Working directory, history log.

This command is meant to safely cancel the effect of **cvs checkout**. Since **cvs** doesn't lock files, it isn't strictly necessary to use this command. You can always simply delete your working directory, if you like; but you risk losing changes you may have forgotten, and you leave no trace in the **cvs** history file (see node ‘history file' in the CVS manual) that you've abandoned your checkout.

Use **cvs release** to avoid these problems. This command checks that no uncommitted changes are present; that you are executing it from immediately above a **cvs** working directory; and that the repository recorded for your files is the same as the repository defined in the module database.

If all these conditions are true, **cvs release** leaves a record of its execution (attesting to your intentionally abandoning your checkout) in the **cvs** history log.

release options

The **release** command supports one command option:

-d

Delete your working copy of the file if the release succeeds. If this flag is not given your files will remain in your working directory.

WARNING: The release command deletes all directories and files recursively. This has the very serious side-effect that any directory that you have created inside your checked-out sources, and not added to the repository (using the add command; see node ‘Adding files' in the CVS manual) will be silently deleted—even if it is non-empty!

release output

Before **release** releases your sources it will print a one-line message for any file that is not up-to-date.

U *file*

P *file*

There exists a newer revision of this file in the repository, and you have not modified your local copy of the file (**U** and **P** mean the same thing).

A *file*

The file has been added to your private copy of the sources, but has not yet been committed to the repository. If you delete your copy of the sources this file will be lost.

R *file*

The file has been removed from your private copy of the sources, but has not yet been removed from the repository, since you have not yet committed the removal. see node ‘commit' in the CVS manual.

M *file*

The file is modified in your working directory. There might also be a newer revision inside the repository.

? *file*

file is in your working directory, but does not correspond to anything in the source repository, and is not in the list of files for **cvs** to ignore (see the description of the **-I** option, and see node ‘cvsignore' in the CVS manual). If you remove your working sources, this file will be lost.

release examples

Release the **tc** directory, and delete your local working copy of the files.

```
$ cd ..        # You must stand immediately above the
               # sources when you issue cvs release.
$ cvs release -d tc
You have [0] altered files in this repository.
Are you sure you want to release (and delete) directory `tc': y
$
```

update

Bring work tree in sync with repository

- update [-ACdflPpR] [-I name] [-j rev [-j rev]] [-k kflag] [-r tag|-D date] [-W spec] files...
- Requires: repository, working directory.
- Changes: working directory.

After you've run checkout to create your private copy of source from the common repository, other developers will continue changing the central source. From time to time, when it is convenient in your development process, you can use the **update** command from within your working directory to reconcile your work with any revisions applied to the source repository since your last checkout or update.

update options

These standard options are available with **update** (see node ‘Common options' in the CVS manual, for a complete description of them):

-D date

Use the most recent revision no later than *date*. This option is sticky, and implies **-P**. See see node ‘Sticky tags' in the CVS manual, for more information on sticky tags/dates.

-f

Only useful with the **-D** *date* or **-r** *tag* flags. If no matching revision is found, retrieve the most recent revision (instead of ignoring the file).

-k *kflag*

Process keywords according to *kflag*. See see node ‘Keyword substitution' in the CVS manual. This option is sticky; future updates of this file in this working directory will use the same *kflag*. The **status** command can be viewed to see the sticky options. See see node ‘Invoking CVS' in the CVS manual, for more information on the **status** command.

-l

Local; run only in current working directory. see node ‘Recursive behavior' in the CVS manual.

-P

Prune empty directories. See see node ‘Moving directories' in the CVS manual.

-p

Pipe files to the standard output.

-R

Update directories recursively (default). see node ‘Recursive behavior' in the CVS manual.

-r rev

Retrieve revision/tag *rev*. This option is sticky, and implies **-P**. See see node ‘Sticky tags' in the CVS manual, for more information on sticky tags/dates.

These special options are also available with **update**.

-A

Reset any sticky tags, dates, or **-k** options. Does not reset sticky **-k** options on modified files. See see node ‘Sticky tags' in the CVS manual, for more information on sticky tags/dates.

-C

Overwrite locally modified files with clean copies from the repository (the modified file is saved in **.#***file***.***revision*, however).

-d

Create any directories that exist in the repository if they're missing from the working directory. Normally, **update** acts only on directories and files that were already enrolled in your working directory.

This is useful for updating directories that were created in the repository since the initial checkout; but it has an unfortunate side effect. If you deliberately avoided certain directories in the repository when you created your working directory (either through use of a module name or by listing explicitly the files and directories you wanted on the command line), then updating with **-d** will create those directories, which may not be what you want.

-I *name*

Ignore files whose names match *name* (in your working directory) during the update. You can specify **-I** more than once on the command line to specify several files to ignore. Use **-I !** to avoid ignoring any files at all. see node ‘cvsignore' in the CVS manual, for other ways to make **cvs** ignore some files.

-W*spec*

Specify file names that should be filtered during update. You can use this option repeatedly.

spec can be a file name pattern of the same type that you can specify in the **.cvswrappers** file. see node ‘Wrappers' in the CVS manual.

-j*revision*

With two **-j** options, merge changes from the revision specified with the first **-j** option to the revision specified with the second **j** option, into the working directory.

With one **-j** option, merge changes from the ancestor revision to the revision specified with the **-j** option, into the working directory. The ancestor revision is the common ancestor of the revision which the working directory is based on, and the revision specified in the **-j** option.

Note that using a single **-j** *tagname* option rather than **-j** *branchname* to merge changes from a branch will often not remove files which were removed on the branch. see node ‘Merging adds and removals' in the CVS manual, for more.

In addition, each **-j** option can contain an optional date specification which, when used with branches, can limit the chosen revision to one within a specific date. An optional date is specified by adding a colon (:) to the tag: **-j***Symbolic_Tag***:***Date_Specifier*.

see node ‘Branching and merging' in the CVS manual.

update output

update and **checkout** keep you informed of their progress by printing a line for each file, preceded by one character indicating the status of the file:

U *file*

The file was brought up to date with respect to the repository. This is done for any file that exists in the repository but not in your working directory, and for files that you haven't changed but are not the most recent versions available in the repository.

P *file*

Like **U**, but the **cvs** server sends a patch instead of an entire file. This accomplishes the same thing as **U** using less bandwidth.

A *file*

The file has been added to your private copy of the sources, and will be added to the source repository when you run **commit** on the file. This is a reminder to you that the file needs to be committed.

R *file*

The file has been removed from your private copy of the sources, and will be removed from the source repository when you run **commit** on the file. This is a reminder to you that the file needs to be committed.

M *file*

The file is modified in your working directory.

M can indicate one of two states for a file you're working on: either there were no modifications to the same file in the repository, so that your file remains as you last saw it; or there were modifications in the repository as well as in your copy, but they were merged successfully, without conflict, in your working directory.

cvs will print some messages if it merges your work, and a backup copy of your working file (as it looked before you ran **update**) will be made. The exact name of that file is printed while **update** runs.

C *file*

A conflict was detected while trying to merge your changes to *file* with changes from the source repository. *file* (the copy in your working directory) is now the result of attempting to merge the two revisions; an unmodified copy of your file is also in your working directory, with the name ***.#file.revision*** where *revision* is the revision that your modified file started from. Resolve the conflict as described in see node ‘Conflicts example' in the CVS manual. (Note that some systems automatically purge files that begin with **.#** if they have not been accessed for a few days. If you intend to keep a copy of your original file, it is a very good idea to rename it.) Under **vms**, the file name starts with __ rather than **.#**.

? *file*

file is in your working directory, but does not correspond to anything in the source repository, and is not in the list of files for **cvs** to ignore (see the description of the **-I** option, and see node ‘cvsignore' in the CVS manual).

AUTHORS

Dick Grune

Original author of the **cvs** shell script version posted to **comp.sources.unix** in the volume6 release of December, 1986. Credited with much of the **cvs** conflict resolution algorithms.

Brian Berliner

Coder and designer of the **cvs** program itself in April, 1989, based on the original work done by Dick.

Jeff Polk

Helped Brian with the design of the **cvs** module and vendor branch support and author of the **checkin**(**1**) shell script (the ancestor of **cvs import**).

Larry Jones, Derek R. Price, and Mark D. Baushke

Have helped maintain **cvs** for many years.

And many others too numerous to mention here.

SEE ALSO

The most comprehensive manual for CVS is Version Management with CVS by Per Cederqvist et al. Depending on your system, you may be able to get it with the **info CVS** command or it may be available as cvs.pdf (Portable Document Format), cvs.ps (PostScript), cvs.texinfo (Texinfo source), or cvs.html.

For CVS updates, more information on documentation, software related to CVS, development of CVS, and more, see:

http://cvs.nongnu.org

ci(**1**), **co**(**1**), **cvs**(**5**), **cvsbug**(**8**), **diff**(**1**), **grep**(**1**), **patch**(**1**), **rcs**(**1**), **rcsdiff**(**1**), **rcsmerge**(**1**), **rlog**(**1**).

NAME

daicctl — ISDN control, test and statistics utility

SYNOPSIS

under construction

DESCRIPTION

under construction.

Call it with -? to get minimal help.

SEE ALSO

daic(4)

HISTORY

The **daicctl** utility is not yet published.

AUTHORS

The **daicctl** utility was written by Martin Husemann.

NAME

date — display or set date and time

SYNOPSIS

date [**-ajnu**] [**-d** *date*] [**-r** *seconds*] [**+***format*] [[[[[[*CC*]*yy*]*mm*]*dd*]*HH*]*MM*[.*SS*]]

DESCRIPTION

date displays the current date and time when invoked without arguments. Providing arguments will format the date and time in a user-defined way or set the date. Only the superuser may set the date.

The options are as follows:

-a Use adjtime(2) to change the local system time slowly, maintaining it as a monotonically increasing function. **-a** implies **-n**.

-d *date*
Parse the provided human-described date and time and display the result without actually changing the system clock.

-j Parse the provided canonical representation of date and time (described below) and display the result without actually changing the system clock.

-n The utility timed(8) is used to synchronize the clocks on groups of machines. By default, if timed is running, **date** will set the time on all of the machines in the local group. The **-n** option stops **date** from setting the time for other than the current machine.

-r Print out the date and time that is *seconds* from the Epoch.

-u Display or set the date in UTC (universal) time.

An operand with a leading plus (**+**) sign signals a user-defined format string which specifies the format in which to display the date and time. The format string may contain any of the conversion specifications described in the strftime(3) manual page, as well as any arbitrary text. A <newline> character is always output after the characters specified by the format string. The format string for the default display is:

```
%a %b %e %H:%M:%S %Z %Y
```

If an operand does not have a leading plus sign, it is interpreted as a value for setting the system's notion of the current date and time. The canonical representation for setting the date and time is:

CC The first two digits of the year (the century).
yy The second two digits of the year. If *yy* is specified, but *CC* is not, a value for *yy* between 69 and 99 results in a *CC* value of 19. Otherwise, a *CC* value of 20 is used.
mm The month of the year, from 01 to 12.
dd The day of the month, from 01 to 31.
HH The hour of the day, from 00 to 23.
MM The minute of the hour, from 00 to 59.
SS The second of the minute, from 00 to 61.

Everything but the minutes is optional.

Time changes for Daylight Saving and Standard time and leap seconds and years are handled automatically.

ENVIRONMENT

The following environment variables affect the execution of **date**:

TZ The timezone to use when displaying dates. See environ(7) for more information.

FILES

`/etc/localtime`	Symlink pointing to system's default timezone information file in `/usr/share/zoneinfo` directory.
`/var/log/wtmp`	A record of date resets and time changes.
`/var/log/messages`	A record of the user setting the time.

EXAMPLES

The command:

```
date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'
```

will display:

```
DATE: 11/21/87
TIME: 13:36:16
```

The command:

```
date 8506131627
```

sets the date to "`June 13, 1985, 4:27 PM`".

The command:

```
date 1432
```

sets the time to `2:32 PM`, without modifying the date.

DIAGNOSTICS

Exit status is 0 on success, 1 if unable to set the date, and 2 if able to set the local date, but unable to set it globally.

Occasionally, when `timed`(8) synchronizes the time on many hosts, the setting of a new time value may require more than a few seconds. On these occasions, **`date`** prints: `Network time being set`. The message `Communication error with timed` occurs when the communication between **`date`** and timed fails.

SEE ALSO

`adjtime`(2), `gettimeofday`(2), `settimeofday`(2), `parsedate`(3), `strftime`(3), `utmp`(5), `timed`(8)

R. Gusella and S. Zatti, *TSP: The Time Synchronization Protocol for UNIX 4.3BSD*.

STANDARDS

The **`date`** utility is expected to be compatible with IEEE Std 1003.2 ("POSIX.2").

NAME

db — manipulate db(3)'s btree(3) and hash(3) databases

SYNOPSIS

db [**-KiNqV**] [**-E** *endian*] [**-f** *infile*] [**-O** *outsep*] [**-S** *visitem*] [**-T** *visspec*] [**-X** *extravis*] *type dbfile* [*key* [...]]
db -d [**-iNq**] [**-E** *endian*] [**-f** *infile*] [**-U** *unvisitem*] *type dbfile* [*key* [...]]
db -w [**-CDiNqR**] [**-E** *endian*] [**-F** *insep*] [**-f** *infile*] [**-m** *mode*] [**-U** *unvisitem*] *type dbfile* [*key value* [...]]

DESCRIPTION

db allows manipulation of btree(3) and hash(3) (db(3)) databases.

db has three modes of operation to perform upon *dbfile*:

read — Displays the given *key*s, and keys described in *infile*. If no keys and no *infile* is specified, the entire database is displayed. This is the default mode of operation.

delete — Enabled with **-d**. Deletes the given *key*s, and keys described in *infile*.

write — Enabled with **-w**. Writes the given *key*s and *value*s, and keys and values described in *infile* (in the latter case, entries are separated by *insep*).

There are two mandatory arguments: *type* is the database type; either '**btree**' or '**hash**', and *dbfile* is the database file to manipulate.

Options valid for all modes are:

-E *endian*
Set the endianness of the database. *endian* may be one of:
B
Big endian
H
Host endian
L
Little endian
Defaults to 'H' (host endian).

-f *infile*
Contains a list of keys (for read and delete), or *insep* separated keys and values (for write) to be used as arguments to the given mode. If *infile* is '-', stdin is used.

-i Keys are converted to lower case before manipulation.

-N Do not include the NUL byte at the end of the key or value.

-q Quiet operation. In read mode, missing keys are not considered to be an error. In delete (**-d**) and write (**-w**) modes, the result of various operations is suppressed.

Read mode specific options are:

-K Display key.

-O *outsep*
Field separator string between key and value. Defaults to a single tab ('\t').

-S *visitem*
Specify items to strvis(3) encode. The *visitem* option-argument is a character specifying if the key (*k*), the value (*v*) or both (*b*) should be encoded.

-T *visspec*
Control how the items specified by the **-S** option are encoded. The *visspec* option-argument is a string specifying `strvis`(3) options. The string consists of the specification characters **b**, **c**, **o**, **s**, **t**, and **w**. See `vis`(1)'s corresponding options for the meaning of these characters.

-V Display value.

-X *extravis*
When encoding items with **-S** option also encode characters in *extravis*, per `svis`(3).

(If neither of **-K** or **-V** is given, both options are enabled.)

Write mode specific options are:

-C Create new database, and truncate existing databases.

-D Allow duplicate entries. (Requires **-R** to be useful.)

-F *insep* Input field separator string between key and value used when parsing *infile*. Defaults to a single space (' ').

-m *mode* Octal mode of created database. Defaults to '0644'.

-R Overwrite existing entries. If not specified, writing to an existing entry raises an error.

Write and delete mode specific options are:

-U *unvisitem*
Specify items to `strunvis`(3) decode. The *unvisitem* option-argument is a character specifying if the key (*k*), the value (*v*) or both (*b*) should be decoded.

SEE ALSO

`vis`(1), `btree`(3), `db`(3), `hash`(3), `strunvis`(3), `strvis`(3), `svis`(3)

HISTORY

The **db** command appeared in NetBSD 2.0.

AUTHORS

Luke Mewburn ⟨lukem@NetBSD.org⟩.

NAME

dc − an arbitrary precision calculator

SYNOPSIS

dc [-V] [--version] [-h] [--help]
[-e scriptexpression] [--expression=scriptexpression]
[-f scriptfile] [--file=scriptfile]
[file ...]

DESCRIPTION

Dc is a reverse-polish desk calculator which supports unlimited precision arithmetic. It also allows you to define and call macros. Normally *dc* reads from the standard input; if any command arguments are given to it, they are filenames, and *dc* reads and executes the contents of the files before reading from standard input. All normal output is to standard output; all error output is to standard error.

A reverse-polish calculator stores numbers on a stack. Entering a number pushes it on the stack. Arithmetic operations pop arguments off the stack and push the results.

To enter a number in *dc*, type the digits with an optional decimal point. Exponential notation is not supported. To enter a negative number, begin the number with "_". "-" cannot be used for this, as it is a binary operator for subtraction instead. To enter two numbers in succession, separate them with spaces or newlines. These have no meaning as commands.

OPTIONS

Dc may be invoked with the following command-line options:

-V

--version
Print out the version of *dc* that is being run and a copyright notice, then exit.

-h

--help Print a usage message briefly summarizing these command-line options and the bug-reporting address, then exit.

-e *script*

--expression=*script*
Add the commands in *script* to the set of commands to be run while processing the input.

-f *script-file*

--file=*script-file*
Add the commands contained in the file *script-file* to the set of commands to be run while processing the input.

If any command-line parameters remain after processing the above, these parameters are interpreted as the names of input files to be processed. A file name of - refers to the standard input stream. The standard input will processed if no file names are specified.

Printing Commands

p Prints the value on the top of the stack, without altering the stack. A newline is printed after the value.

n Prints the value on the top of the stack, popping it off, and does not print a newline after.

P Pops off the value on top of the stack. If it it a string, it is simply printed without a trailing newline. Otherwise it is a number, and the integer portion of its absolute value is printed out as a "base (UCHAR_MAX+1)" byte stream. Assuming that (UCHAR_MAX+1) is 256 (as it is on most machines with 8-bit bytes), the sequence **KSK 0k1/ [_1*]sx d0>x [256˜aPd0<x]dsxx sxLKk** could also accomplish this function, except for the side-effect of clobbering the x register.

f Prints the entire contents of the stack without altering anything. This is a good command to use if you are lost or want to figure out what the effect of some command has been.

Arithmetic

+ Pops two values off the stack, adds them, and pushes the result. The precision of the result is determined only by the values of the arguments, and is enough to be exact.

- Pops two values, subtracts the first one popped from the second one popped, and pushes the result.

***** Pops two values, multiplies them, and pushes the result. The number of fraction digits in the result depends on the current precision value and the number of fraction digits in the two arguments.

/ Pops two values, divides the second one popped from the first one popped, and pushes the result. The number of fraction digits is specified by the precision value.

% Pops two values, computes the remainder of the division that the **/** command would do, and pushes that. The value computed is the same as that computed by the sequence **Sd dld/ Ld*-** .

~ Pops two values, divides the second one popped from the first one popped. The quotient is pushed first, and the remainder is pushed next. The number of fraction digits used in the division is specified by the precision value. (The sequence **SdSn lnld/ LnLd%** could also accomplish this function, with slightly different error checking.)

^ Pops two values and exponentiates, using the first value popped as the exponent and the second popped as the base. The fraction part of the exponent is ignored. The precision value specifies the number of fraction digits in the result.

| Pops three values and computes a modular exponentiation. The first value popped is used as the reduction modulus; this value must be a non-zero number, and should be an integer. The second popped is used as the exponent; this value must be a non-negative number, and any fractional part of this exponent will be ignored. The third value popped is the base which gets exponentiated, which should be an integer. For small integers this is like the sequence **Sm^Lm%**, but, unlike **^**, this command will work with arbitrarily large exponents.

v Pops one value, computes its square root, and pushes that. The precision value specifies the number of fraction digits in the result.

Most arithmetic operations are affected by the ‘‘precision value’’, which you can set with the **k** command. The default precision value is zero, which means that all arithmetic except for addition and subtraction produces integer results.

Stack Control

c Clears the stack, rendering it empty.

d Duplicates the value on the top of the stack, pushing another copy of it. Thus, ‘‘4d*p’’ computes 4 squared and prints it.

r Reverses the order of (swaps) the top two values on the stack.

Registers

Dc provides at least 256 memory registers, each named by a single character. You can store a number or a string in a register and retrieve it later.

s*r* Pop the value off the top of the stack and store it into register *r*.

l*r* Copy the value in register *r* and push it onto the stack. This does not alter the contents of *r*.

Each register also contains its own stack. The current register value is the top of the register’s stack.

S*r* Pop the value off the top of the (main) stack and push it onto the stack of register *r*. The previous value of the register becomes inaccessible.

L*r* Pop the value off the top of register *r*’s stack and push it onto the main stack. The previous value in register *r*’s stack, if any, is now accessible via the **l***r* command.

Parameters

Dc has three parameters that control its operation: the precision, the input radix, and the output radix. The precision specifies the number of fraction digits to keep in the result of most arithmetic operations. The input radix controls the interpretation of numbers typed in; all numbers typed in use this radix. The output

radix is used for printing numbers.

The input and output radices are separate parameters; you can make them unequal, which can be useful or confusing. The input radix must be between 2 and 16 inclusive. The output radix must be at least 2. The precision must be zero or greater. The precision is always measured in decimal digits, regardless of the current input or output radix.

i Pops the value off the top of the stack and uses it to set the input radix.

o Pops the value off the top of the stack and uses it to set the output radix.

k Pops the value off the top of the stack and uses it to set the precision.

I Pushes the current input radix on the stack.

O Pushes the current output radix on the stack.

K Pushes the current precision on the stack.

Strings

Dc can operate on strings as well as on numbers. The only things you can do with strings are print them and execute them as macros (which means that the contents of the string are processed as *dc* commands). All registers and the stack can hold strings, and *dc* always knows whether any given object is a string or a number. Some commands such as arithmetic operations demand numbers as arguments and print errors if given strings. Other commands can accept either a number or a string; for example, the **p** command can accept either and prints the object according to its type.

[*characters***]**
Makes a string containing *characters* (contained between balanced **[** and **]** characters), and pushes it on the stack. For example, **[foo]P** prints the characters **foo** (with no newline).

a The top-of-stack is popped. If it was a number, then the low-order byte of this number is converted into a string and pushed onto the stack. Otherwise the top-of-stack was a string, and the first character of that string is pushed back.

x Pops a value off the stack and executes it as a macro. Normally it should be a string; if it is a number, it is simply pushed back onto the stack. For example, **[1p]x** executes the macro **1p** which pushes **1** on the stack and prints **1** on a separate line.

Macros are most often stored in registers; **[1p]sa** stores a macro to print **1** into register **a**, and **lax** invokes this macro.

>r Pops two values off the stack and compares them assuming they are numbers, executing the contents of register *r* as a macro if the original top-of-stack is greater. Thus, **1 2>a** will invoke register **a**'s contents and **2 1>a** will not.

!>*r* Similar but invokes the macro if the original top-of-stack is not greater than (less than or equal to) what was the second-to-top.

<r Similar but invokes the macro if the original top-of-stack is less.

!<*r* Similar but invokes the macro if the original top-of-stack is not less than (greater than or equal to) what was the second-to-top.

=r Similar but invokes the macro if the two numbers popped are equal.

!=*r* Similar but invokes the macro if the two numbers popped are not equal.

? Reads a line from the terminal and executes it. This command allows a macro to request input from the user.

q exits from a macro and also from the macro which invoked it. If called from the top level, or from a macro which was called directly from the top level, the **q** command will cause *dc* to exit.

Q Pops a value off the stack and uses it as a count of levels of macro execution to be exited. Thus, **3Q** exits three levels. The **Q** command will never cause *dc* to exit.

Status Inquiry

Z Pops a value off the stack, calculates the number of digits it has (or number of characters, if it is a string) and pushes that number.

X Pops a value off the stack, calculates the number of fraction digits it has, and pushes that number. For a string, the value pushed is 0.

z Pushes the current stack depth: the number of objects on the stack before the execution of the **z** command.

Miscellaneous

! Will run the rest of the line as a system command. Note that parsing of the !<, !=, and !> commands take precedence, so if you want to run a command starting with <, =, or > you will need to add a space after the !.

Will interpret the rest of the line as a comment.

:*r* Will pop the top two values off of the stack. The old second-to-top value will be stored in the array *r*, indexed by the old top-of-stack value.

;*r* Pops the top-of-stack and uses it as an index into the array *r*. The selected value is then pushed onto the stack.

Note that each stacked instance of a register has its own array associated with it. Thus **1 0:a 0Sa 2 0:a La 0;ap** will print 1, because the 2 was stored in an instance of 0:a that was later popped.

BUGS

Email bug reports to **bug-dc@gnu.org**.

NAME

dd — convert and copy a file

SYNOPSIS

dd [operands ...]

DESCRIPTION

The **dd** utility copies the standard input to the standard output. Input data is read and written in 512-byte blocks. If input reads are short, input from multiple reads are aggregated to form the output block. When finished, **dd** displays the number of complete and partial input and output blocks and truncated input records to the standard error output.

The following operands are available:

bs=*n* Set both input and output block size, superseding the **ibs** and **obs** operands. If no conversion values other than **noerror**, **notrunc** or **sync** are specified, then each input block is copied to the output as a single block without any aggregation of short blocks.

cbs=*n* Set the conversion record size to *n* bytes. The conversion record size is required by the record oriented conversion values.

count=*n* Copy only *n* input blocks.

files=*n* Copy *n* input files before terminating. This operand is only applicable when the input device is a tape.

ibs=*n* Set the input block size to *n* bytes instead of the default 512.

if=*file* Read input from *file* instead of the standard input.

obs=*n* Set the output block size to *n* bytes instead of the default 512.

of=*file* Write output to *file* instead of the standard output. Any regular output file is truncated unless the **notrunc** conversion value is specified. If an initial portion of the output file is skipped (see the **seek** operand) the output file is truncated at that point.

seek=*n* Seek *n* blocks from the beginning of the output before copying. On non-tape devices, a `lseek`(2) operation is used. Otherwise, existing blocks are read and the data discarded. If the user does not have read permission for the tape, it is positioned using the tape `ioctl`(2) function calls. If the seek operation is past the end of file, space from the current end of file to the specified offset is filled with blocks of NUL bytes.

skip=*n* Skip *n* blocks from the beginning of the input before copying. On input which supports seeks, a `lseek`(2) operation is used. Otherwise, input data is read and discarded. For pipes, the correct number of bytes is read. For all other devices, the correct number of blocks is read without distinguishing between a partial or complete block being read.

progress=*n*

Switch on display of progress if *n* is set to any non-zero value. This will cause a "." to be printed (to the standard error output) for every *n* full or partial blocks written to the output file.

conv=value[,**value...**]

Where **value** is one of the symbols from the following list.

ascii, **oldascii**

The same as the **unblock** value except that characters are translated from EBCDIC to ASCII before the records are converted. (These values imply **unblock** if the operand **cbs** is also specified.) There are two conversion maps for ASCII. The value **ascii** specifies the recommended one which is compatible with AT&T System V

UNIX. The value **oldascii** specifies the one used in historic AT&T and pre-4.3BSD−Reno systems.

block Treats the input as a sequence of newline or end-of-file terminated variable length records independent of input and output block boundaries. Any trailing newline character is discarded. Each input record is converted to a fixed length output record where the length is specified by the **cbs** operand. Input records shorter than the conversion record size are padded with spaces. Input records longer than the conversion record size are truncated. The number of truncated input records, if any, are reported to the standard error output at the completion of the copy.

ebcdic, **ibm**, **oldebcdic**, **oldibm**
The same as the **block** value except that characters are translated from ASCII to EBCDIC after the records are converted. (These values imply **block** if the operand **cbs** is also specified.) There are four conversion maps for EBCDIC. The value **ebcdic** specifies the recommended one which is compatible with AT&T System V UNIX. The value **ibm** is a slightly different mapping, which is compatible with the AT&T System V UNIX **ibm** value. The values **oldebcdic** and **oldibm** are maps used in historic AT&T and pre 4.3BSD−Reno systems.

lcase Transform uppercase characters into lowercase characters.

noerror Do not stop processing on an input error. When an input error occurs, a diagnostic message followed by the current input and output block counts will be written to the standard error output in the same format as the standard completion message. If the **sync** conversion is also specified, any missing input data will be replaced with NUL bytes (or with spaces if a block oriented conversion value was specified) and processed as a normal input buffer. If the **sync** conversion is not specified, the input block is omitted from the output. On input files which are not tapes or pipes, the file offset will be positioned past the block in which the error occurred using `lseek`(2).

notrunc Do not truncate the output file. This will preserve any blocks in the output file not explicitly written by **dd**. The **notrunc** value is not supported for tapes.

osync Pad the final output block to the full output block size. If the input file is not a multiple of the output block size after conversion, this conversion forces the final output block to be the same size as preceding blocks for use on devices that require regularly sized blocks to be written. This option is incompatible with use of the **bs=***n* block size specification.

sparse If one or more non-final output blocks would consist solely of `NUL` bytes, try to seek the output file by the required space instead of filling them with `NUL`s. This results in a sparse file on some file systems.

swab Swap every pair of input bytes. If an input buffer has an odd number of bytes, the last byte will be ignored during swapping.

sync Pad every input block to the input buffer size. Spaces are used for pad bytes if a block oriented conversion value is specified, otherwise NUL bytes are used.

ucase Transform lowercase characters into uppercase characters.

unblock Treats the input as a sequence of fixed length records independent of input and output block boundaries. The length of the input records is specified by the **cbs** operand. Any trailing space characters are discarded and a newline character is appended.

Where sizes are specified, a decimal number of bytes is expected. Two or more numbers may be separated by an "x" to indicate a product. Each number may have one of the following optional suffixes:

- b Block; multiply by 512
- k Kibi; multiply by 1024 (1 KiB)
- m Mebi; multiply by 1048576 (1 MiB)
- g Gibi; multiply by 1073741824 (1 GiB)
- t Tebi; multiply by 1099511627776 (1 TiB)
- w Word; multiply by the number of bytes in an integer

When finished, **dd** displays the number of complete and partial input and output blocks, truncated input records and odd-length byte-swapping blocks to the standard error output. A partial input block is one where less than the input block size was read. A partial output block is one where less than the output block size was written. Partial output blocks to tape devices are considered fatal errors. Otherwise, the rest of the block will be written. Partial output blocks to character devices will produce a warning message. A truncated input block is one where a variable length record oriented conversion value was specified and the input line was too long to fit in the conversion record or was not newline terminated.

Normally, data resulting from input or conversion or both are aggregated into output blocks of the specified size. After the end of input is reached, any remaining output is written as a block. This means that the final output block may be shorter than the output block size.

If **dd** receives a SIGINFO signal (see the **status** argument for stty(1)), the current input and output block counts will be written to the standard error output in the same format as the standard completion message. If **dd** receives a SIGINT signal, the current input and output block counts will be written to the standard error output in the same format as the standard completion message and **dd** will exit.

EXIT STATUS

The **dd** utility exits 0 on success and >0 if an error occurred.

SEE ALSO

cp(1), mt(1), tr(1)

STANDARDS

The **dd** utility is expected to be a superset of the IEEE Std 1003.2 ("POSIX.2") standard. The **files** operand and the **ascii**, **ebcdic**, **ibm**, **oldascii**, **oldebcdic** and **oldibm** values are extensions to the POSIX standard.

NAME

deroff — remove nroff/troff, eqn, pic and tbl constructs

SYNOPSIS

deroff [**-ikpw**][**-m***a* | *e* | *l* | *m* | *s*][*file ...*]

DESCRIPTION

deroff reads each file in sequence and removes all nroff(1) and troff(1) command lines, backslash constructions, macro definitions, eqn(1) constructs (between ".EQ" and ".EN" lines or between delimiters), pic(1) pictures, and table descriptions and writes the remainder to the standard output. **deroff** follows chains of included files (".so" and ".nx" commands); if a file has already been included, a ".so" is ignored and a ".nx" terminates execution. If no input file is given, **deroff** reads from the standard input.

The options are as follows:

-i Ignore ".so" and ".nx" commands.

-k Keep blocks of text intact. This is the default behavior unless the **-m** option is given.

-m Enable support for common macro packages. The **-m** option takes the following arguments:

- a recognize man(7) macros.
- e recognize me(7) macros.
- l remove list constructs.
- m recognize mm(7) macros.
- s recognize ms(7) macros.

-p Preserve paragraph macros. This option only has an effect if the **-m** option is also specified.

-w Output a word list, one 'word' (string of letters, digits, and apostrophes, beginning with a letter; apostrophes are removed) per line, and all other characters ignored. Normally, the output follows the original, with the deletions mentioned above.

SEE ALSO

eqn(1), nroff(1), pic(1), tbl(1), troff(1)

HISTORY

deroff appeared in Version 7 AT&T UNIX.

BUGS

deroff is not a complete troff(1) interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

The **-ml** option does not correctly handle nested lists.

NAME

df — display free disk space

SYNOPSIS

df [**-agklmn**] [**-G** | **-i** | **-P**] [**-t** *type*] [*file* | *file_system* ...]

DESCRIPTION

df displays statistics about the amount of free disk space on the specified *file_system* or on the file system of which *file* is a part. By default, all sizes are reported in 512-byte block counts. If neither a file or a *file_system* operand is specified, statistics for all mounted file systems are displayed (subject to the **-l** and **-t** options below).

Note that the printed count of available blocks takes *minfree* into account, and thus will be negative when the number of free blocks on the filesystem is less than *minfree*.

The following options are available:

-a Show all mount points, including those that were mounted with the MNT_IGNORE flag.

-G Display all the fields of the structure(s) returned by statvfs(2). This option cannot be used with the **-i** or **-P** options, and it is modelled after the Solaris **-g** option. This option will override the **-g**, **-h**, **-k**, and **-m** options, as well as any setting of BLOCKSIZE.

-g The **-g** option causes the numbers to be reported in gigabytes (1024∗1024∗1024 bytes).

-h "Human-readable" output. Use unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte, Exabyte in order to reduce the number of digits to four or less.

-i Include statistics on the number of free inodes.

-k By default, all sizes are reported in 512-byte block counts. The **-k** option causes the numbers to be reported in kilobytes (1024 bytes).

-l Display statistics only about mounted file systems with the MNT_LOCAL flag set. If a non-local file system is given as an argument, a warning is issued and no information is given on that file system.

-m The **-m** option causes the numbers to be reported in megabytes (1024∗1024 bytes).

-n Print out the previously obtained statistics from the file systems. This option should be used if it is possible that one or more file systems are in a state such that they will not be able to provide statistics without a long delay. When this option is specified, **df** will not request new statistics from the file systems, but will respond with the possibly stale statistics that were previously obtained.

-P Produce output in the following portable format:

If both the **-P** and **-k** option are specified, the output will be preceded by the following header line, formatted to match the data following it:

```
"Filesystem 1024-blocks Used Available Capacity Mounted on\n"
```

If the **-P** option is specified without the **-k** options, the output will be preceded by the following header line, formatted to match the data following it:

```
"Filesystem <blksize>-blocks Used Available Capacity Mounted on\n"
```

The header line is followed by data formatted as follows:

```
"%s %d %d %d %d%% %s\n", <file system name>, <total space>,
    <space used>, <space free>, <percentage used>,
    <file system root>
```

Note that the **-i** option may not be specified with **-P**.

-t *type*
Is used to indicate the actions should only be taken on filesystems of the specified type. More than one type may be specified in a comma-separated list. The list of filesystem types can be prefixed with "no" to specify the filesystem types for which action should *not* be taken. If a file system is given on the command line that is not of the specified type, a warning is issued and no information is given on that file system.

ENVIRONMENT

BLOCKSIZE If the environment variable BLOCKSIZE is set, and the **-g**, **-h**, **-k** and **-m** options are not specified, the block counts will be displayed in units of that size block.

SEE ALSO

quota(1), fstatvfs(2), getvfsstat(2), statvfs(2), getbsize(3), getmntinfo(3), fs(5), fstab(5), mount(8), quot(8), tunefs(8)

HISTORY

A **df** utility appeared in Version 6 AT&T UNIX.

NAME

diff − compare files line by line

SYNOPSIS

diff [*OPTION*]... *FILES*

DESCRIPTION

Compare files line by line.

−i −−ignore−case
Ignore case differences in file contents.

−−ignore−file−name−case
Ignore case when comparing file names.

−−no−ignore−file−name−case
Consider case when comparing file names.

−E −−ignore−tab−expansion
Ignore changes due to tab expansion.

−b −−ignore−space−change
Ignore changes in the amount of white space.

−w −−ignore−all−space
Ignore all white space.

−B −−ignore−blank−lines
Ignore changes whose lines are all blank.

−I RE **−−ignore−matching−lines**=*RE*
Ignore changes whose lines all match RE.

−−strip−trailing−cr
Strip trailing carriage return on input.

−a −−text
Treat all files as text.

−c −C NUM **−−context**[=*NUM*]
Output NUM (default 3) lines of copied context.

−u −U NUM **−−unified**[=*NUM*]
Output NUM (default 3) lines of unified context.

−−label LABEL
Use LABEL instead of file name.

−p −−show−c−function
Show which C function each change is in.

−F RE **−−show−function−line**=*RE*
Show the most recent line matching RE.

−q −−brief
Output only whether files differ.

−e −−ed
Output an ed script.

−−normal
Output a normal diff.

−n −−rcs
Output an RCS format diff.

−y −−side−by−side
Output in two columns.

−W NUM **−−width**=*NUM*
Output at most NUM (default 130) print columns.

−−left−column
Output only the left column of common lines.

−−suppress−common−lines
Do not output common lines.

−D NAME **−−ifdef**=*NAME*
Output merged file to show ‘#ifdef NAME’ diffs.

−−GTYPE−group−format=*GFMT*
Similar, but format GTYPE input groups with GFMT.

−−line−format=*LFMT*
Similar, but format all input lines with LFMT.

−−LTYPE−line−format=*LFMT*
Similar, but format LTYPE input lines with LFMT.

LTYPE is ‘old’, ‘new’, or ‘unchanged’.
GTYPE is LTYPE or ‘changed’.

GFMT may contain:

%< lines from FILE1

%> lines from FILE2

%= lines common to FILE1 and FILE2

%[-][WIDTH][.[PREC]]{doxX}LETTER
printf-style spec for LETTER

LETTERs are as follows for new group, lower case for old group:

F first line number

L last line number

N number of lines = L-F+1

E F-1

M L+1

LFMT may contain:

%L contents of line

%l contents of line, excluding any trailing newline

%[-][WIDTH][.[PREC]]{doxX}n
printf-style spec for input line number

Either GFMT or LFMT may contain:

%% %

%c’C’ the single character C

%c’\OOO’
the character with octal code OOO

−l −−paginate
Pass the output through ‘pr’ to paginate it.

−t −−expand−tabs
Expand tabs to spaces in output.

−T −−initial−tab
Make tabs line up by prepending a tab.

−r −−recursive
Recursively compare any subdirectories found.

−N −−new−file
Treat absent files as empty.

−−unidirectional−new−file
Treat absent first files as empty.

−s −−report−identical−files
Report when two files are the same.

−x PAT **−−exclude**=*PAT*
Exclude files that match PAT.

−X FILE **−−exclude−from**=*FILE*
Exclude files that match any pattern in FILE.

−S FILE **−−starting−file**=*FILE*
Start with FILE when comparing directories.

−−from−file=*FILE1*
Compare FILE1 to all operands. FILE1 can be a directory.

−−to−file=*FILE2*
Compare all operands to FILE2. FILE2 can be a directory.

−−horizon−lines=*NUM*
Keep NUM lines of the common prefix and suffix.

−d −−minimal
Try hard to find a smaller set of changes.

−−speed−large−files
Assume large files and many scattered small changes.

−v −−version
Output version info.

−−help Output this help.

FILES are ‘FILE1 FILE2’ or ‘DIR1 DIR2’ or ‘DIR FILE...’ or ‘FILE... DIR’. If **−−from−file** or **−−to−file** is given, there are no restrictions on FILES. If a FILE is ‘-’, read standard input.

AUTHOR

Written by Paul Eggert, Mike Haertel, David Hayes, Richard Stallman, and Len Tower.

REPORTING BUGS

Report bugs to <bug-gnu-utils@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **diff** is maintained as a Texinfo manual. If the **info** and **diff** programs are properly installed at your site, the command

info diff

should give you access to the complete manual.

NAME

diff3 − compare three files line by line

SYNOPSIS

diff3 [*OPTION*]... *MYFILE OLDFILE YOURFILE*

DESCRIPTION

Compare three files line by line.

−e −−ed
Output unmerged changes from OLDFILE to YOURFILE into MYFILE.

−E −−show−overlap
Output unmerged changes, bracketing conflicts.

−A −−show−all
Output all changes, bracketing conflicts.

−x −−overlap−only
Output overlapping changes.

−X Output overlapping changes, bracketing them.

−3 −−easy−only
Output unmerged nonoverlapping changes.

−m −−merge
Output merged file instead of ed script (default **−A**).

−L LABEL **−−label**=*LABEL*
Use LABEL instead of file name.

−i Append ‘w’ and ‘q’ commands to ed scripts.

−a −−text
Treat all files as text.

−T −−initial−tab
Make tabs line up by prepending a tab.

−−diff−program=*PROGRAM*
Use PROGRAM to compare files.

−v −−version
Output version info.

−−help Output this help.

If a FILE is ‘-’, read standard input.

AUTHOR

Written by Randy Smith.

REPORTING BUGS

Report bugs to <bug-gnu-utils@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **diff3** is maintained as a Texinfo manual. If the **info** and **diff3** programs are properly installed at your site, the command

info diff

should give you access to the complete manual.

NAME

dig − DNS lookup utility

SYNOPSIS

dig [@server] [**−b** *address*] [**−c** *class*] [**−f** *filename*] [**−k** *filename*] [**−p** *port#*] [**−q** *name*] [**−t** *type*] [**−x** *addr*] [**−y** *[hmac:]name:key*] [**−4**] [**−6**] [name] [type] [class] [queryopt...]

dig [**−h**]

dig [global−queryopt...] [query...]

DESCRIPTION

dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use **dig** to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than **dig**.

Although **dig** is normally used with command−line arguments, it also has a batch mode of operation for reading lookup requests from a file. A brief summary of its command−line arguments and options is printed when the **−h** option is given. Unlike earlier versions, the BIND9 implementation of **dig** allows multiple lookups to be issued from the command line.

Unless it is told to query a specific name server, **dig** will try each of the servers listed in */etc/resolv.conf*.

When no command line arguments or options are given, will perform an NS query for "." (the root).

It is possible to set per−user defaults for **dig** via *${HOME}/.digrc*. This file is read and any options in it are applied before the command line arguments.

The IN and CH class names overlap with the IN and CH top level domains names. Either use the **−t** and **−c** options to specify the type and class or use the **−q** the specify the domain name or use "IN." and "CH." when looking up these top level domains.

SIMPLE USAGE

A typical invocation of **dig** looks like:

```
dig @server name type
```

where:

server
: is the name or IP address of the name server to query. This can be an IPv4 address in dotted−decimal notation or an IPv6 address in colon−delimited notation. When the supplied *server* argument is a hostname, **dig** resolves that name before querying that name server. If no *server* argument is provided, **dig** consults */etc/resolv.conf* and queries the name servers listed there. The reply from the name server that responds is displayed.

name
: is the name of the resource record that is to be looked up.

type
: indicates what type of query is required — ANY, A, MX, SIG, etc. *type* can be any valid query type. If no *type* argument is supplied, **dig** will perform a lookup for an A record.

OPTIONS

The **−b** option sets the source IP address of the query to *address*. This must be a valid address on one of the host's network interfaces or "0.0.0.0" or "::". An optional port may be specified by appending "#<port>"

The default query class (IN for internet) is overridden by the **−c** option. *class* is any valid class, such as HS for Hesiod records or CH for CHAOSNET records.

The **−f** option makes **dig** operate in batch mode by reading a list of lookup requests to process from the file *filename*. The file contains a number of queries, one per line. Each entry in the file should be organised in the same way they would be presented as queries to **dig** using the command−line interface.

If a non−standard port number is to be queried, the **−p** option is used. *port#* is the port number that **dig** will send its queries instead of the standard DNS port number 53. This option would be used to test a name server that has been configured to listen for queries on a non−standard port number.

The **−4** option forces **dig** to only use IPv4 query transport. The **−6** option forces **dig** to only use IPv6 query transport.

The **−t** option sets the query type to *type*. It can be any valid query type which is supported in BIND9. The default query type "A", unless the **−x** option is supplied to indicate a reverse lookup. A zone transfer can be requested by specifying a type of AXFR. When an incremental zone transfer (IXFR) is required, *type* is set to ixfr=N. The incremental zone transfer will contain the changes made to the zone since the serial number in the zone's SOA record was *N*.

The **−q** option sets the query name to *name*. This useful do distingish the *name* from other arguments.

Reverse lookups − mapping addresses to names − are simplified by the **−x** option. *addr* is an IPv4 address in dotted−decimal notation, or a colon−delimited IPv6 address. When this option is used, there is no need to provide the *name*, *class* and *type* arguments. **dig** automatically performs a lookup for a name like 11.12.13.10.in−addr.arpa and sets the query type and class to PTR and IN respectively. By default, IPv6 addresses are looked up using nibble format under the IP6.ARPA domain. To use the older RFC1886 method using the IP6.INT domain specify the **−i** option. Bit string labels (RFC2874) are now experimental and are not attempted.

To sign the DNS queries sent by **dig** and their responses using transaction signatures (TSIG), specify a TSIG key file using the **−k** option. You can also specify the TSIG key itself on the command line using the **−y** option; *hmac* is the type of the TSIG, default HMAC−MD5, *name* is the name of the TSIG key and *key* is the actual key. The key is a base−64 encoded string, typically generated by **dnssec−keygen**(8). Caution should be taken when using the **−y** option on multi−user systems as the key can be visible in the output from **ps**(1) or in the shell's history file. When using TSIG authentication with **dig**, the name server that is queried needs to know the key and algorithm that is being used. In BIND, this is done by providing appropriate **key** and **server** statements in *named.conf*.

QUERY OPTIONS

dig provides a number of query options which affect the way in which lookups are made and the results displayed. Some of these set or reset flag bits in the query header, some determine which sections of the answer get printed, and others determine the timeout and retry strategies.

Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These may be preceded by the string no to negate the meaning of that keyword. Other keywords assign values to options like the timeout interval. They have the form **+keyword=value**. The query options are:

+[no]tcp
Use [do not use] TCP when querying name servers. The default behaviour is to use UDP unless an AXFR or IXFR query is requested, in which case a TCP connection is used.

+[no]vc
Use [do not use] TCP when querying name servers. This alternate syntax to *+[no]tcp* is provided for backwards compatibility. The "vc" stands for "virtual circuit".

+[no]ignore
Ignore truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

+domain=somename
Set the search list to contain the single domain *somename*, as if specified in a **domain** directive in */etc/resolv.conf*, and enable search list processing as if the *+search* option were given.

+[no]search
Use [do not use] the search list defined by the searchlist or domain directive in *resolv.conf* (if any). The search list is not used by default.

+[no]showsearch
Perform [do not perform] a search showing intermediate results.

+[no]defname
Deprecated, treated as a synonym for *+[no]search*

+[no]aaonly
Sets the "aa" flag in the query.

+[no]aaflag
A synonym for *+[no]aaonly*.

+[no]adflag
Set [do not set] the AD (authentic data) bit in the query. The AD bit currently has a standard meaning only in responses, not in queries, but the ability to set the bit in the query is provided for completeness.

+[no]cdflag
Set [do not set] the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

+[no]cl
Display [do not display] the CLASS when printing the record.

+[no]ttlid
Display [do not display] the TTL when printing the record.

+[no]recurse
Toggle the setting of the RD (recursion desired) bit in the query. This bit is set by default, which means **dig** normally sends recursive queries. Recursion is automatically disabled when the *+nssearch* or *+trace* query options are used.

+[no]nssearch
When this option is set, **dig** attempts to find the authoritative name servers for the zone containing the name being looked up and display the SOA record that each name server has for the zone.

+[no]trace
Toggle tracing of the delegation path from the root name servers for the name being looked up. Tracing is disabled by default. When tracing is enabled, **dig** makes iterative queries to resolve the name being looked up. It will follow referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

+[no]cmd
toggles the printing of the initial comment in the output identifying the version of **dig** and the query options that have been applied. This comment is printed by default.

+[no]short
Provide a terse answer. The default is to print the answer in a verbose form.

+[no]identify
Show [or do not show] the IP address and port number that supplied the answer when the *+short* option is enabled. If short form answers are requested, the default is not to show the source address and port number of the server that provided the answer.

+[no]comments
Toggle the display of comment lines in the output. The default is to print comments.

+[no]stats
This query option toggles the printing of statistics: when the query was made, the size of the reply and so on. The default behaviour is to print the query statistics.

+[no]qr
Print [do not print] the query as it is sent. By default, the query is not printed.

+[no]question
Print [do not print] the question section of a query when an answer is returned. The default is to print

the question section as a comment.

+[no]answer
Display [do not display] the answer section of a reply. The default is to display it.

+[no]authority
Display [do not display] the authority section of a reply. The default is to display it.

+[no]additional
Display [do not display] the additional section of a reply. The default is to display it.

+[no]all
Set or clear all display flags.

+time=T
Sets the timeout for a query to *T* seconds. The default time out is 5 seconds. An attempt to set *T* to less than 1 will result in a query timeout of 1 second being applied.

+tries=T
Sets the number of times to try UDP queries to server to *T* instead of the default, 3. If *T* is less than or equal to zero, the number of tries is silently rounded up to 1.

+retry=T
Sets the number of times to retry UDP queries to server to *T* instead of the default, 2. Unlike *+tries*, this does not include the initial query.

+ndots=D
Set the number of dots that have to appear in *name* to *D* for it to be considered absolute. The default value is that defined using the ndots statement in */etc/resolv.conf*, or 1 if no ndots statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the **search** or **domain** directive in */etc/resolv.conf*.

+bufsize=B
Set the UDP message buffer size advertised using EDNS0 to *B* bytes. The maximum and minimum sizes of this buffer are 65535 and 0 respectively. Values outside this range are rounded up or down appropriately. Values other than zero will cause a EDNS query to be sent.

+edns=#
Specify the EDNS version to query with. Valid values are 0 to 255. Setting the EDNS version will cause a EDNS query to be sent. **+noedns** clears the remembered EDNS version.

+[no]multiline
Print records like the SOA records in a verbose multi−line format with human−readable comments. The default is to print each record on a single line, to facilitate machine parsing of the **dig** output.

+[no]fail
Do not try the next server if you receive a SERVFAIL. The default is to not try the next server which is the reverse of normal stub resolver behaviour.

+[no]besteffort
Attempt to display the contents of messages which are malformed. The default is to not display malformed answers.

+[no]dnssec
Requests DNSSEC records be sent by setting the DNSSEC OK bit (DO) in the OPT record in the additional section of the query.

+[no]sigchase
Chase DNSSEC signature chains. Requires dig be compiled with −DDIG_SIGCHASE.

+trusted−key=####
Specifies a file containing trusted keys to be used with **+sigchase**. Each DNSKEY record must be on its own line.

If not specified **dig** will look for */etc/trusted−key.key* then *trusted−key.key* in the current directory.

Requires dig be compiled with −DDIG_SIGCHASE.

+[no]topdown
When chasing DNSSEC signature chains perform a top down validation. Requires dig be compiled with −DDIG_SIGCHASE.

MULTIPLE QUERIES

The BIND 9 implementation of **dig** supports specifying multiple queries on the command line (in addition to supporting the **−f** batch file option). Each of those queries can be supplied with its own set of flags, options and query options.

In this case, each *query* argument represent an individual query in the command−line syntax described above. Each consists of any of the standard options and flags, the name to be looked up, an optional query type and class and any query options that should be applied to that query.

A global set of query options, which should be applied to all queries, can also be supplied. These global query options must precede the first tuple of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except the **+[no]cmd** option) can be overridden by a query−specific set of query options. For example:

dig +qr www.isc.org any −x 127.0.0.1 isc.org ns +noqr

shows how **dig** could be used from the command line to make three lookups: an ANY query for www.isc.org, a reverse lookup of 127.0.0.1 and a query for the NS records of isc.org. A global query option of *+qr* is applied, so that **dig** shows the initial query it made for each lookup. The final query has a local query option of *+noqr* which means that **dig** will not print the initial query when it looks up the NS records for isc.org.

IDN SUPPORT

If **dig** has been built with IDN (internationalized domain name) support, it can accept and display non−ASCII domain names. **dig** appropriately converts character encoding of domain name before sending a request to DNS server or displaying a reply from the server. If you'd like to turn off the IDN support for some reason, defines the **IDN_DISABLE** environment variable. The IDN support is disabled if the variable is set when **dig** runs.

FILES

/etc/resolv.conf

${HOME}/.digrc

SEE ALSO

host(1), **named**(8), **dnssec−keygen**(8), RFC1035.

BUGS

There are probably too many query options.

COPYRIGHT

NAME

dlltool − Create files needed to build and use DLLs.

SYNOPSIS

dlltool [**−d**|**−−input−def** *def-file-name*]
[**−b**|**−−base−file** *base-file-name*]
[**−e**|**−−output−exp** *exports-file-name*]
[**−z**|**−−output−def** *def-file-name*]
[**−l**|**−−output−lib** *library-file-name*]
[**−−export−all−symbols**] [**−−no−export−all−symbols**]
[**−−exclude−symbols** *list*]
[**−−no−default−excludes**]
[**−S**|**−−as** *path-to-assembler*] [**−f**|**−−as−flags** *options*]
[**−D**|**−−dllname** *name*] [**−m**|**−−machine** *machine*]
[**−a**|**−−add−indirect**] [**−U**|**−−add−underscore**] [**−k**|**−−kill−at**]
[**−A**|**−−add−stdcall−alias**]
[**−p**|**−−ext−prefix−alias** *prefix*]
[**−x**|**−−no−idata4**] [**−c**|**−−no−idata5**] [**−i**|**−−interwork**]
[**−n**|**−−nodelete**] [**−t**|**−−temp−prefix** *prefix*]
[**−v**|**−−verbose**]
[**−h**|**−−help**] [**−V**|**−−version**]
[object−file ...]

DESCRIPTION

dlltool reads its inputs, which can come from the **−d** and **−b** options as well as object files specified on the command line. It then processes these inputs and if the **−e** option has been specified it creates a exports file. If the **−l** option has been specified it creates a library file and if the **−z** option has been specified it creates a def file. Any or all of the **−e**, **−l** and **−z** options can be present in one invocation of dlltool.

When creating a DLL, along with the source for the DLL, it is necessary to have three other files. **dlltool** can help with the creation of these files.

The first file is a *.def* file which specifies which functions are exported from the DLL, which functions the DLL imports, and so on. This is a text file and can be created by hand, or **dlltool** can be used to create it using the **−z** option. In this case **dlltool** will scan the object files specified on its command line looking for those functions which have been specially marked as being exported and put entries for them in the *.def* file it creates.

In order to mark a function as being exported from a DLL, it needs to have an **−export:<name_of_function>** entry in the **.drectve** section of the object file. This can be done in C by using the *asm()* operator:

```
asm (".section .drectve");
asm (".ascii \"−export:my_func\"");

int my_func (void) { ... }
```

The second file needed for DLL creation is an exports file. This file is linked with the object files that make up the body of the DLL and it handles the interface between the DLL and the outside world. This is a binary file and it can be created by giving the **−e** option to **dlltool** when it is creating or reading in a *.def* file.

The third file needed for DLL creation is the library file that programs will link with in order to access the functions in the DLL. This file can be created by giving the **−l** option to dlltool when it is creating or reading in a *.def* file.

dlltool builds the library file by hand, but it builds the exports file by creating temporary files containing assembler statements and then assembling these. The **−S** command line option can be used to specify the path to the assembler that dlltool will use, and the **−f** option can be used to pass specific flags to that assembler. The **−n** can be used to prevent dlltool from deleting these temporary assembler files when it is done, and if **−n** is specified twice then this will prevent dlltool from deleting the temporary object files it used to build the library.

Here is an example of creating a DLL from a source file **dll.c** and also creating a program (from an object file called **program.o**) that uses that DLL:

```
gcc -c dll.c
dlltool -e exports.o -l dll.lib dll.o
gcc dll.o exports.o -o dll.dll
gcc program.o dll.lib -o program
```

OPTIONS

The command line options have the following meanings:

-d *filename*
--input-def *filename*
Specifies the name of a *.def* file to be read in and processed.

-b *filename*
--base-file *filename*
Specifies the name of a base file to be read in and processed. The contents of this file will be added to the relocation section in the exports file generated by dlltool.

-e *filename*
--output-exp *filename*
Specifies the name of the export file to be created by dlltool.

-z *filename*
--output-def *filename*
Specifies the name of the *.def* file to be created by dlltool.

-l *filename*
--output-lib *filename*
Specifies the name of the library file to be created by dlltool.

--export-all-symbols
Treat all global and weak defined symbols found in the input object files as symbols to be exported. There is a small list of symbols which are not exported by default; see the **--no-default-excludes** option. You may add to the list of symbols to not export by using the **--exclude-symbols** option.

--no-export-all-symbols
Only export symbols explicitly listed in an input *.def* file or in **.drectve** sections in the input object files. This is the default behaviour. The **.drectve** sections are created by **dllexport** attributes in the source code.

--exclude-symbols *list*
Do not export the symbols in *list*. This is a list of symbol names separated by comma or colon characters. The symbol names should not contain a leading underscore. This is only meaningful when **--export-all-symbols** is used.

--no-default-excludes
When **--export-all-symbols** is used, it will by default avoid exporting certain special symbols. The current list of symbols to avoid exporting is **DllMain@12**, **DllEntryPoint@0**, **impure_ptr**. You may use the **--no-default-excludes** option to go ahead and export these special symbols. This is only meaningful when **--export-all-symbols** is used.

-S *path*
--as *path*
Specifies the path, including the filename, of the assembler to be used to create the exports file.

-f *options*
--as-flags *options*
Specifies any specific command line options to be passed to the assembler when building the exports file. This option will work even if the **-S** option is not used. This option only takes one argument, and if it occurs more than once on the command line, then later occurrences will override earlier

occurrences. So if it is necessary to pass multiple options to the assembler they should be enclosed in double quotes.

−D *name*
−−dll−name *name*
Specifies the name to be stored in the *.def* file as the name of the DLL when the **−e** option is used. If this option is not present, then the filename given to the **−e** option will be used as the name of the DLL.

−m *machine*
−machine *machine*
Specifies the type of machine for which the library file should be built. **dlltool** has a built in default type, depending upon how it was created, but this option can be used to override that. This is normally only useful when creating DLLs for an ARM processor, when the contents of the DLL are actually encode using Thumb instructions.

−a
−−add−indirect
Specifies that when **dlltool** is creating the exports file it should add a section which allows the exported functions to be referenced without using the import library. Whatever the hell that means!

−U
−−add−underscore
Specifies that when **dlltool** is creating the exports file it should prepend an underscore to the names of the exported functions.

−k
−−kill−at
Specifies that when **dlltool** is creating the exports file it should not append the string **@ <number>**. These numbers are called ordinal numbers and they represent another way of accessing the function in a DLL, other than by name.

−A
−−add−stdcall−alias
Specifies that when **dlltool** is creating the exports file it should add aliases for stdcall symbols without **@ <number>** in addition to the symbols with **@ <number>**.

−p
−−ext−prefix−alias *prefix*
Causes **dlltool** to create external aliases for all DLL imports with the specified prefix. The aliases are created for both external and import symbols with no leading underscore.

−x
−−no−idata4
Specifies that when **dlltool** is creating the exports and library files it should omit the `.idata4` section. This is for compatibility with certain operating systems.

−c
−−no−idata5
Specifies that when **dlltool** is creating the exports and library files it should omit the `.idata5` section. This is for compatibility with certain operating systems.

−i
−−interwork
Specifies that **dlltool** should mark the objects in the library file and exports file that it produces as supporting interworking between ARM and Thumb code.

−n
−−nodelete
Makes **dlltool** preserve the temporary assembler files it used to create the exports file. If this option is repeated then dlltool will also preserve the temporary object files it uses to create the library file.

−t *prefix*
−−temp−prefix *prefix*
Makes **dlltool** use *prefix* when constructing the names of temporary assembler and object files. By default, the temp file prefix is generated from the pid.

−v
−−verbose
Make dlltool describe what it is doing.

−h
−−help
Displays a list of command line options and then exits.

−V
−−version
Displays dlltool's version number and then exits.

SEE ALSO

The Info pages for *binutils*.

COPYRIGHT

NAME

domainname — set or print YP domain of current host system

SYNOPSIS

domainname [*name-of-domain*]

DESCRIPTION

domainname prints the YP domain name of the current host. The super-user can set the domain name by supplying an argument; this is usually done in the network initialization script `/etc/rc.d/network`, normally run at boot time.

SEE ALSO

`hostname`(1), `getdomainname`(3), `setdomainname`(3)

HISTORY

The **domainname** utility is derived from the **hostname** utility, which appeared in 4.2BSD. The **domainname** utility appeared in NetBSD 0.8.

NAME

dtmfdecode — decodes DTMF tones from A-law audio data

SYNOPSIS

dtmfdecode

DESCRIPTION

dtmfdecode is part of the isdn4bsd package and is used to detect DTMF tones in the audio stream.

It reads audio G.711 A-law coded data from stdin and outputs the detected numbers values as ASCII characters to stdout.

The detector is implemented as 8 narrow band-pass filters realized with an integer double-cross recursive algorithm. Various ad-hoc methods are employed to provide hysteresis and anti-bounce for the detected signals.

EXAMPLES

The command:

```
dtmfdecode < beep.al
```

will print a "1" to stdout.

STANDARDS

ITU Recommendations G.711

AUTHORS

The **dtmfdecode** utility was written by Poul-Henning Kamp ⟨phk@freebsd.org⟩. This man page was written by
Hellmuth Michaelis ⟨hm@freebsd.org⟩.

NAME

du — display disk usage statistics

SYNOPSIS

du [**-H** | **-L** | **-P**] [**-a** | **-d** *depth* | **-s**] [**-cghkmnrx**] [*file ...*]

DESCRIPTION

The **du** utility displays the file system block usage for each file argument and for each directory in the file hierarchy rooted in each directory argument. If no file is specified, the block usage of the hierarchy rooted in the current directory is displayed.

The options are as follows:

-H Symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.)

-L All symbolic links are followed.

-P No symbolic links are followed.

-a Display an entry for each file in the file hierarchy.

-c Display the grand total after all the arguments have been processed.

-d Display an entry files and directories *depth* directories deep.

-g If the **-g** flag is specified, the number displayed is the number of gigabyte (1024∗1024∗1024 bytes) blocks.

-h If the **-h** flag is specified, the numbers will be displayed in "human-readable" format. Use unit suffixes: B (Byte), K (Kilobyte), M (Megabyte), G (Gigabyte), T (Terabyte) and P (Petabyte).

-k By default, **du** displays the number of blocks as returned by the `stat`(2) system call, i.e. 512-byte blocks. If the **-k** flag is specified, the number displayed is the number of kilobyte (1024 bytes) blocks. Partial numbers of blocks are rounded up.

-m If the **-m** flag is specified, the number displayed is the number of megabyte (1024∗1024 bytes) blocks.

-n Ignore files and directories with user "nodump" flag (`UF_NODUMP`) set.

-r Generate warning messages about directories that cannot be read. This is the default behaviour.

-s Display only the grand total for the specified files.

-x Filesystem mount points are not traversed.

du counts the storage used by symbolic links and not the files they reference unless the **-H** or **-L** option is specified. If either the **-H** or **-L** options are specified, storage used by any symbolic links which are followed is not counted or displayed. The **-H**, **-L** and **-P** options override each other and the command's actions are determined by the last one specified.

Files having multiple hard links are counted (and displayed) a single time per **du** execution.

ENVIRONMENT

`BLOCKSIZE` If the environment variable `BLOCKSIZE` is set, and the **-g**, **-h**, **-k**, and **-m** options are not specified, the block counts will be displayed in units of that size block.

SEE ALSO

`df`(1), `chflags`(2), `fts`(3), `getbsize`(3), `symlink`(7), `quot`(8)

HISTORY

A **du** command appeared in Version 6 AT&T UNIX.

NAME

echo — write arguments to the standard output

SYNOPSIS

echo [**-n**] [*string ...*]

DESCRIPTION

The **echo** utility writes any specified operands, separated by single blank (‘‘ ’’) characters and followed by a newline (‘‘\n’’) character, to the standard output.

The following option is available:

-n Do not print the trailing newline character.

EXIT STATUS

The **echo** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

printf(1)

STANDARDS

The **echo** utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

echo — produce message in a shell script

SYNOPSIS

echo [**-n** | **-e**] *args ...*

DESCRIPTION

echo prints its arguments on the standard output, separated by spaces. Unless the **-n** option is present, a newline is output following the arguments. The **-e** option causes **echo** to treat the escape sequences specially, as described in the following paragraph. The **-e** option is the default, and is provided solely for compatibility with other systems. Only one of the options **-n** and **-e** may be given.

If any of the following sequences of characters is encountered during output, the sequence is not output. Instead, the specified action is performed:

`\b` A backspace character is output.

`\c` Subsequent output is suppressed. This is normally used at the end of the last argument to suppress the trailing newline that **echo** would otherwise output.

`\f` Output a form feed.

`\n` Output a newline character.

`\r` Output a carriage return.

`\t` Output a (horizontal) tab character.

`\v` Output a vertical tab.

`\0`*digits*
Output the character whose value is given by zero to three digits. If there are zero digits, a nul character is output.

`\\` Output a backslash.

HINTS

Remember that backslash is special to the shell and needs to be escaped. To output a message to standard error, say

```
echo message >&2
```

BUGS

The octal character escape mechanism (`\0`*digits*) differs from the C language mechanism.

There is no way to force **echo** to treat its arguments literally, rather than interpreting them as options and escape sequences.

NAME

ed — text editor

SYNOPSIS

ed [**-**] [**-Esx**] [**-p** *string*] [*file*]

DESCRIPTION

ed is a line-oriented text editor. It is used to create, display, modify, and otherwise manipulate text files. If invoked with a *file* argument, then a copy of *file* is read into the editor's buffer. Changes are made to this copy and not directly to *file* itself. Upon quitting **ed**, any changes not explicitly saved with a **w** command are lost.

Editing is done in two distinct modes: *command* and *input*. When first invoked, **ed** is in command mode. In this mode, commands are read from the standard input and executed to manipulate the contents of the editor buffer.

A typical command might look like:

```
,s/old/new/g
```

which replaces all occurrences of the string `old` with `new`.

When an input command, such as **a** (append), **i** (insert), or **c** (change) is given, **ed** enters input mode. This is the primary means of adding text to a file. In this mode, no commands are available; instead, the standard input is written directly to the editor buffer. Lines consist of text up to and including a newline character. Input mode is terminated by entering a single period (`.`) on a line.

All **ed** commands operate on whole lines or ranges of lines; e.g., the **d** command deletes lines; the **m** command moves lines, and so on. It is possible to modify only a portion of a line by means of replacement, as in the example above. However, even here, the **s** command is applied to whole lines at a time.

In general, **ed** commands consist of zero or more line addresses, followed by a single character command and possibly additional parameters; i.e., commands have the structure:

```
[address [,address]]command[parameters]
```

The address(es) indicate the line or range of lines to be affected by the command. If fewer addresses are given than the command accepts, then default addresses are supplied.

The options are as follows:

- Same as the **-s** option (deprecated).

-E Enables the use of extended regular expressions instead of the basic regular expressions that are normally used.

-p *string*
Specifies a command prompt. This may be toggled on and off with the **P** command.

-s Suppress diagnostics. This should be used if **ed** standard input is from a script.

-x Prompt for an encryption key to be used in subsequent reads and writes (see the **x** command).

file Specifies the name of a file to read. If *file* is prefixed with a bang (`!`), then it is interpreted as a shell command. In this case, what is read is the standard output of *file* executed via `sh`(1). To read a file whose name begins with a bang, prefix the name with a backslash (`\`). The default filename is set to *file* only if it is not prefixed with a bang.

LINE ADDRESSING

An address represents the number of a line in the buffer. **ed** maintains a *current address* which is typically supplied to commands as the default address when none is specified. When a file is first read, the current address is set to the last line of the file. In general, the current address is set to the last line affected by a command.

A line address is constructed from one of the bases in the list below, optionally followed by a numeric offset. The offset may include any combination of digits, operators (i.e., '+', '-', and '^'), and whitespace. Addresses are read from left to right, and their values are computed relative to the current address.

One exception to the rule that addresses represent line numbers is the address *0* (zero). This means "before the first line", and is legal wherever it makes sense.

An address range is two addresses separated either by a comma or semi-colon. The value of the first address in a range cannot exceed the value of the second. If only one address is given in a range, then the second address is set to the given address. If an *n*-tuple of addresses is given where *n* > *2*, then the corresponding range is determined by the last two addresses in the *n*-tuple. If only one address is expected, then the last address is used.

Each address in a comma-delimited range is interpreted relative to the current address. In a semi-colon-delimited range, the first address is used to set the current address, and the second address is interpreted relative to the first.

The following address symbols are recognized:

. The current line (address) in the buffer.

$ The last line in the buffer.

n The *n*th line in the buffer where *n* is a number in the range *[0,$]*.

- or ^ The previous line. This is equivalent to *-1* and may be repeated with cumulative effect.

-n or *^n* The *n*th previous line, where *n* is a non-negative number.

\+ The next line. This is equivalent to *+1* and may be repeated with cumulative effect.

+n The *n*th next line, where *n* is a non-negative number.

whitespace n
whitespace followed by a number *n* is interpreted as '*+n*'.

, or % The first through last lines in the buffer. This is equivalent to the address range *1,$*.

; The current through last lines in the buffer. This is equivalent to the address range *.,$*.

/re/ The next line containing the regular expression *re*. The search wraps to the beginning of the buffer and continues down to the current line, if necessary. *//* repeats the last search.

?*re*? The previous line containing the regular expression *re*. The search wraps to the end of the buffer and continues up to the current line, if necessary. *??* repeats the last search.

'lc The line previously marked by a **k** (mark) command, where *lc* is a lower case letter.

REGULAR EXPRESSIONS

Regular expressions are patterns used in selecting text. For example, the **ed** command

```
g/string/
```

prints all lines containing *string*. Regular expressions are also used by the **s** command for selecting old text to be replaced with new.

In addition to specifying string literals, regular expressions can represent classes of strings. Strings thus represented are said to be matched by the corresponding regular expression. If it is possible for a regular expression to match several strings in a line, then the leftmost longest match is the one selected.

The following symbols are used in constructing regular expressions:

c Any character *c* not listed below, including '{', '}', '(', ')', '<', and '>' matches itself.

c Any backslash-escaped character *c*, except for '{', '}', '(', ')', '<', and '>' matches itself.

. Matches any single character.

[char-class]
Matches any single character in the character class *char-class*. See **CHARACTER CLASSES** below for further information.

[^char-class]
Matches any single character, other than newline, not in the character class *char-class*.

^ If ^ is the first character of a regular expression, then it anchors the regular expression to the beginning of a line. Otherwise, it matches itself.

$* If *$ is the last character of a regular expression, it anchors the regular expression to the end of a line. Otherwise, it matches itself.

\< Anchors the single character regular expression or subexpression immediately following it to the beginning of a word. (This may not be available.)

\> Anchors the single character regular expression or subexpression immediately following it to the end of a word. (This may not be available.)

\(*re*\) Defines a subexpression *re*. Subexpressions may be nested. A subsequent backreference of the form *n*, where *n* is a number in the range [1,9], expands to the text matched by the *n*th subexpression. For example, the regular expression \(.*\)\1 matches any string consisting of identical adjacent substrings. Subexpressions are ordered relative to their left delimiter.

* Matches the single character regular expression or subexpression immediately preceding it zero or more times. If * is the first character of a regular expression or subexpression, then it matches itself. The * operator sometimes yields unexpected results. For example, the regular expression *b** matches the beginning of the string *abbb* (as opposed to the substring *bbb*), since a null match is the only leftmost match.

\{n,m\} \{n,\} \{n\}
Matches the single character regular expression or subexpression immediately preceding it at least *n* and at most *m* times. If *m* is omitted, then it matches at least *n* times. If the comma is also omitted, then it matches exactly *n* times.

Additional regular expression operators may be defined depending on the particular regex(3) implementation.

CHARACTER CLASSES

A character class specifies a set of characters. It is written within square brackets ([]) and in its most basic form contains just the characters in the set.

To include a ']' in a character class, it must be the first character. A range of characters may be specified by separating the end characters of the range with a '-', e.g., 'a-z' specifies the lower case characters.

The following literals can also be used within character classes as shorthand for particular sets of characters:

[:alnum:]	Alphanumeric characters.
[:cntrl:]	Control characters.
[:lower:]	Lowercase alphabetic characters.
[:space:]	Whitespace (space, tab, newline, form feed, etc.)
[:alpha:]	Alphabetic characters.
[:digit:]	Numeric characters (digits).
[:print:]	Printable characters.
[:upper:]	Uppercase alphabetic characters.
[:blank:]	Blank characters (space and tab).
[:graph:]	Graphical characters (printing nonblank characters).
[:punct:]	Punctuation characters.
[:xdigit:]	Hexadecimal digits.

If '-' appears as the first or last character of a character class, then it matches itself. All other characters in a character class match themselves.

Patterns in a character class of the form *[.col-elm.]* or *[=col-elm=]* where *col-elm* is a *collating element* are interpreted according to `locale`(5) (not currently supported). See `regex`(3) for an explanation of these constructs.

COMMANDS

All **ed** commands are single characters, though some require additional parameters. If a command's parameters extend over several lines, then each line except for the last must be terminated with a backslash (`'\'`).

In general, at most one command is allowed per line. However, most commands accept a print suffix, which is any of **p** (print), **l** (list), or **n** (enumerate), to print the last line affected by the command.

An interrupt (typically ^C) has the effect of aborting the current command and returning the editor to command mode.

ed recognizes the following commands. The commands are shown together with the default address or address range supplied if none is specified (in parentheses), and other possible arguments on the right.

(.)**a** Appends text to the buffer after the addressed line. Text is entered in input mode. The current address is set to last line entered.

(.,.)**c** Changes lines in the buffer. The addressed lines are deleted from the buffer, and text is appended in their place. Text is entered in input mode. The current address is set to last line entered.

(.,.)**d** Deletes the addressed lines from the buffer. If there is a line after the deleted range, then the current address is set to this line. Otherwise the current address is set to the line before the deleted range.

e *file*
Edits *file*, and sets the default filename. If *file* is not specified, then the default filename is used. Any lines in the buffer are deleted before the new file is read. The current address is set to the last line read.

e *!command*
Edits the standard output of *command*, (see ! *command* below). The default filename is unchanged. Any lines in the buffer are deleted before the output of *command* is read. The current address is set to the last line read.

E *file*
Edits *file* unconditionally. This is similar to the **e** command, except that unwritten changes are discarded without warning. The current address is set to the last line read.

f *file*

Sets the default filename to *file*. If *file* is not specified, then the default unescaped filename is printed.

(1,$)**g**/*re*/*command-list*

Applies *command-list* to each of the addressed lines matching a regular expression *re*. The current address is set to the line currently matched before *command-list* is executed. At the end of the **g** command, the current address is set to the last line affected by *command-list*.

Each command in *command-list* must be on a separate line, and every line except for the last must be terminated by a backslash ('\'). Any commands are allowed, except for **g**, **G**, **v**, and **V**. A newline alone in *command-list* is equivalent to a **p** command.

(1,$)**G**/*re*/

Interactively edits the addressed lines matching a regular expression *re*. For each matching line, the line is printed, the current address is set, and the user is prompted to enter a *command-list*. At the end of the **G** command, the current address is set to the last line affected by (the last) *command-list*.

The format of *command-list* is the same as that of the **g** command. A newline alone acts as a null command list. A single '&' repeats the last non-null command list.

H Toggles the printing of error explanations. By default, explanations are not printed. It is recommended that **ed** scripts begin with this command to aid in debugging.

h Prints an explanation of the last error.

(.)**i** Inserts text in the buffer before the current line. Text is entered in input mode. The current address is set to the last line entered.

(.,.+1)**j**

Joins the addressed lines. The addressed lines are deleted from the buffer and replaced by a single line containing their joined text. The current address is set to the resultant line.

(.)**k***lc*

Marks a line with a lower case letter *lc*. The line can then be addressed as *'lc* (i.e., a single quote followed by *lc*) in subsequent commands. The mark is not cleared until the line is deleted or otherwise modified.

(.,.)**l** Prints the addressed lines unambiguously. If a single line fills more than one screen (as might be the case when viewing a binary file, for instance), a "--More--" prompt is printed on the last line. **ed** waits until the RETURN key is pressed before displaying the next screen. The current address is set to the last line printed.

(.,.)**m**(.)

Moves lines in the buffer. The addressed lines are moved to after the right-hand destination address, which may be the address *0* (zero). The current address is set to the last line moved.

(.,.)**n** Prints the addressed lines along with their line numbers. The current address is set to the last line printed.

(.,.)**p** Prints the addressed lines. The current address is set to the last line printed.

P Toggles the command prompt on and off. Unless a prompt was specified with the command-line option **-p** *string*, the command prompt is by default turned off.

q Quits **ed**.

Q Quits **ed** unconditionally. This is similar to the **q** command, except that unwritten changes are discarded without warning.

($)**r** *file*
Reads *file* to after the addressed line. If *file* is not specified, then the default filename is used. If there was no default filename prior to the command, then the default filename is set to *file*. Otherwise, the default filename is unchanged. The current address is set to the last line read.

($)**r** *!command*
Reads to after the addressed line the standard output of *command*, (see the ! command below). The default filename is unchanged. The current address is set to the last line read.

(.,.)**s***/re/replacement/*, *(.,.)***s***/re/replacement/g*, *(.,.)***s***/re/replacement/n*
Replaces text in the addressed lines matching a regular expression *re* with *replacement*. By default, only the first match in each line is replaced. If the *g* (global) suffix is given, then every match to be replaced. The *n* suffix, where *n* is a positive number, causes only the *n*th match to be replaced. It is an error if no substitutions are performed on any of the addressed lines. The current address is set the last line affected.

re and *replacement* may be delimited by any character other than space and newline (see the **s** command below). If one or two of the last delimiters is omitted, then the last line affected is printed as though the print suffix *p* were specified.

An unescaped '&' in *replacement* is replaced by the currently matched text. The character sequence *m*, where *m* is a number in the range [1,9], is replaced by the *m*th backreference expression of the matched text. If *replacement* consists of a single '%', then *replacement* from the last substitution is used. Newlines may be embedded in *replacement* if they are escaped with a backslash ('\').

(.,.)**s** Repeats the last substitution. This form of the **s** command accepts a count suffix *n*, or any combination of the characters *r*, *g*, and *p*. If a count suffix *n* is given, then only the *n*th match is replaced. The *r* suffix causes the regular expression of the last search to be used instead of that of the last substitution. The *g* suffix toggles the global suffix of the last substitution. The *p* suffix toggles the print suffix of the last substitution. The current address is set to the last line affected.

(.,.)**t**(.)
Copies (i.e., transfers) the addressed lines to after the right-hand destination address, which may be the address *0* (zero). The current address is set to the last line copied.

u Undoes the last command and restores the current address to what it was before the command. The global commands **g**, **G**, **v**, and **V** are treated as a single command by undo. **u** is its own inverse.

(1,$)**v***/re/command-list*
Applies *command-list* to each of the addressed lines not matching a regular expression *re*. This is similar to the **g** command.

(1,$)**V***/re/*
Interactively edits the addressed lines not matching a regular expression *re*. This is similar to the **G** command.

(1,$)**w** *file*
Writes the addressed lines to *file*. Any previous contents of *file* are lost without warning. If there is no default filename, then the default filename is set to *file*, otherwise it is unchanged. If no filename is specified, then the default filename is used. The current address is unchanged.

(1,$)**wq** *file*
Writes the addressed lines to *file*, and then executes a **q** command.

(1,$)**w** *!command*
Writes the addressed lines to the standard input of *command*, (see the ! command below). The default filename and current address are unchanged.

(1,$)**W** *file*
Appends the addressed lines to the end of *file*. This is similar to the **w** command, except that the previous contents of file are not clobbered. The current address is unchanged.

x Prompts for an encryption key which is used in subsequent reads and writes. If a newline alone is entered as the key, then encryption is turned off. Otherwise, echoing is disabled while a key is read. Encryption/decryption is done using the bdes(1) algorithm.

(.+1)**z***n*
Scrolls *n* lines at a time starting at addressed line. If *n* is not specified, then the current window size is used. The current address is set to the last line printed.

($)**=** Prints the line number of the addressed line.

(.+1)**newline**
Prints the addressed line, and sets the current address to that line.

!*command*
Executes *command* via sh(1). If the first character of *command* is !, then it is replaced by text of the previous **!command**. **ed** does not process *command* for '\' (backslash) escapes. However, an unescaped '%' is replaced by the default filename. When the shell returns from execution, a ''! is printed to the standard output. The current line is unchanged.

LIMITATIONS

ed processes *file* arguments for backslash escapes, i.e., in a filename, any characters preceded by a backslash ('\') are interpreted literally.

If a text (non-binary) file is not terminated by a newline character, then **ed** appends one on reading/writing it. In the case of a binary file, **ed** does not append a newline on reading/writing.

ENVIRONMENT

TMPDIR The location used to store temporary files.

FILES

/tmp/ed.* buffer file
ed.hup where **ed** attempts to write the buffer if the terminal hangs up

DIAGNOSTICS

When an error occurs, **ed** prints a ""? and either returns to command mode or exits if its input is from a script. An explanation of the last error can be printed with the **h** (help) command.

Since the **g** (global) command masks any errors from failed searches and substitutions, it can be used to perform conditional operations in scripts; e.g.,

```
g/old/s//new/
```

replaces any occurrences of *old* with *new*.

If the **u** (undo) command occurs in a global command list, then the command list is executed only once.

If diagnostics are not disabled, attempting to quit **ed** or edit another file before writing a modified buffer results in an error. If the command is entered a second time, it succeeds, but any changes to the buffer are lost.

SEE ALSO

`bdes`(1), `sed`(1), `sh`(1), `vi`(1), `regex`(3)

USD:09-10

B. W. Kernighan and P. J. Plauger, *Software Tools in Pascal*, *Addison-Wesley*, 1981.

HISTORY

An **ed** command appeared in Version 1 AT&T UNIX.

NAME

edahdi — modify AHDI partition identifiers

SYNOPSIS

edahdi *device*

DESCRIPTION

edahdi allows you to modify the partition identifiers on a disk partitioned with AHDI or an AHDI compatible formatter. An AHDI partition format is usually only present on disks shared between NetBSD and some other OS. The partition identifiers are used by NetBSD as a guideline to emulate a disklabel on such a disk.

edahdi supports the following options:

device The name of the raw device you want to edit.

The following partition identifiers are recognized by NetBSD:

NBD	Partition is reserved for NetBSD. This can be either a root or an user partition. The first NBD partition on a disk will be mapped to partition *a* in NetBSD. The following NBD partitions will be mapped from *d* up. The filesystem type is ffs by default.
SWP	The first SWP partition is mapped to partition *b*.
GEM or BGM	These partitions are mapped from *d* up. The filesystem type is msdos.
NBR	NetBSD root partition (deprecated).
NBU	NetBSD user partition (deprecated).
NBS	NetBSD swap partition (deprecated).

EXAMPLES

Say, you have a disk with that is partitioned like:

Number	Id
1	GEM
2	GEM
3	GEM
4	GEM

This partitioning will show up in NetBSD as (Number refers to the first table):

Partition	Fstype	Number
c (whole disk)	unused	
d (user part)	MSDOS	1
e (user part)	MSDOS	2
f (user part)	MSDOS	3
g (user part)	MSDOS	4

Now you decide to change the id of partition 2 and 3 to NBD. Now NetBSD will show the partitioning as (Number refers to the first table):

Partition	Fstype	Number
a (root)	4.2BSD	2
c (whole disk)	unused	
d (user part)	MSDOS	1
e (user part)	4.2BSD	3
f (user part)	MSDOS	4

You will notice that the order of the partitions has changed! You will have to watchout for this. It is a consequence of NetBSD habit of assigning a predefined meaning to the partitions *a/b* and *c*.

SEE ALSO

`disklabel`(8), `installboot`(8)

HISTORY

The **`edahdi`** command first appeared in NetBSD 1.2.

BUGS

The changes made to the AHDI partitions will become active on the next *first open* of the device. You are advised to use **`edahdi`** only on a device without any mounted or otherwise active partitions. This is not enforced by **`edahdi`**. This is particularly confusing when your change caused partitions to shift, as shown in the example above.

As soon as a disk contains at least one NBD partition, you are allowed to write disklabels and install bootstraps.

NAME

eject — eject a floppy disk, cdrom or tape

SYNOPSIS

eject [**-fv**] [**-l** | **-L** | **-U**] [**-t** *device-type*] [**-d**] *device*

eject -n

DESCRIPTION

The **eject** program ejects a medium from the specified device. It can also load a cdrom in the drive if this operation is supported by the hardware. The *device* argument specifies a device either by its full path name (identified by a /dev/ prefix), or by one of the built-in nicknames. If the medium contains a file system that is currently mounted, **eject** will attempt to unmount the file system before ejecting.

The following options are available:

-d Deprecated.

-f Force the eject operation without attempting to unmount any file systems first.

-l Load media in the drive (only supported for the cdrom device type).

-L Lock the media into the drive (but see **BUGS** below).

-n List the built-in nicknames on standard output.

-t *device-type*
Specify the device type. The argument must be one of **diskette**, **floppy**, **cdrom**, **disk**, or **tape**. This option is necessary when ejecting a device for which no built-in knowledge is available.

-U Unlock the media from the drive.

-v Display some of the actions taken on standard output.

BUGS

Most disk drivers automatically lock the media on the first open and unlock it on the last close, making **eject -L** almost useless, since when it closes the device, it gets unlocked again.

NAME

elf2aout — convert a NetBSD ELF-format executable to NetBSD a.out format

SYNOPSIS

elf2aout *elf-file aout-file*

DESCRIPTION

Reads a fully-linked ELF executable (such as a linked kernel) and produces an equivalent a.out format executable file.

The **elf2aout** utility is used to convert native NetBSD ELF binaries to a.out format, for compatibility with bootblocks and kernel-reading utilities like kvm(3) and kvm_mkdb(8), which currently expect an a.out format kernel.

SEE ALSO

elf2ecoff(1), ld(1), kvm(3), a.out(5), elf(5), kvm_mkdb(8)

HISTORY

elf2aout was originally developed for NetBSD/pmax by Ted Lemon and was first distributed with the pmax port of NetBSD 1.1.

BUGS

elf2aout assumes there are no multiply-referenced symbols in the input ELF symbol section. It may be necessary to link with **-x** to avoid such duplicate symbols.

In some environments, the GNU binutils objcopy(1) utility may be a better solution than **elf2aout**.

NAME

elf2ecoff — convert a NetBSD ELF-format executable to NetBSD ECOFF format

SYNOPSIS

elf2ecoff *elf-file ecoff-file*

DESCRIPTION

Reads a fully-linked ELF executable (such as a linked kernel) and produces an equivalent ECOFF format executable file.

The **elf2ecoff** utility is used to convert native NetBSD ELF binaries to ECOFF format, for compatibility with DECstation diskless-boot PROM code and diskless-boot servers that require ECOFF format binaries.

SEE ALSO

elf2aout(1), ld(1), kvm(3), a.out(5), elf(5), kvm_mkdb(8)

HISTORY

elf2ecoff was originally developed for NetBSD/pmax by Ted Lemon and was first distributed with the pmax port of NetBSD 1.1.

BUGS

elf2ecoff assumes there are no multiply-referenced symbols in the input ELF symbol section. It may be necessary to link with **-x** to avoid duplicate symbols.

In some environments, the GNU binutils objcopy(1) utility may be a better solution than **elf2ecoff**.

NAME

env — set and print environment

SYNOPSIS

env [**-i**] [*name=value . . .*] [*utility* [argument . . .]]

DESCRIPTION

env executes *utility* after modifying the environment as specified on the command line. The option *name=value* specifies an environmental variable, *name*, with a value of *value*. The option '**-i**' causes **env** to completely ignore the environment it inherits.

If no *utility* is specified, **env** prints out the names and values of the variables in the environment, with one *name=value* pair per line.

EXIT STATUS

env exits with one of the following values:

0 *utility* was invoked and completed successfully. In this case the exit code is returned by the utility itself, not **env**. If no utility was specified, then **env** completed successfully and returned the exit code itself.

1 An invalid command line option was passed to **env**.

1−125 *utility* was invoked, but failed in some way; see its manual page for more information. In this case the exit code is returned by the utility itself, not **env**.

126 *utility* was found, but could not be invoked.

127 *utility* could not be found.

COMPATIBILITY

The historic **-** option has been deprecated but is still supported in this implementation.

SEE ALSO

execvp(3), environ(7)

STANDARDS

The **env** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2").

BUGS

env doesn't handle commands with equal ("=") signs in their names, for obvious reasons.

NAME

envsubst − substitutes environment variables in shell format strings

SYNOPSIS

envsubst [*OPTION*] [*SHELL-FORMAT*]

DESCRIPTION

Substitutes the values of environment variables.

Operation mode:

−v, **−−variables**

output the variables occurring in SHELL-FORMAT

Informative output:

−h, **−−help**

display this help and exit

−V, **−−version**

output version information and exit

In normal operation mode, standard input is copied to standard output, with references to environment variables of the form $VARIABLE or ${VARIABLE} being replaced with the corresponding values. If a SHELL-FORMAT is given, only those environment variables that are referenced in SHELL-FORMAT are substituted; otherwise all environment variables references occurring in standard input are substituted.

When **−−variables** is used, standard input is ignored, and the output consists of the environment variables that are referenced in SHELL-FORMAT, one per line.

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **envsubst** is maintained as a Texinfo manual. If the **info** and **envsubst** programs are properly installed at your site, the command

info envsubst

should give you access to the complete manual.

NAME

error — analyze and disperse compiler error messages

SYNOPSIS

error [**-n**] [**-s**] [**-q**] [**-v**] [**-t** *suffixlist*] [**-I** *ignorefile*] [name]

DESCRIPTION

error analyzes and optionally disperses the diagnostic error messages produced by a number of compilers and language processors to the source file and line where the errors occurred. It can replace the painful, traditional methods of scribbling abbreviations of errors on paper, and permits error messages and source code to be viewed simultaneously without machinations of multiple windows in a screen editor.

Options are:

-n Do *not* touch any files; all error messages are sent to the standard output.

-q The user is *queried* whether s/he wants to touch the file. A ‘‘y’’ or ‘‘n’’ to the question is necessary to continue. Absence of the **-q** option implies that all referenced files (except those referring to discarded error messages) are to be touched.

-v After all files have been touched, overlay the visual editor vi(1) with it set up to edit all files touched, and positioned in the first touched file at the first error. If vi(1) can’t be found, try ex(1) or ed(1) from standard places.

-t Take the following argument as a suffix list. Files whose suffixes do not appear in the suffix list are not touched. The suffix list is dot separated, and ‘‘*’’ wildcards work. Thus the suffix list:

```
.c.y.foo*.h
```

allows **error** to touch files ending with ‘‘.c’’, ‘‘.y’’, ‘‘.foo*’’ and ‘‘.h’’.

-s Print out *statistics* regarding the error categorization. Not too useful.

error looks at the error messages, either from the specified file *name* or from the standard input, and attempts to determine which language processor produced each error message, determines the source file and line number to which the error message refers, determines if the error message is to be ignored or not, and inserts the (possibly slightly modified) error message into the source file as a comment on the line preceding to which the line the error message refers. Error messages which can’t be categorized by language processor or content are not inserted into any file, but are sent to the standard output. **error** touches source files only after all input has been read.

error is intended to be run with its standard input connected via a pipe to the error message source. Some language processors put error messages on their standard error file; others put their messages on the standard output. Hence, both error sources should be piped together into **error**. For example, when using the csh(1) syntax,

```
make -s lint | error -q -v
```

will analyze all the error messages produced by whatever programs make(1) runs when making lint.

error knows about the error messages produced by: make(1), cc(1), cpp(1), **ccom**, as(1), ld(1), lint(1), **pi**, **pc**, f77(1), and *DEC Western Research Modula-2*. **error** knows a standard format for error messages produced by the language processors, so is sensitive to changes in these formats. For all languages except *Pascal*, error messages are restricted to be on one line. Some error messages refer to more than one line in more than one files; **error** will duplicate the error message and insert it at all of the places referenced.

error will do one of six things with error messages.

synchronize Some language processors produce short errors describing which file it is processing. **error** uses these to determine the file name for languages that don't include the file name in each error message. These synchronization messages are consumed entirely by **error**.

discard Error messages from `lint`(1) that refer to one of the two `lint`(1) libraries, `/usr/libdata/lint/llib-lc` and `/usr/libdata/lint/llib-port` are discarded, to prevent accidentally touching these libraries. Again, these error messages are consumed entirely by **error**.

nullify Error messages from `lint`(1) can be nullified if they refer to a specific function, which is known to generate diagnostics which are not interesting. Nullified error messages are not inserted into the source file, but are written to the standard output. The names of functions to ignore are taken from either the file named `.errorrc` in the user's home directory, or from the file named by the **-I** option. If the file does not exist, no error messages are nullified. If the file does exist, there must be one function name per line.

not file specific Error messages that can't be intuited are grouped together, and written to the standard output before any files are touched. They will not be inserted into any source file.

file specific Error message that refer to a specific file, but to no specific line, are written to the standard output when that file is touched.

true errors Error messages that can be intuited are candidates for insertion into the file to which they refer.

Only true error messages are candidates for inserting into the file they refer to. Other error messages are consumed entirely by **error** or are written to the standard output. **error** inserts the error messages into the source file on the line preceding the line the language processor found in error. Each error message is turned into a one line comment for the language, and is internally flagged with the string ''###'' at the beginning of the error, and ''%%%'' at the end of the error. This makes pattern searching for errors easier with an editor, and allows the messages to be easily removed. In addition, each error message contains the source line number for the line the message refers to. A reasonably formatted source program can be recompiled with the error messages still in it, without having the error messages themselves cause future errors. For poorly formatted source programs in free format languages, such as C or Pascal, it is possible to insert a comment into another comment, which can wreak havoc with a future compilation. To avoid this, programs with comments and source on the same line should be formatted so that language statements appear before comments.

error catches interrupt and terminate signals, and if in the insertion phase, will orderly terminate what it is doing.

FILES

`~/.errorrc` function names to ignore for `lint`(1) error messages
`/dev/tty` user's teletype

HISTORY

The **error** command appeared in 4.0BSD.

AUTHORS

Robert Henry

BUGS

Opens the teletype directly to do user querying.

Source files with links make a new copy of the file with only one link to it.

Changing a language processor's format of error messages may cause **error** to not understand the error message.

error, since it is purely mechanical, will not filter out subsequent errors caused by 'floodgating' initiated by one syntactically trivial error. Humans are still much better at discarding these related errors.

Pascal error messages belong after the lines affected (error puts them before). The alignment of the '\' marking the point of error is also disturbed by **error**.

error was designed for work on CRT's at reasonably high speed. It is less pleasant on slow speed terminals, and has never been used on hardcopy terminals.

NAME

expand, **unexpand** — expand tabs to spaces, and vice versa

SYNOPSIS

expand [**-t** *tabstop*] [**-t** *tab1,tab2,...,tabn*] [*file ...*]
unexpand [**-a**] [**-t** *tabstop*] [*file ...*]

DESCRIPTION

expand processes the named files or the standard input writing the standard output with tabs changed into blanks. Backspace characters are preserved into the output and decrement the column count for tab calculations. **expand** is useful for pre-processing character files (before sorting, looking at specific columns, etc.) that contain tabs.

If a single *tabstop* argument is given, then tabs are set *tabstop* spaces apart instead of the default 8. If multiple tabstops are given then the tabs are set at those specific columns.

unexpand puts tabs back into the data from the standard input or the named files and writes the result on the standard output.

Option (with **unexpand** only):

-a By default, only leading blanks and tabs are reconverted to maximal strings of tabs. If the **-a** option is given, then tabs are inserted whenever they would compress the resultant file by replacing two or more characters.

HISTORY

The **expand** command appeared in 3.0 BSD.

NAME

expn − recursively expand mail aliases

SYNOPSIS

expn [*-a*] [*-v*] [*-w*] [*-d*] [*-1*] *user*[*@hostname*] [*user*[*@hostname*]]...

DESCRIPTION

expn will use the SMTP **expn** and **vrfy** commands to expand mail aliases. It will first look up the addresses you provide on the command line. If those expand into addresses on other systems, it will connect to the other systems and expand again. It will keep doing this until no further expansion is possible.

OPTIONS

The default output of **expn** can contain many lines which are not valid email addresses. With the *-aa* flag, only expansions that result in legal addresses are used. Since many mailing lists have an illegal address or two, the single *-a*, address, flag specifies that a few illegal addresses can be mixed into the results. More *-a* flags vary the ratio. Read the source to track down the formula. With the *-a* option, you should be able to construct a new mailing list out of an existing one.

If you wish to limit the number of levels deep that **expn** will recurse as it traces addresses, use the *-1* option. For each *-1* another level will be traversed. So, *-111* will traverse no more than three levels deep.

The normal mode of operation for **expn** is to do all of its work silently. The following options make it more verbose. It is not necessary to make it verbose to see what it is doing because as it works, it changes its **argv**[0] variable to reflect its current activity. To see how it is expanding things, the *-v*, verbose, flag will cause **expn** to show each address before and after translation as it works. The *-w*, watch, flag will cause **expn** to show you its conversations with the mail daemons. Finally, the *-d*, debug, flag will expose many of the inner workings so that it is possible to eliminate bugs.

ENVIRONMENT

No environment variables are used.

FILES

/tmp/expn$$ temporary file used as input to nslookup.

SEE ALSO

aliases(5), **sendmail**(8), **nslookup**(8), RFC 823, and RFC 1123.

BUGS

Not all mail daemons will implement **expn** or **vrfy**. It is not possible to verify addresses that are served by such daemons.

When attempting to connect to a system to verify an address, **expn** only tries one IP address. Most mail daemons will try harder.

It is assumed that you are running domain names and that the **nslookup**(8) program is available. If not, **expn** will not be able to verify many addresses. It will also pause for a long time unless you change the code where it says *$have_nslookup = 1* to read *$have_nslookup = 0*.

Lastly, **expn** does not handle every valid address. If you have an example, please submit a bug report.

CREDITS

In 1986 or so, Jon Broome wrote a program of the same name that did about the same thing. It has since suffered bit rot and Jon Broome has dropped off the face of the earth! (Jon, if you are out there, drop me a line)

AVAILABILITY

The latest version of **expn** is available through anonymous ftp at *ftp://ftp.idiom.com/pub/muir-programs/expn*.

AUTHOR

David Muir Sharnoff <*muir@idiom.com*>

NAME

expr — evaluate expression

SYNOPSIS

expr *expression*

DESCRIPTION

The **expr** utility evaluates *expression* and writes the result on standard output.

All operators are separate arguments to the **expr** utility. Characters special to the command interpreter must be escaped.

Operators are listed below in order of increasing precedence. Operators with equal precedence are grouped within { } symbols.

expr1 | *expr2*
: Returns the evaluation of *expr1* if it is neither an empty string nor zero; otherwise, returns the evaluation of *expr2*.

expr1 & *expr2*
: Returns the evaluation of *expr1* if neither expression evaluates to an empty string or zero; otherwise, returns zero.

expr1 {=, >, ≥, <, ≤, !=} *expr2*
: Returns the results of integer comparison if both arguments are integers; otherwise, returns the results of string comparison using the locale-specific collation sequence. The result of each comparison is 1 if the specified relation is true, or 0 if the relation is false.

expr1 {+, -} *expr2*
: Returns the results of addition or subtraction of integer-valued arguments.

expr1 {∗, /, %} *expr2*
: Returns the results of multiplication, integer division, or remainder of integer-valued arguments.

expr1 : *expr2*
: The ":" operator matches *expr1* against *expr2*, which must be a regular expression. The regular expression is anchored to the beginning of the string with an implicit "^".

  If the match succeeds and the pattern contains at least one regular expression subexpression "\(...\)", the string corresponding to "\1" is returned; otherwise the matching operator returns the number of characters matched. If the match fails and the pattern contains a regular expression subexpression the null string is returned; otherwise 0.

(*expr*)
: Parentheses are used for grouping in the usual manner.

Additionally, the following keywords are recognized:

length *expr*
: Returns the length of the specified string in bytes.

Operator precedence (from highest to lowest):

1. parentheses
2. length
3. ":"
4. "∗", "/", and "%"

5. “+” and “-”
6. compare operators
7. “&”
8. “|”

EXIT STATUS

The **expr** utility exits with one of the following values:

0 the expression is neither an empty string nor 0.
1 the expression is an empty string or 0.
2 the expression is invalid.
>2 an error occurred (such as memory allocation failure).

EXAMPLES

1. The following example adds one to the variable a.
```
a=‘expr $a + 1‘
```
2. The following example returns zero, due to deduction having higher precedence than ’&’ operator.
```
expr 1 ’&’ 1 - 1
```
3. The following example returns the filename portion of a pathname stored in variable a.
```
expr /$a : ’.*/\(.*\)’
```
4. The following example returns the number of characters in variable a.
```
expr $a : ’.*’
```

STANDARDS

The **expr** utility conforms to IEEE Std 1003.2 (“POSIX.2”). The *length* keyword is an extension for compatibility with GNU **expr**.

AUTHORS

Original implementation was written by J.T. Conklin ⟨jtc@NetBSD.org⟩. It was rewritten for NetBSD 1.6 by Jaromir Dolecek ⟨jdolecek@NetBSD.org⟩.

NOTES

The empty string “” cannot be matched with the intuitive:

```
expr ’’ : ’$’
```

The reason is that the returned number of matched characters (zero) is indistinguishable from a failed match, so this returns failure. To match the empty string, use something like:

```
expr x’’ : ’x$’
```

COMPATIBILITY

This implementation of **expr** internally uses 64 bit representation of integers and checks for over- and underflows. It also treats / (division mark) and option ’--’ correctly depending upon context.

expr on other systems (including NetBSD up to and including NetBSD 1.5) might not be so graceful. Arithmetic results might be arbitrarily limited on such systems, most commonly to 32 bit quantities. This means such **expr** can only process values between -2147483648 and +2147483647.

On other systems, **expr** might also not work correctly for regular expressions where either side contains single forward slash, like this:

```
expr / : ’.*/\(.*\)’
```

If this is the case, you might use // (double forward slash) to avoid confusion with the division operator:

```
expr "//$a" : '.*/\(.*\)'
```

According to IEEE Std 1003.2 ("POSIX.2"), **expr** has to recognize special option '--', treat it as an end of command line options and ignore it. Some **expr** implementations don't recognize it at all, others might ignore it even in cases where doing so results in syntax error. There should be same result for both following examples, but it might not always be:

1. expr -- : .
2. expr -- -- : .

Although NetBSD **expr** handles both cases correctly, you should not depend on this behavior for portability reasons and avoid passing bare '--' as first argument.

NAME

false — return false value

SYNOPSIS

false

DESCRIPTION

The **false** utility always exits with a nonzero exit code.

SEE ALSO

csh(1), sh(1), true(1)

STANDARDS

The **false** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

NAME

fdformat — format a floppy diskette

SYNOPSIS

fdformat [**-f** *device*] [**-t** *type*] [**-n**] [**-B** *nbps*] [**-S** *nspt*] [**-T** *ntrk*] [**-C** *ncyl*] [**-P** *stepspercyl*] [**-G** *gaplen*] [**-F** *fillbyte*] [**-X** *xfer_rate*] [**-I** *interleave*]

DESCRIPTION

The **fdformat** utility formats a floppy diskette. With no arguments, it formats the default floppy device with the default density parameters (as provided by the floppy diskette device driver).

As the each track of the floppy diskette is formatted, it is read to verify the format was successful.

The **fdformat** utility does not create a filesystem of any kind. Use tools like `newfs`(8), `newfs_msdos`(8), or `mformat`(1) (part of the pkgsrc/sysutils/mtools package) depending on what filesystem type you want to use on the floppy disk to do so.

Available command-line flags are:

-f *device*
: Format the floppy using *device* instead of the default `/dev/rfd0a`.

-t *type*
: Format the floppy using parameters for the diskette named *type* in `/etc/floppytab` instead of the device default parameters.

-n
: Do not verify each track as it is read.

-B *nbps*
: Set the number of bytes per sector for the formatted diskette.

-S *nspt*
: Set the number of sectors per track for the formatted diskette.

-T *ntrk*
: Set the number of tracks (heads) per cylinder for the formatted diskette.

-C *ncyl*
: Set the number of cylinders for the formatted diskette.

-P *stepspercyl*
: Set the number of motor steps per cylinder for the formatted diskette.

-G *gaplen*
: Set the sector gap length for the formatted diskette.

-F *fillbyte*
: Set the fill byte for the formatted diskette.

-X *xfer_rate*
: Set the bit transfer rate for the formatted diskette.

-I *interleave*
: Set the interleave factor for the formatted diskette.

SEE ALSO

`fdc`(4) (amiga, i386 and sparc ports)

HISTORY

The **fdformat** utility appeared in NetBSD 1.3.

BUGS

Some floppy drive units have physical diskette format sensors which automatically select a diskette density for reading. Such drives can format at alternate densities, but they cannot successfully verify the formatted

diskette except at the diskette’s normal density.

NAME

fgen — IEEE 1275 Open Firmware FCode Tokenizer

SYNOPSIS

fgen [**-d** *level*] [**-o** *outfile*] *infile*

DESCRIPTION

Reads Forth source and generates tokenized FCode object file.

AUTHORS

Written by Eduardo E. Horvath <eeh@one-o.com>

BUGS

String escape sequences are not recognized so things such as

```
" foo "(01 02) "n "
```

will result in the string

“foo "(01 02) "n”.

Hexadecimal numbers with dots in them such as `100.0000` are not parsed.

Permissions on the output file are often incorrect.

Output to the standard output device can cause problems.

NAME

file — determine file type

SYNOPSIS

file [**-bchikLnNprsvz**] [**-f** *namefile*] [**-F** *separator*] [**-m** *magicfiles*] *file*
file -C [**-m** *magicfile*]

DESCRIPTION

This manual page documents version 4.21 of the **file** command.

file tests each argument in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests, and language tests. The *first* test that succeeds causes the file type to be printed.

The type printed will usually contain one of the words *text* (the file contains only printing characters and a few common control characters and is probably safe to read on an ASCII terminal), *executable* (the file contains the result of compiling a program in a form understandable to some UNIX kernel or another), or *data* meaning anything else (data is usually 'binary' or non-printable). Exceptions are well-known file formats (core files, tar archives) that are known to contain binary data. When modifying the file /usr/share/misc/magic or the program itself, make sure to *preserve these keywords*. People depend on knowing that all the readable files in a directory have the word "text" printed. Don't do as Berkeley did and change "shell commands text" to "shell script". Note that the file /usr/share/misc/magic is built mechanically from a large number of small files in the subdirectory Magdir in the source distribution of this program.

The filesystem tests are based on examining the return from a stat(2) system call. The program checks to see if the file is empty, or if it's some sort of special file. Any known file types appropriate to the system you are running on (sockets, symbolic links, or named pipes (FIFOs) on those systems that implement them) are intuited if they are defined in the system header file <sys/stat.h>.

The magic number tests are used to check for files with data in particular fixed formats. The canonical example of this is a binary executable (compiled program) a.out file, whose format is defined in <elf.h>, <a.out.h> and possibly <exec.h> in the standard include directory. These files have a 'magic number' stored in a particular place near the beginning of the file that tells the UNIX operating system that the file is a binary executable, and which of several types thereof. The concept of a 'magic number' has been applied by extension to data files. Any file with some invariant identifier at a small fixed offset into the file can usually be described in this way. The information identifying these files is read from the compiled magic file /usr/share/misc/magic.mgc, or /usr/share/misc/magic if the compile file does not exist. In addition **file** will look in $HOME/.magic.mgc, or $HOME/.magic for magic entries.

If a file does not match any of the entries in the magic file, it is examined to see if it seems to be a text file. ASCII, ISO-8859-x, non-ISO 8-bit extended-ASCII character sets (such as those used on Macintosh and IBM PC systems), UTF-8-encoded Unicode, UTF-16-encoded Unicode, and EBCDIC character sets can be distinguished by the different ranges and sequences of bytes that constitute printable text in each set. If a file passes any of these tests, its character set is reported. ASCII, ISO-8859-x, UTF-8, and extended-ASCII files are identified as "text" because they will be mostly readable on nearly any terminal; UTF-16 and EBCDIC are only "character data" because, while they contain text, it is text that will require translation before it can be read. In addition, **file** will attempt to determine other characteristics of text-type files. If the lines of a file are terminated by CR, CRLF, or NEL, instead of the Unix-standard LF, this will be reported. Files that contain embedded escape sequences or overstriking will also be identified.

Once **file** has determined the character set used in a text-type file, it will attempt to determine in what language the file is written. The language tests look for particular strings (cf <names.h> that can appear anywhere in the first few blocks of a file. For example, the keyword *.br* indicates that the file is most likely a troff(1) input file, just as the keyword *struct* indicates a C program. These tests are less reliable than the

previous two groups, so they are performed last. The language test routines also test for some miscellany (such as `tar`(1) archives).

Any file that cannot be identified as having been written in any of the character sets listed above is simply said to be ‘‘data’’.

OPTIONS

`-b`, **`--brief`**
Do not prepend filenames to output lines (brief mode).

`-c`, **`--checking-printout`**
Cause a checking printout of the parsed form of the magic file. This is usually used in conjunction with the **`-m`** flag to debug a new magic file before installing it.

`-C`, **`--compile`**
Write a `magic.mgc` output file that contains a pre-parsed version of the magic file.

`-e`, **`--exclude`** *`testname`*
Exclude the test named in *`testname`* from the list of tests made to determine the file type. Valid test names are:

apptype
Check for `EMX` application type (only on EMX).

ascii
Check for various types of ascii files.

compress
Don’t look for, or inside compressed files.

elf
Don’t print elf details.

fortran
Don’t look for fortran sequences inside ascii files.

soft
Don’t consult magic files.

tar
Don’t examine tar files.

token
Don’t look for known tokens inside ascii files.

troff
Don’t look for troff sequences inside ascii files.

`-f`, **`--files-from`** *`namefile`*
Read the names of the files to be examined from *`namefile`* (one per line) before the argument list. Either *`namefile`* or at least one filename argument must be present; to test the standard input, use ‘-’ as a filename argument.

`-F`, **`--separator`** *`separator`*
Use the specified string as the separator between the filename and the file result returned. Defaults to ‘:’.

`-h`, **`--no-dereference`**
option causes symlinks not to be followed (on systems that support symbolic links). This is the default if the environment variable `POSIXLY_CORRECT` is not defined.

-i, **--mime**
Causes the file command to output mime type strings rather than the more traditional human readable ones. Thus it may say “text/plain; charset=us-ascii” rather than “ASCII text”. In order for this option to work, file changes the way it handles files recognized by the command itself (such as many of the text file types, directories etc), and makes use of an alternative “magic” file. (See “FILES” section, below).

-k, **--keep-going**
Don’t stop at the first match, keep going.

-L, **--dereference**
option causes symlinks to be followed, as the like-named option in `ls`(1) (on systems that support symbolic links). This is the default if the environment variable `POSIXLY_CORRECT` is defined.

-m, **--magic-file** *list*
Specify an alternate list of files containing magic numbers. This can be a single file, or a colon-separated list of files. If a compiled magic file is found alongside, it will be used instead. With the **-i -or -mime** option, the program adds “.mime” to each file name.

-n, **--no-buffer**
Force stdout to be flushed after checking each file. This is only useful if checking a list of files. It is intended to be used by programs that want filetype output from a pipe.

-N, **--no-pad**
Don’t pad filenames so that they align in the output.

-p, **--preserve-date**
On systems that support `utime`(2) or `utimes`(2), attempt to preserve the access time of files analyzed, to pretend that **file** never read them.

-r, **--raw**
Don’t translate unprintable characters to \ooo. Normally **file** translates unprintable characters to their octal representation.

-s, **--special-files**
Normally, **file** only attempts to read and determine the type of argument files which `stat`(2) reports are ordinary files. This prevents problems, because reading special files may have peculiar consequences. Specifying the **-s** option causes **file** to also read argument files which are block or character special files. This is useful for determining the filesystem types of the data in raw disk partitions, which are block special files. This option also causes **file** to disregard the file size as reported by `stat`(2) since on some systems it reports a zero size for raw disk partitions.

-v, **--version**
Print the version of the program and exit.

-z, **--uncompress**
Try to look inside compressed files.

-0, **--print0**
Output a null character ‘\0’ after the end of the filename. Nice to `cut`(1) the output. This does not affect the separator which is still printed.

--help
Print a help message and exit.

FILES

`/usr/share/misc/magic.mgc`	Default compiled list of magic numbers
`/usr/share/misc/magic`	Default list of magic numbers
`/usr/share/misc/magic.mime.mgc`	Default compiled list of magic numbers, used to output mime types when the **`-i`** option is specified.
`/usr/share/misc/magic.mime`	Default list of magic numbers, used to output mime types when the **`-i`** option is specified.

ENVIRONMENT

The environment variable `MAGIC` can be used to set the default magic number file name. If that variable is set, then **`file`** will not attempt to open `$HOME/.magic`. **`file`** adds ".mime" and/or ".mgc" to the value of this variable as appropriate. The environment variable `POSIXLY_CORRECT` controls (on systems that support symbolic links), if **`file`** will attempt to follow symlinks or not. If set, then **`file`** follows symlink, otherwise it does not. This is also controlled by the **`-L`** and **`-h`** options.

SEE ALSO

`magic`(5), `strings`(1), `od`(1), `hexdump`(1)

STANDARDS CONFORMANCE

This program is believed to exceed the System V Interface Definition of FILE(CMD), as near as one can determine from the vague language contained therein. Its behavior is mostly compatible with the System V program of the same name. This version knows more magic, however, so it will produce different (albeit more accurate) output in many cases.

The one significant difference between this version and System V is that this version treats any white space as a delimiter, so that spaces in pattern strings must be escaped. For example,

```
>10     string  language impress       (imPRESS data)
```

in an existing magic file would have to be changed to

```
>10     string  language\ impress      (imPRESS data)
```

In addition, in this version, if a pattern string contains a backslash, it must be escaped. For example

```
0       string          \begindata     Andrew Toolkit document
```

in an existing magic file would have to be changed to

```
0       string          \\begindata    Andrew Toolkit document
```

SunOS releases 3.2 and later from Sun Microsystems include a **`file`** command derived from the System V one, but with some extensions. My version differs from Sun's only in minor ways. It includes the extension of the '&' operator, used as, for example,

```
>16     long&0x7fffffff        >0              not stripped
```

MAGIC DIRECTORY

The magic file entries have been collected from various sources, mainly USENET, and contributed by various authors. Christos Zoulas (address below) will collect additional or corrected magic file entries. A consolidation of magic file entries will be distributed periodically.

The order of entries in the magic file is significant. Depending on what system you are using, the order that they are put together may be incorrect. If your old **`file`** command uses a magic file, keep the old magic file around for comparison purposes (rename it to `/usr/share/misc/magic.orig ).`

EXAMPLES

```
$ file file.c obj/file /dev/wd0a
file.c:   C program text
obj/file: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
          for NetBSD 4.99.12, dynamically linked (uses shared libs),
          not stripped
/dev/wd0a: block special (0/0)

# file -s /dev/rwd0[abe]
/dev/rwd0a: x86 boot sector, BSD disklabel
/dev/rwd0b: data
/dev/rwd0e: Unix Fast File system (little-endian), last mounted on
/usr, last written at Mon Feb 10 13:22:40 2003, clean flag 2, number
of blocks 28754208, number of data blocks 27812712, number of cylin-
der groups 3566, block size 8192, fragment size 1024, minimum per-
centage of free blocks 5, rotational delay 0ms, disk rotational
speed 60rps, TIME optimization
```

HISTORY

There has been a **file** command in every UNIX since at least Research Version 4 (man page dated November, 1973). The System V version introduced one significant major change: the external list of magic number types. This slowed the program down slightly but made it a lot more flexible.

This program, based on the System V version, was written by Ian Darwin <ian@darwinsys.com> without looking at anybody else's source code.

John Gilmore revised the code extensively, making it better than the first version. Geoff Collyer found several inadequacies and provided some magic file entries. Contributions by the '&' operator by Rob McMahon, cudcv@warwick.ac.uk, 1989.

Guy Harris, guy@netapp.com, made many changes from 1993 to the present.

Primary development and maintenance from 1990 to the present by Christos Zoulas (christos@astron.com).

Altered by Chris Lowth, chris@lowth.com, 2000: Handle the **-i** option to output mime type strings and using an alternative magic file and internal logic.

Altered by Eric Fischer (enf@pobox.com), July, 2000, to identify character codes and attempt to identify the languages of non-ASCII files.

The list of contributors to the "Magdir" directory (source for the /usr/share/misc/magic file) is too long to include here. You know who you are; thank you.

LEGAL NOTICE

The files tar.h and is_tar.c were written by John Gilmore from his public-domain tar(1) program, and are not covered by the above license.

BUGS

There must be a better way to automate the construction of the Magic file from all the glop in Magdir. What is it?

file uses several algorithms that favor speed over accuracy, thus it can be misled about the contents of text files.

The support for text files (primarily for programming languages) is simplistic, inefficient and requires recompilation to update. Their use of `ASCII TAB` as a field delimiter is ugly and makes it hard to edit the files, but is entrenched.

It might be advisable to allow upper-case letters in keywords for e.g., `troff`(1) commands vs man page macros. Regular expression support would make this easy.

The program doesn't grok `FORTRAN`. It should be able to figure `FORTRAN` by seeing some keywords which appear indented at the start of line. Regular expression support would make this easy.

The list of keywords in `ascmagic` probably belongs in the Magic file. This could be done by using some keyword like '*' for the offset value.

Complain about conflicts in the magic file entries. Make a rule that the magic entries sort based on file offset rather than position within the magic file?

The program should provide a way to give an estimate of "how good" a guess is. We end up removing guesses (e.g. "From " as first 5 chars of file) because they are not as good as other guesses (e.g. "Newsgroups:" versus "Return-Path:"). Still, if the others don't pan out, it should be possible to use the first guess.

This program is slower than some vendors' file commands. The new support for multiple character codes makes it even slower.

This manual page, and particularly this section, is too long.

AVAILABILITY

You can obtain the original author's latest version by anonymous FTP on `ftp.astron.com` in the directory `/pub/file/file-X.YZ.tar.gz`

NAME

find — walk a file hierarchy

SYNOPSIS

find [**-H** | **-L** | **-P**] [**-dEhsXx**] *file* [*file ...*] [*expression*]
find [**-H** | **-L** | **-P**] [**-dEhsXx**] **-f** *file* [*file ...*] [*expression*]

DESCRIPTION

find recursively descends the directory tree for each *file* listed, evaluating an *expression* (composed of the "primaries" and "operands" listed below) in terms of each file in the tree.

The options are as follows:

-H The **-H** option causes the file information and file type (see `stat`(2)), returned for each symbolic link encountered on the command line to be those of the file referenced by the link, not the link itself. If the referenced file does not exist, the file information and type will be for the link itself. File information of all symbolic links not on the command line is that of the link itself.

-L The **-L** option causes the file information and file type (see `stat`(2)) returned for each symbolic link to be those of the file referenced by the link, not the link itself. If the referenced file does not exist, the file information and type will be for the link itself.

-P The **-P** option causes the file information and file type (see `stat`(2)) returned for each symbolic link to be those of the link itself.

-d The **-d** option causes **find** to perform a depth-first traversal, i.e., directories are visited in post-order and all entries in a directory will be acted on before the directory itself. By default, **find** visits directories in pre-order, i.e., before their contents. Note, the default is *not* a breadth-first traversal.

-E The **-E** option causes *regexp* arguments to primaries to be interpreted as extended regular expressions (see `re_format`(7)).

-f The **-f** option specifies a file hierarchy for **find** to traverse. File hierarchies may also be specified as the operands immediately following the options.

-h The **-h** option causes the file information and file type (see `stat`(2)), returned for each symbolic link to be those of the file referenced by the link, not the link itself. If the referenced file does not exist, the file information and type will be for the link itself.

-s The **-s** option causes the entries of each directory to be sorted in lexicographical order. Note that the sorting is done only inside of each directory; files in different directories are not sorted. Therefore, '`a/b`' appears before '`a.b`', which is different from "`find ... | sort`" order.

-X The **-X** option is a modification to permit **find** to be safely used in conjunction with `xargs`(1). If a file name contains any of the delimiting characters used by **xargs**, a diagnostic message is displayed on standard error, and the file is skipped. The delimiting characters include single ("'") and double ("") quotes, backslash ("\"), space, tab and newline characters. Alternatively, the **-print0** or **-printx** primaries can be used to format the output in a way that **xargs** can accept.

-x The **-x** option restricts the search to the file system containing the directory specified. Does not list mount points to other file systems.

PRIMARIES

-amin *n*

True if the difference between the file last access time and the time **find** was started, rounded up to the next full minute, is *n* minutes.

-anewer *file*
True if the current file has a more recent last access time than *file*.

-atime *n*
True if the difference between the file last access time and the time **find** was started, rounded up to the next full 24-hour period, is *n* 24-hour periods.

-cmin *n*
True if the difference between the time of last change of file status information and the time **find** was started, rounded up to the next full minute, is *n* minutes.

-cnewer *file*
True if the current file has a more recent last change time than *file*.

-ctime *n*
True if the difference between the time of last change of file status information and the time **find** was started, rounded up to the next full 24-hour period, is *n* 24-hour periods.

-delete
Delete found files and/or directories. Always returns True. This executes from the current working directory as **find** recurses down the tree. It will not attempt to delete a filename with a "" character in its pathname relative to "". for security reasons. Depth-first traversal processing is implied by this option. This can also be invoked as **-rm**.

-empty
True if the current file or directory is empty.

-exec *utility* [argument ...] ;
-exec *utility* [argument ...] {} +
Execute the specified *utility* with the specified arguments. The list of arguments is terminated by ";" or "+". *utility* will be executed from the directory from which **find** was executed.

If terminated by a semicolon (";"), the *utility* is invoked once per path. If the string "{}" appears anywhere in the utility name or the arguments, it is replaced by the pathname of the current file.

If terminated by a plus sign ("+"), the pathnames for which the primary is evaluated are aggregated into sets, and *utility* will be invoked once per set, similar to xargs(1). If any invocation exits with non-zero exit status, then **find** will eventually do so as well, but this does not cause **find** to exit early. The string "{}" must appear, and must appear last. Each set is limitted to no more than 5,000 pathnames, and is also limitted such that the invokation of *utility* does not exceed ARG_MAX.

-execdir *utility* [argument ...] ;
The **-execdir** primary is similar to the semicolon-terminated (";") variant of the **-exec** primary, with the exception that *utility* will be executed from the directory that holds the current file. The filename substituted for the string "{}" is not qualified. Set aggregation ("+" termination) is not supported.

-exit [*n*]
This primary causes **find** to stop traversing the filesystem and exit immediately if a previous condition was met. If no value is specified, the exit value will be 0, else *n*. Note that other primaries will be evaluated and acted upon before exiting.

-false
This primary always evaluates to false. This can be used following a primary that caused the expression to be true to make the expression to be false. This can be useful after using a **-fprint** primary so it can continue to the next expression (using an **-or** operator, for example).

-flags [**-**]*flags*
If *flags* are preceded by a dash ("-"), this primary evaluates to true if at least all of the bits in *flags* are set in the file's flags bits. If *flags* are not preceded by a dash, this primary evaluates to true if the bits in *flags* exactly match the file's flags bits. If *flags* is "none", files with no flags bits set are matched. (See chflags(1) for more information about file flags.)

-follow
Follow symbolic links.

-fprint *filename*
This primary always evaluates to true. This creates *filename* or overwrites the file if it already exists. The file is created at startup. It writes the pathname of the current file to this file, followed by a newline character. The file will be empty if no files are matched.

-fstype *type*
True if the file is contained in a file system of type *type*. The sysctl(8) command can be used to find out the types of filesystems that are available on the system:

```
sysctl vfs.generic.fstypes
```

In addition, there are two pseudo-types, "local" and "rdonly". The former matches any file system physically mounted on the system where the **find** is being executed, and the latter matches any file system which is mounted read-only.

-group *gname*
True if the file belongs to the group *gname*. If *gname* is numeric and there is no such group name, then *gname* is treated as a group id.

-iname *pattern*
True if the last component of the pathname being examined matches *pattern*. Case insensitive.

-inum *n*
True if the file has inode number *n*.

-iregex *regexp*
True if the path name of the current file matches the case-insensitive basic regular expression (see re_format(7)) *regexp*. This is a match on the whole path, not a search for the regular expression within the path.

-links *n*
True if the file has *n* links.

-rm This is an alias for **-delete**.

-ls This primary always evaluates to true. The following information for the current file is written to standard output: its inode number, size in 512-byte blocks, file permissions, number of hard links, owner, group, size in bytes, last modification time, and pathname. If the file is a block or character special file, the major and minor numbers will be displayed instead of the size in bytes. If the file is a symbolic link, the pathname of the linked-to file will be displayed preceded by "->". The format is identical to that produced by "ls -dgils".

-maxdepth *n*
True if the current search depth is less than or equal to what is specified in *n*.

-mindepth *n*
True if the current search depth is at least what is specified in *n*.

-mmin *n*
True if the difference between the file last modification time and the time **find** was started, rounded up to the next full minute, is *n* minutes.

-mtime *n*
True if the difference between the file last modification time and the time **find** was started, rounded up to the next full 24-hour period, is *n* 24-hour periods.

-ok *utility* [argument ...] ;
The **-ok** primary is similar to the semicolon-terminated (";") variant of the **-exec** primary, with the exception that **find** requests user affirmation for the execution of the utility by printing a message to the terminal and reading a response. If the response is other than "y", the command is not executed and the *-ok* primary evaluates to false. Set aggregation ("+" termination) is not supported.

-name *pattern*
True if the last component of the pathname being examined matches *pattern*. Special shell pattern matching characters ("[", "]", "*", "?") may be used as part of *pattern*. These characters may be matched explicitly by escaping them with a backslash ("\").

-newer *file*
True if the current file has a more recent last modification time than *file*.

-nouser
True if the file belongs to an unknown user.

-nogroup
True if the file belongs to an unknown group.

-path *pattern*
True if the pathname being examined matches *pattern*. Special shell pattern matching characters ("[", "]", "*", and "?") may be used as part of *pattern*. These characters may be matched explicitly by escaping them with a backslash ("\"). Slashes ("/") are treated as normal characters and do not have to be matched explicitly.

-perm [**-**]*mode*
The *mode* may be either symbolic (see chmod(1)) or an octal number. If the mode is symbolic, a starting value of zero is assumed and the mode sets or clears permissions without regard to the process' file mode creation mask. If the mode is octal, only bits 07777 (S_ISUID | S_ISGID | S_ISTXT | S_IRWXU | S_IRWXG | S_IRWXO) of the file's mode bits participate in the comparison. If the mode is preceded by a dash ("-"), this primary evaluates to true if at least all of the bits in the mode are set in the file's mode bits. If the mode is not preceded by a dash, this primary evaluates to true if the bits in the mode exactly match the file's mode bits. Note, the first character of a symbolic mode may not be a dash ("-").

-print
This primary always evaluates to true. It prints the pathname of the current file to standard output, followed by a newline character. If none of **-exec**, **-exit**, **-fprint**, **-ls**, **-ok**, **-print0**, nor **-printx** is specified, the given expression shall be effectively replaced by **(***given expression***) -print**.

-print0
This primary always evaluates to true. It prints the pathname of the current file to standard output, followed by a null character.

-printx
This primary always evaluates to true. It prints the pathname of the current file to standard output, with each space, tab, newline, backslash, dollar sign, and single, double, or back quotation mark prefixed by a backslash, so the output of **find** can safely be used as input to **xargs**.

-prune
This primary always evaluates to true. It causes **find** to not descend into the current file. Note, the **-prune** primary has no effect if the **-d** option was specified.

-regex *regexp*
True if the path name of the current file matches the case-sensitive basic regular expression (see re_format(7)) *regexp*. This is a match on the whole path, not a search for the regular expression within the path.

-size *n*[**c**]
True if the file's size, rounded up, in 512-byte blocks is *n*. If *n* is followed by a "c", then the primary is true if the file's size is *n* bytes.

-type *t*
True if the file is of the specified type. Possible file types are as follows:

b	block special
c	character special
d	directory
f	regular file
l	symbolic link
p	FIFO
s	socket
W	whiteout
w	whiteout

-user *uname*
True if the file belongs to the user *uname*. If *uname* is numeric and there is no such user name, then *uname* is treated as a user id (and considered a numeric argument).

-xdev This primary always evaluates to true. It causes find not to descend past directories that have a different device ID (st_dev, see stat(2) S5.6.2 [POSIX.1]).

All primaries which take a numeric argument allow the number to be preceded by a plus sign ("+") or a minus sign ("−"). A preceding plus sign means "more than n", a preceding minus sign means "less than n", and neither means "exactly n".

OPERATORS

The primaries may be combined using the following operators. The operators are listed in order of decreasing precedence.

(*expression* **)**
This evaluates to true if the parenthesized expression evaluates to true.

! *expression* This is the unary NOT operator. It evaluates to true if the expression is false.

expression **-and** *expression*

expression expression
The **-and** operator is the logical AND operator. As it is implied by the juxtaposition of two expressions it does not have to be specified. The expression evaluates to true if both expressions are true. The second expression is not evaluated if the first expression is false.

expression **-or** *expression*
The **-or** operator is the logical OR operator. The expression evaluates to true if either the first or the second expression is true. The second expression is not evaluated if the first expression is true.

All operands and primaries must be separate arguments to **find**. Primaries which themselves take arguments expect each argument to be a separate argument to **find**.

EXIT STATUS

The **find** utility normally exits 0 on success, and exits with 1 under certain internal error conditions. If any invokations of "**-exec** ... +" primaries return non-zero exit-status, then **find** will do so as well.

EXAMPLES

The following examples are shown as given to the shell:

`find / \! -name "*.c" -print`
Print out a list of all the files whose names do not end in ".c".

`find / -newer ttt -user wnj -print`
Print out a list of all the files owned by user "wnj" that are newer than the file "ttt".

`find / \! \( -newer ttt -user wnj \) -print`
Print out a list of all the files which are not both newer than "ttt" and owned by "wnj".

`find / \( -newer ttt -or -user wnj \) -print`
Print out a list of all the files that are either owned by "wnj" or that are newer than "ttt".

`find / \( -newer ttt -or -user wnj \) -exit 1`
Return immediately with a value of 1 if any files are found that are either owned by "wnj" or that are newer than "ttt", but do not print them.

`find / \( -newer ttt -or -user wnj \) -ls -exit 1`
Same as above, but list the first file matching the criteria before exiting with a value of 1.

SEE ALSO

`chflags`(1), `chmod`(1), `locate`(1), `xargs`(1), `stat`(2), `fts`(3), `getgrent`(3), `getpwent`(3), `strmode`(3), `symlink`(7), `sysctl`(8)

STANDARDS

The **find** utility syntax is a superset of the syntax specified by the IEEE Std 1003.2 ("POSIX.2") standard.

The options and the **-amin**, **-anewer**, **-cmin**, **-cnewer**, **-delete**, **-empty**, **-execdir**, **-follow**, **-fstype**, **-iname**, **-inum**, **-iregex**, **-links**, **-ls**, **-maxdepth**, **-mindepth**, **-mmin**, **-path**, **-print0**, **-printx**, **-regex**, and **-rm** primaries are extensions to IEEE Std 1003.2 ("POSIX.2").

Historically, the **-d**, **-h**, and **-x** options were implemented using the primaries "-depth", "-follow", and "-xdev". These primaries always evaluated to true. As they were really global variables that took effect before the traversal began, some legal expressions could have unexpected results. An example is the expression "-print -o -depth". As -print always evaluates to true, the standard order of evaluation implies that -depth would never be evaluated. This is not the case.

The operator "-or" was implemented as "-o", and the operator "-and" was implemented as "-a".

Historic implementations of the **-exec** and **-ok** primaries did not replace the string "{}" in the utility name or the utility arguments if it had preceding or following non-whitespace characters. This version replaces it no matter where in the utility name or arguments it appears.

Support for "**-exec** **...** +" is consistent with *IEEE PASC Interpretation 1003.2 #210*, though the feature originated in SVR4.

The **-delete** primary does not interact well with other options that cause the filesystem tree traversal options to be changed.

HISTORY

A much simpler **find** command appeared in First Edition AT&T Unix. The syntax had become similar to the present version by the time of the Fifth Edition.

BUGS

The special characters used by **find** are also special characters to many shell programs. In particular, the characters "*", "[", "]", "?", "(", ")", "!", "\", and ";" may have to be escaped from the shell.

As there is no delimiter separating options and file names or file names and the *expression*, it is difficult to specify files named "-xdev" or "!". These problems are handled by the **-f** option and the getopt(3) "--" construct.

NAME

finger — user information lookup program

SYNOPSIS

finger [**-8ghlmops**] [*user ...*] [*user@host ...*]

DESCRIPTION

The **finger** displays information about the system users.

Options are:

-8 Pass through 8-bit data. This option is intended for enabling 8-bit data output in the `fingerd`(8) service. Using this from the command line is *dangerous*, as the output data may include control characters for your terminal.

-g This option restricts the gecos output to only the users' real names.

-h When used in conjunction with the **-s** option, the name of the remote host is displayed instead of the office location and office phone.

-l Produces a multi-line format displaying all of the information described for the **-s** option as well as the user's home directory, home phone number, login shell, mail status, and the contents of the files "`.forward`", "`.plan`" and "`.project`" from the user's home directory.

If idle time is at least a minute and less than a day, it is presented in the form "hh:mm". Idle times greater than a day are presented as "d day[s]hh:mm".

Phone numbers specified as eleven digits are printed as "+N-NNN-NNN-NNNN". Numbers specified as ten or seven digits are printed as the appropriate subset of that string. Numbers specified as five digits are printed as "xN-NNNN". Numbers specified as four digits are printed as "xNNNN".

If write permission is denied to the device, the phrase "(messages off)" is appended to the line containing the device name. One entry per user is displayed with the **-l** option; if a user is logged on multiple times, terminal information is repeated once per login.

Mail status is shown as "No Mail." if there is no mail at all, ''Mail last read DDD MMM ## HH:MM YYYY (TZ)'' if the person has looked at their mailbox since new mail arriving, or "New mail received ...", "Unread since ..." if they have new mail.

-m Prevent matching of *user* names. *User* is usually a login name; however, matching will also be done on the users' real names, unless the **-m** option is supplied. All name matching performed by **finger** is case insensitive.

-o When used in conjunction with the **-s** option, the office location and office phone information is displayed instead of the name of the remote host.

-p Prevents the **-l** option of **finger** from displaying the contents of the "`.forward`", "`.plan`" and "`.project`" files.

-s **finger** displays the user's login name, real name, terminal name and write status (as a "*" after the terminal name if write permission is denied), idle time, login time, and either office location and office phone number, or the remote host. If **-h** is given, the remote is printed. If **-o** is given, the office location and phone number is printed instead (the default).

Idle time is in minutes if it is a single integer, hours and minutes if a ":" is present, or days if a "d" is present. Login time is displayed as the dayname if less than six days, else month, day, hours and minutes, unless more than six months ago, in which case the year is displayed rather than the hours and minutes.

Unknown devices as well as nonexistent idle and login times are displayed as single asterisks.

If no options are specified, **finger** defaults to the **-l** style output if operands are provided, otherwise to the **-s** style. Note that some fields may be missing, in either format, if information is not available for them.

If no arguments are specified, **finger** will print an entry for each user currently logged into the system.

finger may be used to look up users on a remote machine. The format is to specify a *user* as "user@host", or "@host", where the default output format for the former is the **-l** style, and the default output format for the latter is the **-s** style. The **-l** option is the only option that may be passed to a remote machine.

FILES

/var/log/lastlog last login data base

SEE ALSO

chpass(1), w(1), who(1)

HISTORY

The **finger** command appeared in 3.0 BSD.

NAME

flex, lex − fast lexical analyzer generator

SYNOPSIS

flex [−bcdfhilnpstvwBFILTV78+? −C[aefFmr] −ooutput −Pprefix −Sskeleton] [−−help −−version] *[filename ...]*

OVERVIEW

This manual describes *flex*, a tool for generating programs that perform pattern-matching on text. The manual includes both tutorial and reference sections:

Description
a brief overview of the tool

Some Simple Examples

Format Of The Input File

Patterns
the extended regular expressions used by flex

How The Input Is Matched
the rules for determining what has been matched

Actions
how to specify what to do when a pattern is matched

The Generated Scanner
details regarding the scanner that flex produces;
how to control the input source

Start Conditions
introducing context into your scanners, and
managing "mini-scanners"

Multiple Input Buffers
how to manipulate multiple input sources; how to
scan from strings instead of files

End-of-file Rules
special rules for matching the end of the input

Miscellaneous Macros
a summary of macros available to the actions

Values Available To The User
a summary of values available to the actions

Interfacing With Yacc
connecting flex scanners together with yacc parsers

Options
flex command-line options, and the "%option"
directive

Performance Considerations

DESCRIPTION

flex is a tool for generating *scanners:* programs which recognized lexical patterns in text. *flex* reads the given input files, or its standard input if no file names are given, for a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, called *rules. flex* generates as output a C source file, **lex.yy.c,** which defines a routine **yylex().** This file is compiled and linked with the **−lfl** library to produce an executable. When the executable is run, it analyzes its input for occurrences of the regular expressions. Whenever it finds one, it executes the corresponding C code.

SOME SIMPLE EXAMPLES

First some simple examples to get the flavor of how one uses *flex.* The following *flex* input specifies a scanner which whenever it encounters the string "username" will replace it with the user's login name:

```
%%
username    printf( "%s", getlogin() );
```

By default, any text not matched by a *flex* scanner is copied to the output, so the net effect of this scanner is to copy its input file to its output with each occurrence of "username" expanded. In this input, there is just one rule. "username" is the *pattern* and the "printf" is the *action.* The "%%" marks the beginning of the rules.

Here's another simple example:

```
    int num_lines = 0, num_chars = 0;

%%
\n      ++num_lines; ++num_chars;
.       ++num_chars;

%%
```

```
main()
        {
        yylex();
        printf( "# of lines = %d, # of chars = %d\n",
                num_lines, num_chars );
        }
```

This scanner counts the number of characters and the number of lines in its input (it produces no output other than the final report on the counts). The first line declares two globals, "num_lines" and "num_chars", which are accessible both inside **yylex()** and in the **main()** routine declared after the second "%%". There are two rules, one which matches a newline ("\n") and increments both the line count and the character count, and one which matches any character other than a newline (indicated by the "." regular expression).

A somewhat more complicated example:

```
/* scanner for a toy Pascal-like language */

%{
/* need this for the call to atof() below */
#include <math.h>
%}

DIGIT    [0-9]
ID       [a-z][a-z0-9]*

%%

{DIGIT}+    {
            printf( "An integer: %s (%d)\n", yytext,
                    atoi( yytext ) );
            }

{DIGIT}+"."{DIGIT}*        {
            printf( "A float: %s (%g)\n", yytext,
                    atof( yytext ) );
            }

if|then|begin|end|procedure|function        {
            printf( "A keyword: %s\n", yytext );
            }

{ID}        printf( "An identifier: %s\n", yytext );

"+"|"-"|"*"|"/"   printf( "An operator: %s\n", yytext );

"{"[^}\n]*"}"     /* eat up one-line comments */

[ \t\n]+          /* eat up whitespace */

.           printf( "Unrecognized character: %s\n", yytext );

%%

main( argc, argv )
```

```
int argc;
char **argv;
   {
   ++argv, --argc;  /* skip over program name */
   if ( argc > 0 )
         yyin = fopen( argv[0], "r" );
   else
         yyin = stdin;

   yylex();
   }
```

This is the beginnings of a simple scanner for a language like Pascal. It identifies different types of *tokens* and reports on what it has seen.

The details of this example will be explained in the following sections.

FORMAT OF THE INPUT FILE

The *flex* input file consists of three sections, separated by a line with just **%%** in it:

```
definitions
%%
rules
%%
user code
```

The *definitions* section contains declarations of simple *name* definitions to simplify the scanner specification, and declarations of *start conditions,* which are explained in a later section.

Name definitions have the form:

```
name definition
```

The "name" is a word beginning with a letter or an underscore (’_’) followed by zero or more letters, digits, ’_’, or ’-’ (dash). The definition is taken to begin at the first non-white-space character following the name and continuing to the end of the line. The definition can subsequently be referred to using "{name}", which will expand to "(definition)". For example,

```
DIGIT    [0-9]
ID       [a-z][a-z0-9]*
```

defines "DIGIT" to be a regular expression which matches a single digit, and "ID" to be a regular expression which matches a letter followed by zero-or-more letters-or-digits. A subsequent reference to

```
{DIGIT}+"."{DIGIT}*
```

is identical to

```
([0-9])+"."([0-9])*
```

and matches one-or-more digits followed by a ’.’ followed by zero-or-more digits.

The *rules* section of the *flex* input contains a series of rules of the form:

```
pattern   action
```

where the pattern must be unindented and the action must begin on the same line.

See below for a further description of patterns and actions.

Finally, the user code section is simply copied to **lex.yy.c** verbatim. It is used for companion routines which call or are called by the scanner. The presence of this section is optional; if it is missing, the second **%%** in the input file may be skipped, too.

In the definitions and rules sections, any *indented* text or text enclosed in **%{** and **%}** is copied verbatim to the output (with the %{}'s removed). The %{}'s must appear unindented on lines by themselves.

In the rules section, any indented or %{} text appearing before the first rule may be used to declare variables which are local to the scanning routine and (after the declarations) code which is to be executed whenever the scanning routine is entered. Other indented or %{} text in the rule section is still copied to the output, but its meaning is not well-defined and it may well cause compile-time errors (this feature is present for *POSIX* compliance; see below for other such features).

In the definitions section (but not in the rules section), an unindented comment (i.e., a line beginning with "/*") is also copied verbatim to the output up to the next "*/".

PATTERNS

The patterns in the input are written using an extended set of regular expressions. These are:

```
x         match the character 'x'
.         any character (byte) except newline
[xyz]     a "character class"; in this case, the pattern
          matches either an 'x', a 'y', or a 'z'
[abj-oZ]  a "character class" with a range in it; matches
          an 'a', a 'b', any letter from 'j' through 'o',
          or a 'Z'
[^A-Z]    a "negated character class", i.e., any character
          but those in the class.  In this case, any
          character EXCEPT an uppercase letter.
[^A-Z\n]  any character EXCEPT an uppercase letter or
          a newline
r*        zero or more r's, where r is any regular expression
r+        one or more r's
r?        zero or one r's (that is, "an optional r")
r{2,5}    anywhere from two to five r's
r{2,}     two or more r's
r{4}      exactly 4 r's
{name}    the expansion of the "name" definition
          (see above)
"[xyz]\"foo"
          the literal string: [xyz]"foo
\X        if X is an 'a', 'b', 'f', 'n', 'r', 't', or 'v',
          then the ANSI-C interpretation of \x.
          Otherwise, a literal 'X' (used to escape
          operators such as '*')
\0        a NUL character (ASCII code 0)
\123      the character with octal value 123
\x2a      the character with hexadecimal value 2a
(r)       match an r; parentheses are used to override
          precedence (see below)


rs        the regular expression r followed by the
          regular expression s; called "concatenation"
```

r|s either an r or an s

r/s an r but only if it is followed by an s. The text matched by s is included when determining whether this rule is the "longest match", but is then returned to the input before the action is executed. So the action only sees the text matched by r. This type of pattern is called trailing context". (There are some combinations of r/s that flex cannot match correctly; see notes in the Deficiencies / Bugs section below regarding "dangerous trailing context".)

^r an r, but only at the beginning of a line (i.e., which just starting to scan, or right after a newline has been scanned).

r$ an r, but only at the end of a line (i.e., just before a newline). Equivalent to "r/\n".

Note that flex's notion of "newline" is exactly whatever the C compiler used to compile flex interprets '\n' as; in particular, on some DOS systems you must either filter out \r's in the input yourself, or explicitly use r/\r\n for "r$".

<s>r an r, but only in start condition s (see below for discussion of start conditions)

<s1,s2,s3>r same, but in any of start conditions s1, s2, or s3

<*>r an r in any start condition, even an exclusive one.

<<EOF>> an end-of-file

<s1,s2><<EOF>> an end-of-file when in start condition s1 or s2

Note that inside of a character class, all regular expression operators lose their special meaning except escape ('\') and the character class operators, '-', ']', and, at the beginning of the class, '^'.

The regular expressions listed above are grouped according to precedence, from highest precedence at the top to lowest at the bottom. Those grouped together have equal precedence. For example,

```
foo|bar*
```

is the same as

```
(foo)|(ba(r*))
```

since the '*' operator has higher precedence than concatenation, and concatenation higher than alternation ('|'). This pattern therefore matches *either* the string "foo" *or* the string "ba" followed by zero-or-more r's. To match "foo" or zero-or-more "bar"'s, use:

```
foo|(bar)*
```

and to match zero-or-more "foo"'s-or-"bar"'s:

```
(foo|bar)*
```

In addition to characters and ranges of characters, character classes can also contain character class *expressions.* These are expressions enclosed inside **[:** and **:]** delimiters (which themselves must appear between the '[' and ']' of the character class; other elements may occur inside the character class, too). The valid expressions are:

```
[:alnum:] [:alpha:] [:blank:]
[:cntrl:] [:digit:] [:graph:]
[:lower:] [:print:] [:punct:]
[:space:] [:upper:] [:xdigit:]
```

These expressions all designate a set of characters equivalent to the corresponding standard C **isXXX** function. For example, **[:alnum:]** designates those characters for which **isalnum()** returns true - i.e., any alphabetic or numeric. Some systems don't provide **isblank(),** so flex defines **[:blank:]** as a blank or a tab.

For example, the following character classes are all equivalent:

```
[[:alnum:]]
[[:alpha:][:digit:]]
[[:alpha:]0-9]
[a-zA-Z0-9]
```

If your scanner is case-insensitive (the **−i** flag), then **[:upper:]** and **[:lower:]** are equivalent to **[:alpha:].**

Some notes on patterns:

- A negated character class such as the example "[^A-Z]" above *will match a newline* unless "\n" (or an equivalent escape sequence) is one of the characters explicitly present in the negated character class (e.g., "[^A-Z\n]"). This is unlike how many other regular expression tools treat negated character classes, but unfortunately the inconsistency is historically entrenched. Matching newlines means that a pattern like [^"]* can match the entire input unless there's another quote in the input.

- A rule can have at most one instance of trailing context (the '/' operator or the '$' operator). The start condition, '^', and "<<EOF>>" patterns can only occur at the beginning of a pattern, and, as well as with '/' and '$', cannot be grouped inside parentheses. A '^' which does not occur at the beginning of a rule or a '$' which does not occur at the end of a rule loses its special properties and is treated as a normal character.

  The following are illegal:

  ```
  foo/bar$
  <sc1>foo<sc2>bar
  ```

  Note that the first of these, can be written "foo/bar\n".

  The following will result in '$' or '^' being treated as a normal character:

  ```
  foo|(bar$)
  foo|^bar
  ```

  If what's wanted is a "foo" or a bar-followed-by-a-newline, the following could be used (the special '|' action is explained below):

```
foo     |
bar$    /* action goes here */
```

A similar trick will work for matching a foo or a bar-at-the-beginning-of-a-line.

HOW THE INPUT IS MATCHED

When the generated scanner is run, it analyzes its input looking for strings which match any of its patterns. If it finds more than one match, it takes the one matching the most text (for trailing context rules, this includes the length of the trailing part, even though it will then be returned to the input). If it finds two or more matches of the same length, the rule listed first in the *flex* input file is chosen.

Once the match is determined, the text corresponding to the match (called the *token)* is made available in the global character pointer **yytext,** and its length in the global integer **yyleng.** The *action* corresponding to the matched pattern is then executed (a more detailed description of actions follows), and then the remaining input is scanned for another match.

If no match is found, then the *default rule* is executed: the next character in the input is considered matched and copied to the standard output. Thus, the simplest legal *flex* input is:

```
%%
```

which generates a scanner that simply copies its input (one character at a time) to its output.

Note that **yytext** can be defined in two different ways: either as a character *pointer* or as a character *array.* You can control which definition *flex* uses by including one of the special directives **%pointer** or **%array** in the first (definitions) section of your flex input. The default is **%pointer,** unless you use the **-l** lex compatibility option, in which case **yytext** will be an array. The advantage of using **%pointer** is substantially faster scanning and no buffer overflow when matching very large tokens (unless you run out of dynamic memory). The disadvantage is that you are restricted in how your actions can modify **yytext** (see the next section), and calls to the **unput()** function destroys the present contents of **yytext,** which can be a considerable porting headache when moving between different *lex* versions.

The advantage of **%array** is that you can then modify **yytext** to your heart's content, and calls to **unput()** do not destroy **yytext** (see below). Furthermore, existing *lex* programs sometimes access **yytext** externally using declarations of the form:

```
extern char yytext[];
```

This definition is erroneous when used with **%pointer,** but correct for **%array.**

%array defines **yytext** to be an array of **YYLMAX** characters, which defaults to a fairly large value. You can change the size by simply #define'ing **YYLMAX** to a different value in the first section of your *flex* input. As mentioned above, with **%pointer** yytext grows dynamically to accommodate large tokens. While this means your **%pointer** scanner can accommodate very large tokens (such as matching entire blocks of comments), bear in mind that each time the scanner must resize **yytext** it also must rescan the entire token from the beginning, so matching such tokens can prove slow. **yytext** presently does *not* dynamically grow if a call to **unput()** results in too much text being pushed back; instead, a run-time error results.

Also note that you cannot use **%array** with C++ scanner classes (the **c++** option; see below).

ACTIONS

Each pattern in a rule has a corresponding action, which can be any arbitrary C statement. The pattern ends at the first non-escaped whitespace character; the remainder of the line is its action. If the action is empty, then when the pattern is matched the input token is simply discarded. For example, here is the specification for a program which deletes all occurrences of "zap me" from its input:

```
%%
"zap me"
```

(It will copy all other characters in the input to the output since they will be matched by the default rule.)

Here is a program which compresses multiple blanks and tabs down to a single blank, and throws away whitespace found at the end of a line:

```
%%
[ \t]+        putchar( ' ' );
[ \t]+$       /* ignore this token */
```

If the action contains a '{', then the action spans till the balancing '}' is found, and the action may cross multiple lines. *flex* knows about C strings and comments and won't be fooled by braces found within them, but also allows actions to begin with **%{** and will consider the action to be all the text up to the next **%}** (regardless of ordinary braces inside the action).

An action consisting solely of a vertical bar ('|') means "same as the action for the next rule." See below for an illustration.

Actions can include arbitrary C code, including **return** statements to return a value to whatever routine called **yylex().** Each time **yylex()** is called it continues processing tokens from where it last left off until it either reaches the end of the file or executes a return.

Actions are free to modify **yytext** except for lengthening it (adding characters to its end--these will overwrite later characters in the input stream). This however does not apply when using **%array** (see above); in that case, **yytext** may be freely modified in any way.

Actions are free to modify **yyleng** except they should not do so if the action also includes use of **yymore()** (see below).

There are a number of special directives which can be included within an action:

- **ECHO** copies yytext to the scanner's output.
- **BEGIN** followed by the name of a start condition places the scanner in the corresponding start condition (see below).
- **REJECT** directs the scanner to proceed on to the "second best" rule which matched the input (or a prefix of the input). The rule is chosen as described above in "How the Input is Matched", and **yytext** and **yyleng** set up appropriately. It may either be one which matched as much text as the originally chosen rule but came later in the *flex* input file, or one which matched less text. For example, the following will both count the words in the input and call the routine special() whenever "frob" is seen:

```
      int word_count = 0;
%%

frob        special(); REJECT;
[^ \t\n]+   ++word_count;
```

Without the **REJECT,** any "frob"'s in the input would not be counted as words, since the scanner normally executes only one action per token. Multiple **REJECT's** are allowed, each one finding the next best choice to the currently active rule. For example, when the following scanner scans the token "abcd", it will write "abcdabcaba" to the output:

```
%%
a        |
ab       |
abc      |
abcd     ECHO; REJECT;
.|\n     /* eat up any unmatched character */
```

(The first three rules share the fourth's action since they use the special '|' action.) **REJECT** is a

particularly expensive feature in terms of scanner performance; if it is used in *any* of the scanner's actions it will slow down *all* of the scanner's matching. Furthermore, **REJECT** cannot be used with the *-Cf* or *-CF* options (see below).

Note also that unlike the other special actions, **REJECT** is a *branch;* code immediately following it in the action will *not* be executed.

- **yymore()** tells the scanner that the next time it matches a rule, the corresponding token should be *appended* onto the current value of **yytext** rather than replacing it. For example, given the input "mega-kludge" the following will write "mega-mega-kludge" to the output:

```
%%
mega-   ECHO; yymore();
kludge  ECHO;
```

First "mega-" is matched and echoed to the output. Then "kludge" is matched, but the previous "mega-" is still hanging around at the beginning of **yytext** so the **ECHO** for the "kludge" rule will actually write "mega-kludge".

Two notes regarding use of **yymore().** First, **yymore()** depends on the value of *yyleng* correctly reflecting the size of the current token, so you must not modify *yyleng* if you are using **yymore().** Second, the presence of **yymore()** in the scanner's action entails a minor performance penalty in the scanner's matching speed.

- **yyless(n)** returns all but the first *n* characters of the current token back to the input stream, where they will be rescanned when the scanner looks for the next match. **yytext** and **yyleng** are adjusted appropriately (e.g., **yyleng** will now be equal to *n*). For example, on the input "foobar" the following will write out "foobarbar":

```
%%
foobar    ECHO; yyless(3);
[a-z]+    ECHO;
```

An argument of 0 to **yyless** will cause the entire current input string to be scanned again. Unless you've changed how the scanner will subsequently process its input (using **BEGIN,** for example), this will result in an endless loop.

Note that **yyless** is a macro and can only be used in the flex input file, not from other source files.

- **unput(c)** puts the character *c* back onto the input stream. It will be the next character scanned. The following action will take the current token and cause it to be rescanned enclosed in parentheses.

```
{
int i;
/* Copy yytext because unput() trashes yytext */
char *yycopy = strdup( yytext );
unput( ')' );
for ( i = yyleng - 1; i  0; --i )
   unput( yycopy[i] );
unput( '(' );
free( yycopy );
}
```

Note that since each **unput()** puts the given character back at the *beginning* of the input stream, pushing back strings must be done back-to-front.

An important potential problem when using **unput()** is that if you are using **%pointer** (the default), a call to **unput()** *destroys* the contents of *yytext,* starting with its rightmost character and devouring one character

to the left with each call. If you need the value of yytext preserved after a call to **unput()** (as in the above example), you must either first copy it elsewhere, or build your scanner using **%array** instead (see How The Input Is Matched).

Finally, note that you cannot put back **EOF** to attempt to mark the input stream with an end-of-file.

- **input()** reads the next character from the input stream. For example, the following is one way to eat up C comments:

```
%%
"/*"      {
          register int c;

          for ( ; ; )
             {
             while ( (c = input()) != '*' &&
                     c != EOF )
                ;    /* eat up text of comment */

             if ( c == '*' )
                {
                while ( (c = input()) == '*' )
                   ;
                if ( c == '/' )
                   break;    /* found the end */
                }

             if ( c == EOF )
                {
                error( "EOF in comment" );
                break;
                }
             }
          }
```

(Note that if the scanner is compiled using **C++,** then **input()** is instead referred to as **yyinput(),** in order to avoid a name clash with the **C++** stream by the name of *input.)*

- **YY_FLUSH_BUFFER** flushes the scanner's internal buffer so that the next time the scanner attempts to match a token, it will first refill the buffer using **YY_INPUT** (see The Generated Scanner, below). This action is a special case of the more general **yy_flush_buffer()** function, described below in the section Multiple Input Buffers.

- **yyterminate()** can be used in lieu of a return statement in an action. It terminates the scanner and returns a 0 to the scanner's caller, indicating "all done". By default, **yyterminate()** is also called when an end-of-file is encountered. It is a macro and may be redefined.

THE GENERATED SCANNER

The output of *flex* is the file **lex.yy.c,** which contains the scanning routine **yylex(),** a number of tables used by it for matching tokens, and a number of auxiliary routines and macros. By default, **yylex()** is declared as follows:

```
int yylex()
   {
   ... various definitions and the actions in here ...
   }
```

(If your environment supports function prototypes, then it will be "int yylex(void)".) This definition may be changed by defining the "YY_DECL" macro. For example, you could use:

```
#define YY_DECL float lexscan( a, b ) float a, b;
```

to give the scanning routine the name *lexscan*, returning a float, and taking two floats as arguments. Note that if you give arguments to the scanning routine using a K&R-style/non-prototyped function declaration, you must terminate the definition with a semi-colon (;).

Whenever **yylex()** is called, it scans tokens from the global input file *yyin* (which defaults to stdin). It continues until it either reaches an end-of-file (at which point it returns the value 0) or one of its actions executes a *return* statement.

If the scanner reaches an end-of-file, subsequent calls are undefined unless either *yyin* is pointed at a new input file (in which case scanning continues from that file), or **yyrestart()** is called. **yyrestart()** takes one argument, a **FILE *** pointer (which can be nil, if you've set up **YY_INPUT** to scan from a source other than *yyin)*, and initializes *yyin* for scanning from that file. Essentially there is no difference between just assigning *yyin* to a new input file or using **yyrestart()** to do so; the latter is available for compatibility with previous versions of *flex,* and because it can be used to switch input files in the middle of scanning. It can also be used to throw away the current input buffer, by calling it with an argument of *yyin;* but better is to use **YY_FLUSH_BUFFER** (see above). Note that **yyrestart()** does *not* reset the start condition to **INITIAL** (see Start Conditions, below).

If **yylex()** stops scanning due to executing a *return* statement in one of the actions, the scanner may then be called again and it will resume scanning where it left off.

By default (and for purposes of efficiency), the scanner uses block-reads rather than simple *getc()* calls to read characters from *yyin.* The nature of how it gets its input can be controlled by defining the **YY_INPUT** macro. YY_INPUT's calling sequence is "YY_INPUT(buf,result,max_size)". Its action is to place up to *max_size* characters in the character array *buf* and return in the integer variable *result* either the number of characters read or the constant YY_NULL (0 on Unix systems) to indicate EOF. The default YY_INPUT reads from the global file-pointer "yyin".

A sample definition of YY_INPUT (in the definitions section of the input file):

```
%{
#define YY_INPUT(buf,result,max_size) \
   { \
   int c = getchar(); \
   result = (c == EOF) ? YY_NULL : (buf[0] = c, 1); \
   }
%}
```

This definition will change the input processing to occur one character at a time.

When the scanner receives an end-of-file indication from YY_INPUT, it then checks the **yywrap()** function. If **yywrap()** returns false (zero), then it is assumed that the function has gone ahead and set up *yyin* to point to another input file, and scanning continues. If it returns true (non-zero), then the scanner terminates, returning 0 to its caller. Note that in either case, the start condition remains unchanged; it does *not* revert to **INITIAL.**

If you do not supply your own version of **yywrap(),** then you must either use **%option noyywrap** (in which case the scanner behaves as though **yywrap()** returned 1), or you must link with **−lfl** to obtain the default version of the routine, which always returns 1.

Three routines are available for scanning from in-memory buffers rather than files: **yy_scan_string(), yy_scan_bytes(),** and **yy_scan_buffer().** See the discussion of them below in the section Multiple Input Buffers.

The scanner writes its **ECHO** output to the *yyout* global (default, stdout), which may be redefined by the

user simply by assigning it to some other **FILE** pointer.

START CONDITIONS

flex provides a mechanism for conditionally activating rules. Any rule whose pattern is prefixed with "<sc>" will only be active when the scanner is in the start condition named "sc". For example,

```
<STRING>[^"]*        { /* eat up the string body ... */
        ...
        }
```

will be active only when the scanner is in the "STRING" start condition, and

```
<INITIAL,STRING,QUOTE>\.        { /* handle an escape ... */
        ...
        }
```

will be active only when the current start condition is either "INITIAL", "STRING", or "QUOTE".

Start conditions are declared in the definitions (first) section of the input using unindented lines beginning with either **%s** or **%x** followed by a list of names. The former declares *inclusive* start conditions, the latter *exclusive* start conditions. A start condition is activated using the **BEGIN** action. Until the next **BEGIN** action is executed, rules with the given start condition will be active and rules with other start conditions will be inactive. If the start condition is *inclusive*, then rules with no start conditions at all will also be active. If it is *exclusive*, then *only* rules qualified with the start condition will be active. A set of rules contingent on the same exclusive start condition describe a scanner which is independent of any of the other rules in the *flex* input. Because of this, exclusive start conditions make it easy to specify "mini-scanners" which scan portions of the input that are syntactically different from the rest (e.g., comments).

If the distinction between inclusive and exclusive start conditions is still a little vague, here's a simple example illustrating the connection between the two. The set of rules:

```
%s example
%%

<example>foo   do_something();

bar            something_else();
```

is equivalent to

```
%x example
%%

<example>foo   do_something();

<INITIAL,example>bar    something_else();
```

Without the **<INITIAL,example>** qualifier, the *bar* pattern in the second example wouldn't be active (i.e., couldn't match) when in start condition **example.** If we just used **<example>** to qualify *bar*, though, then it would only be active in **example** and not in **INITIAL,** while in the first example it's active in both, because in the first example the **example** starting condition is an *inclusive* **(%s)** start condition.

Also note that the special start-condition specifier **<*>** matches every start condition. Thus, the above example could also have been written;

```
%x example
%%
```

```
<example>foo   do_something();

<*>bar    something_else();
```

The default rule (to **ECHO** any unmatched character) remains active in start conditions. It is equivalent to:

```
<*>.|\n     ECHO;
```

BEGIN(0) returns to the original state where only the rules with no start conditions are active. This state can also be referred to as the start-condition "INITIAL", so **BEGIN(INITIAL)** is equivalent to **BEGIN(0).** (The parentheses around the start condition name are not required but are considered good style.)

BEGIN actions can also be given as indented code at the beginning of the rules section. For example, the following will cause the scanner to enter the "SPECIAL" start condition whenever **yylex()** is called and the global variable *enter_special* is true:

```
        int enter_special;

%x SPECIAL
%%
        if ( enter_special )
            BEGIN(SPECIAL);

<SPECIAL>blahblahblah
...more rules follow...
```

To illustrate the uses of start conditions, here is a scanner which provides two different interpretations of a string like "123.456". By default it will treat it as three tokens, the integer "123", a dot (’.’), and the integer "456". But if the string is preceded earlier in the line by the string "expect-floats" it will treat it as a single token, the floating-point number 123.456:

```
%{
#include <math.h>
%}
%s expect

%%
expect-floats        BEGIN(expect);

<expect>[0-9]+"."[0-9]+      {
            printf( "found a float, = %f\n",
                    atof( yytext ) );
            }
<expect>\n           {
            /* that’s the end of the line, so
             * we need another "expect-number"
             * before we’ll recognize any more
             * numbers
             */
            BEGIN(INITIAL);
            }

[0-9]+      {
            printf( "found an integer, = %d\n",
```

```
                atoi( yytext ) );
            }

    "."     printf( "found a dot\n" );
```

Here is a scanner which recognizes (and discards) C comments while maintaining a count of the current input line.

```
    %x comment
    %%
            int line_num = 1;

    "/*"        BEGIN(comment);

    <comment>[^*\n]*        /* eat anything that's not a '*' */
    <comment>"*"+[^*/\n]*   /* eat up '*'s not followed by '/'s */
    <comment>\n             ++line_num;
    <comment>"*"+"/"        BEGIN(INITIAL);
```

This scanner goes to a bit of trouble to match as much text as possible with each rule. In general, when attempting to write a high-speed scanner try to match as much possible in each rule, as it's a big win.

Note that start-conditions names are really integer values and can be stored as such. Thus, the above could be extended in the following fashion:

```
    %x comment foo
    %%
            int line_num = 1;
            int comment_caller;

    "/*"        {
                comment_caller = INITIAL;
                BEGIN(comment);
                }

    ...

    <foo>"/*"   {
                comment_caller = foo;
                BEGIN(comment);
                }

    <comment>[^*\n]*        /* eat anything that's not a '*' */
    <comment>"*"+[^*/\n]*   /* eat up '*'s not followed by '/'s */
    <comment>\n             ++line_num;
    <comment>"*"+"/"        BEGIN(comment_caller);
```

Furthermore, you can access the current start condition using the integer-valued **YY_START** macro. For example, the above assignments to *comment_caller* could instead be written

```
    comment_caller = YY_START;
```

Flex provides **YYSTATE** as an alias for **YY_START** (since that is what's used by AT&T *lex).*

Note that start conditions do not have their own name-space; %s's and %x's declare names in the same fashion as #define's.

Finally, here's an example of how to match C-style quoted strings using exclusive start conditions, including expanded escape sequences (but not including checking for a string that's too long):

```
%x str

%%
        char string_buf[MAX_STR_CONST];
        char *string_buf_ptr;


\"      string_buf_ptr = string_buf; BEGIN(str);

<str>\"        { /* saw closing quote - all done */
        BEGIN(INITIAL);
        *string_buf_ptr = '\0';
        /* return string constant token type and
         * value to parser
         */
        }

<str>\n        {
        /* error - unterminated string constant */
        /* generate error message */
        }

<str>\\[0-7]{1,3} {
        /* octal escape sequence */
        int result;

        (void) sscanf( yytext + 1, "%o", &result );

        if ( result > 0xff )
                /* error, constant is out-of-bounds */

        *string_buf_ptr++ = result;
        }

<str>\\[0-9]+ {
        /* generate error - bad escape sequence; something
         * like '\48' or '\0777777'
         */
        }

<str>\\n  *string_buf_ptr++ = '\n';
<str>\\t  *string_buf_ptr++ = '\t';
<str>\\r  *string_buf_ptr++ = '\r';
<str>\\b  *string_buf_ptr++ = '\b';
<str>\\f  *string_buf_ptr++ = '\f';

<str>\\(.|\n)  *string_buf_ptr++ = yytext[1];

<str>[^\\\n\"]+        {
        char *yptr = yytext;
```

```
while ( *yptr )
      *string_buf_ptr++ = *yptr++;
}
```

Often, such as in some of the examples above, you wind up writing a whole bunch of rules all preceded by the same start condition(s). Flex makes this a little easier and cleaner by introducing a notion of start condition *scope.* A start condition scope is begun with:

```
<SCs>{
```

where *SCs* is a list of one or more start conditions. Inside the start condition scope, every rule automatically has the prefix *<SCs>* applied to it, until a *'}'* which matches the initial *'{'.* So, for example,

```
<ESC>{
   "\\n"   return '\n';
   "\\r"   return '\r';
   "\\f"   return '\f';
   "\\0"   return '\0';
}
```

is equivalent to:

```
<ESC>"\\n"  return '\n';
<ESC>"\\r"  return '\r';
<ESC>"\\f"  return '\f';
<ESC>"\\0"  return '\0';
```

Start condition scopes may be nested.

Three routines are available for manipulating stacks of start conditions:

void yy_push_state(int new_state)
: pushes the current start condition onto the top of the start condition stack and switches to *new_state* as though you had used **BEGIN new_state** (recall that start condition names are also integers).

void yy_pop_state()
: pops the top of the stack and switches to it via **BEGIN.**

int yy_top_state()
: returns the top of the stack without altering the stack's contents.

The start condition stack grows dynamically and so has no built-in size limitation. If memory is exhausted, program execution aborts.

To use start condition stacks, your scanner must include a **%option stack** directive (see Options below).

MULTIPLE INPUT BUFFERS

Some scanners (such as those which support "include" files) require reading from several input streams. As *flex* scanners do a large amount of buffering, one cannot control where the next input will be read from by simply writing a **YY_INPUT** which is sensitive to the scanning context. **YY_INPUT** is only called when the scanner reaches the end of its buffer, which may be a long time after scanning a statement such as an "include" which requires switching the input source.

To negotiate these sorts of problems, *flex* provides a mechanism for creating and switching between multiple input buffers. An input buffer is created by using:

```
YY_BUFFER_STATE yy_create_buffer( FILE *file, int size )
```

which takes a *FILE* pointer and a size and creates a buffer associated with the given file and large enough to hold *size* characters (when in doubt, use **YY_BUF_SIZE** for the size). It returns a **YY_BUFFER_STATE** handle, which may then be passed to other routines (see below). The **YY_BUFFER_STATE** type is a pointer to an opaque **struct yy_buffer_state** structure, so you may safely initialize YY_BUFFER_STATE variables to **((YY_BUFFER_STATE) 0)** if you wish, and also refer to the opaque structure in order to correctly declare input buffers in source files other than that of your scanner. Note that the *FILE* pointer in the call to **yy_create_buffer** is only used as the value of *yyin* seen by **YY_INPUT;** if you redefine **YY_INPUT** so it no longer uses *yyin*, then you can safely pass a nil *FILE* pointer to **yy_create_buffer.** You select a particular buffer to scan from using:

```
void yy_switch_to_buffer( YY_BUFFER_STATE new_buffer )
```

switches the scanner's input buffer so subsequent tokens will come from *new_buffer.* Note that **yy_switch_to_buffer()** may be used by yywrap() to set things up for continued scanning, instead of opening a new file and pointing *yyin* at it. Note also that switching input sources via either **yy_switch_to_buffer()** or **yywrap()** does *not* change the start condition.

```
void yy_delete_buffer( YY_BUFFER_STATE buffer )
```

is used to reclaim the storage associated with a buffer. (**buffer** can be nil, in which case the routine does nothing.) You can also clear the current contents of a buffer using:

```
void yy_flush_buffer( YY_BUFFER_STATE buffer )
```

This function discards the buffer's contents, so the next time the scanner attempts to match a token from the buffer, it will first fill the buffer anew using **YY_INPUT.**

yy_new_buffer() is an alias for **yy_create_buffer(),** provided for compatibility with the C++ use of *new* and *delete* for creating and destroying dynamic objects.

Finally, the **YY_CURRENT_BUFFER** macro returns a **YY_BUFFER_STATE** handle to the current buffer.

Here is an example of using these features for writing a scanner which expands include files (the **<<EOF>>** feature is discussed below):

```
/* the "incl" state is used for picking up the name
 * of an include file
 */
%x incl

%{
#define MAX_INCLUDE_DEPTH 10
YY_BUFFER_STATE include_stack[MAX_INCLUDE_DEPTH];
int include_stack_ptr = 0;
%}

%%
include           BEGIN(incl);

[a-z]+            ECHO;
[^a-z\n]*\n?      ECHO;

<incl>[ \t]*      /* eat the whitespace */
<incl>[^ \t\n]+   { /* got the include file name */
      if ( include_stack_ptr  MAX_INCLUDE_DEPTH )
```

```
        {
        fprintf( stderr, "Includes nested too deeply" );
        exit( 1 );
        }

    include_stack[include_stack_ptr++] =
        YY_CURRENT_BUFFER;

    yyin = fopen( yytext, "r" );

    if ( ! yyin )
        error( ... );

    yy_switch_to_buffer(
        yy_create_buffer( yyin, YY_BUF_SIZE ) );

    BEGIN(INITIAL);
    }

<<EOF>> {
    if ( --include_stack_ptr < 0 )
        {
        yyterminate();
        }

    else
        {
        yy_delete_buffer( YY_CURRENT_BUFFER );
        yy_switch_to_buffer(
            include_stack[include_stack_ptr] );
        }
    }
```

Three routines are available for setting up input buffers for scanning in-memory strings instead of files. All of them create a new input buffer for scanning the string, and return a corresponding **YY_BUFFER_STATE** handle (which you should delete with **yy_delete_buffer()** when done with it). They also switch to the new buffer using **yy_switch_to_buffer(),** so the next call to **yylex()** will start scanning the string.

yy_scan_string(const char *str)
scans a NUL-terminated string.

yy_scan_bytes(const char *bytes, int len)
scans *len* bytes (including possibly NUL's) starting at location *bytes.*

Note that both of these functions create and scan a *copy* of the string or bytes. (This may be desirable, since **yylex()** modifies the contents of the buffer it is scanning.) You can avoid the copy by using:

yy_scan_buffer(char *base, yy_size_t size)
which scans in place the buffer starting at *base*, consisting of *size* bytes, the last two bytes of which *must* be **YY_END_OF_BUFFER_CHAR** (ASCII NUL). These last two bytes are not scanned; thus, scanning consists of **base[0]** through **base[size-2],** inclusive.

If you fail to set up *base* in this manner (i.e., forget the final two **YY_END_OF_BUFFER_CHAR** bytes), then **yy_scan_buffer()** returns a nil pointer instead of creating a new input buffer.

The type **yy_size_t** is an integral type to which you can cast an integer expression reflecting the size of the buffer.

END-OF-FILE RULES

The special rule "<<EOF>>" indicates actions which are to be taken when an end-of-file is encountered and yywrap() returns non-zero (i.e., indicates no further files to process). The action must finish by doing one of four things:

- assigning *yyin* to a new input file (in previous versions of flex, after doing the assignment you had to call the special action **YY_NEW_FILE;** this is no longer necessary);
- executing a *return* statement;
- executing the special **yyterminate()** action;
- or, switching to a new buffer using **yy_switch_to_buffer()** as shown in the example above.

<<EOF>> rules may not be used with other patterns; they may only be qualified with a list of start conditions. If an unqualified <<EOF>> rule is given, it applies to *all* start conditions which do not already have <<EOF>> actions. To specify an <<EOF>> rule for only the initial start condition, use

```
<INITIAL><<EOF>>
```

These rules are useful for catching things like unclosed comments. An example:

```
%x quote
%%

...other rules for dealing with quotes...

<quote><<EOF>>   {
        error( "unterminated quote" );
        yyterminate();
        }
<<EOF>>  {
        if ( *++filelist )
          yyin = fopen( *filelist, "r" );
        else
          yyterminate();
        }
```

MISCELLANEOUS MACROS

The macro **YY_USER_ACTION** can be defined to provide an action which is always executed prior to the matched rule's action. For example, it could be #define'd to call a routine to convert yytext to lower-case. When **YY_USER_ACTION** is invoked, the variable *yy_act* gives the number of the matched rule (rules are numbered starting with 1). Suppose you want to profile how often each of your rules is matched. The following would do the trick:

```
#define YY_USER_ACTION ++ctr[yy_act]
```

where *ctr* is an array to hold the counts for the different rules. Note that the macro **YY_NUM_RULES** gives the total number of rules (including the default rule, even if you use **−s),** so a correct declaration for *ctr* is:

```
int ctr[YY_NUM_RULES];
```

The macro **YY_USER_INIT** may be defined to provide an action which is always executed before the first

scan (and before the scanner's internal initializations are done). For example, it could be used to call a routine to read in a data table or open a logging file.

The macro **yy_set_interactive(is_interactive)** can be used to control whether the current buffer is considered *interactive.* An interactive buffer is processed more slowly, but must be used when the scanner's input source is indeed interactive to avoid problems due to waiting to fill buffers (see the discussion of the **−I** flag below). A non-zero value in the macro invocation marks the buffer as interactive, a zero value as non-interactive. Note that use of this macro overrides **%option always-interactive** or **%option never-interactive** (see Options below). **yy_set_interactive()** must be invoked prior to beginning to scan the buffer that is (or is not) to be considered interactive.

The macro **yy_set_bol(at_bol)** can be used to control whether the current buffer's scanning context for the next token match is done as though at the beginning of a line. A non-zero macro argument makes rules anchored with

The macro **YY_AT_BOL()** returns true if the next token scanned from the current buffer will have '^' rules active, false otherwise.

In the generated scanner, the actions are all gathered in one large switch statement and separated using **YY_BREAK,** which may be redefined. By default, it is simply a "break", to separate each rule's action from the following rule's. Redefining **YY_BREAK** allows, for example, C++ users to #define YY_BREAK to do nothing (while being very careful that every rule ends with a "break" or a "return"!) to avoid suffering from unreachable statement warnings where because a rule's action ends with "return", the **YY_BREAK** is inaccessible.

VALUES AVAILABLE TO THE USER

This section summarizes the various values available to the user in the rule actions.

- **char *yytext** holds the text of the current token. It may be modified but not lengthened (you cannot append characters to the end).

  If the special directive **%array** appears in the first section of the scanner description, then **yytext** is instead declared **char yytext[YYLMAX],** where **YYLMAX** is a macro definition that you can redefine in the first section if you don't like the default value (generally 8KB). Using **%array** results in somewhat slower scanners, but the value of **yytext** becomes immune to calls to *input()* and *unput(),* which potentially destroy its value when **yytext** is a character pointer. The opposite of **%array** is **%pointer,** which is the default.

  You cannot use **%array** when generating C++ scanner classes (the **−+** flag).

- **int yyleng** holds the length of the current token.

- **FILE *yyin** is the file which by default *flex* reads from. It may be redefined but doing so only makes sense before scanning begins or after an EOF has been encountered. Changing it in the midst of scanning will have unexpected results since *flex* buffers its input; use **yyrestart()** instead. Once scanning terminates because an end-of-file has been seen, you can assign *yyin* at the new input file and then call the scanner again to continue scanning.

- **void yyrestart(FILE *new_file)** may be called to point *yyin* at the new input file. The switchover to the new file is immediate (any previously buffered-up input is lost). Note that calling **yyrestart()** with *yyin* as an argument thus throws away the current input buffer and continues scanning the same input file.

- **FILE *yyout** is the file to which **ECHO** actions are done. It can be reassigned by the user.

- **YY_CURRENT_BUFFER** returns a **YY_BUFFER_STATE** handle to the current buffer.

- **YY_START** returns an integer value corresponding to the current start condition. You can subsequently use this value with **BEGIN** to return to that start condition.

INTERFACING WITH YACC

One of the main uses of *flex* is as a companion to the *yacc* parser-generator. *yacc* parsers expect to call a routine named **yylex()** to find the next input token. The routine is supposed to return the type of the next

token as well as putting any associated value in the global **yylval.** To use *flex* with *yacc*, one specifies the **−d** option to *yacc* to instruct it to generate the file **y.tab.h** containing definitions of all the **%tokens** appearing in the *yacc* input. This file is then included in the *flex* scanner. For example, if one of the tokens is "TOK_NUMBER", part of the scanner might look like:

```
%{
#include "y.tab.h"
%}

%%

[0-9]+        yylval = atoi( yytext ); return TOK_NUMBER;
```

OPTIONS

flex has the following options:

−b Generate backing-up information to *lex.backup*. This is a list of scanner states which require backing up and the input characters on which they do so. By adding rules one can remove backing-up states. If *all* backing-up states are eliminated and **−Cf** or **−CF** is used, the generated scanner will run faster (see the **−p** flag). Only users who wish to squeeze every last cycle out of their scanners need worry about this option. (See the section on Performance Considerations below.)

−c is a do-nothing, deprecated option included for POSIX compliance.

−d makes the generated scanner run in *debug* mode. Whenever a pattern is recognized and the global **yy_flex_debug** is non-zero (which is the default), the scanner will write to *stderr* a line of the form:

```
--accepting rule at line 53 ("the matched text")
```

The line number refers to the location of the rule in the file defining the scanner (i.e., the file that was fed to flex). Messages are also generated when the scanner backs up, accepts the default rule, reaches the end of its input buffer (or encounters a NUL; at this point, the two look the same as far as the scanner's concerned), or reaches an end-of-file.

−f specifies *fast scanner*. No table compression is done and stdio is bypassed. The result is large but fast. This option is equivalent to **−Cfr** (see below).

−h generates a "help" summary of *flex's* options to *stdout* and then exits. **−?** and **−−help** are synonyms for **−h.**

−i instructs *flex* to generate a *case-insensitive* scanner. The case of letters given in the *flex* input patterns will be ignored, and tokens in the input will be matched regardless of case. The matched text given in *yytext* will have the preserved case (i.e., it will not be folded).

−l turns on maximum compatibility with the original AT&T *lex* implementation. Note that this does not mean *full* compatibility. Use of this option costs a considerable amount of performance, and it cannot be used with the **−+, -f, -F, -Cf,** or **-CF** options. For details on the compatibilities it provides, see the section "Incompatibilities With Lex And POSIX" below. This option also results in the name **YY_FLEX_LEX_COMPAT** being #define'd in the generated scanner.

−n is another do-nothing, deprecated option included only for POSIX compliance.

−p generates a performance report to stderr. The report consists of comments regarding features of the *flex* input file which will cause a serious loss of performance in the resulting scanner. If you give the flag twice, you will also get comments regarding features that lead to minor performance losses.

Note that the use of **REJECT, %option yylineno,** and variable trailing context (see the Deficiencies / Bugs section below) entails a substantial performance penalty; use of *yymore()*, the ^

operator, and the **−I** flag entail minor performance penalties.

−s causes the *default rule* (that unmatched scanner input is echoed to *stdout)* to be suppressed. If the scanner encounters input that does not match any of its rules, it aborts with an error. This option is useful for finding holes in a scanner's rule set.

−t instructs *flex* to write the scanner it generates to standard output instead of **lex.yy.c.**

−v specifies that *flex* should write to *stderr* a summary of statistics regarding the scanner it generates. Most of the statistics are meaningless to the casual *flex* user, but the first line identifies the version of *flex* (same as reported by **−V),** and the next line the flags used when generating the scanner, including those that are on by default.

−w suppresses warning messages.

−B instructs *flex* to generate a *batch* scanner, the opposite of *interactive* scanners generated by **−I** (see below). In general, you use **−B** when you are *certain* that your scanner will never be used interactively, and you want to squeeze a *little* more performance out of it. If your goal is instead to squeeze out a *lot* more performance, you should be using the **−Cf** or **−CF** options (discussed below), which turn on **−B** automatically anyway.

−F specifies that the *fast* scanner table representation should be used (and stdio bypassed). This representation is about as fast as the full table representation **(-f),** and for some sets of patterns will be considerably smaller (and for others, larger). In general, if the pattern set contains both "keywords" and a catch-all, "identifier" rule, such as in the set:

```
"case"    return TOK_CASE;
"switch"  return TOK_SWITCH;
...
"default" return TOK_DEFAULT;
[a-z]+    return TOK_ID;
```

then you're better off using the full table representation. If only the "identifier" rule is present and you then use a hash table or some such to detect the keywords, you're better off using **-F.**

This option is equivalent to **−CFr** (see below). It cannot be used with **−+.**

−I instructs *flex* to generate an *interactive* scanner. An interactive scanner is one that only looks ahead to decide what token has been matched if it absolutely must. It turns out that always looking one extra character ahead, even if the scanner has already seen enough text to disambiguate the current token, is a bit faster than only looking ahead when necessary. But scanners that always look ahead give dreadful interactive performance; for example, when a user types a newline, it is not recognized as a newline token until they enter *another* token, which often means typing in another whole line.

Flex scanners default to *interactive* unless you use the **−Cf** or **−CF** table-compression options (see below). That's because if you're looking for high-performance you should be using one of these options, so if you didn't, *flex* assumes you'd rather trade off a bit of run-time performance for intuitive interactive behavior. Note also that you *cannot* use **−I** in conjunction with **−Cf** or **−CF.** Thus, this option is not really needed; it is on by default for all those cases in which it is allowed.

You can force a scanner to *not* be interactive by using **−B** (see above).

−L instructs *flex* not to generate **#line** directives. Without this option, *flex* peppers the generated scanner with #line directives so error messages in the actions will be correctly located with respect to either the original *flex* input file (if the errors are due to code in the input file), or **lex.yy.c** (if the errors are *flex's* fault -- you should report these sorts of errors to the email address given below).

−T makes *flex* run in *trace* mode. It will generate a lot of messages to *stderr* concerning the form of the input and the resultant non-deterministic and deterministic finite automata. This option is mostly for use in maintaining *flex.*

−V prints the version number to *stdout* and exits. **−−version** is a synonym for **−V.**

−7 instructs *flex* to generate a 7-bit scanner, i.e., one which can only recognized 7-bit characters in its input. The advantage of using **−7** is that the scanner's tables can be up to half the size of those generated using the **−8** option (see below). The disadvantage is that such scanners often hang or crash if their input contains an 8-bit character.

Note, however, that unless you generate your scanner using the **−Cf** or **−CF** table compression options, use of **−7** will save only a small amount of table space, and make your scanner considerably less portable. *Flex's* default behavior is to generate an 8-bit scanner unless you use the **−Cf** or **−CF,** in which case *flex* defaults to generating 7-bit scanners unless your site was always configured to generate 8-bit scanners (as will often be the case with non-USA sites). You can tell whether flex generated a 7-bit or an 8-bit scanner by inspecting the flag summary in the **−v** output as described above.

Note that if you use **−Cfe** or **−CFe** (those table compression options, but also using equivalence classes as discussed see below), flex still defaults to generating an 8-bit scanner, since usually with these compression options full 8-bit tables are not much more expensive than 7-bit tables.

−8 instructs *flex* to generate an 8-bit scanner, i.e., one which can recognize 8-bit characters. This flag is only needed for scanners generated using **−Cf** or **−CF,** as otherwise flex defaults to generating an 8-bit scanner anyway.

See the discussion of **−7** above for flex's default behavior and the tradeoffs between 7-bit and 8-bit scanners.

−+ specifies that you want flex to generate a C++ scanner class. See the section on Generating C++ Scanners below for details.

−C[aefFmr]

controls the degree of table compression and, more generally, trade-offs between small scanners and fast scanners.

−Ca ("align") instructs flex to trade off larger tables in the generated scanner for faster performance because the elements of the tables are better aligned for memory access and computation. On some RISC architectures, fetching and manipulating longwords is more efficient than with smaller-sized units such as shortwords. This option can double the size of the tables used by your scanner.

−Ce directs *flex* to construct *equivalence classes,* i.e., sets of characters which have identical lexical properties (for example, if the only appearance of digits in the *flex* input is in the character class "[0-9]" then the digits '0', '1', ..., '9' will all be put in the same equivalence class). Equivalence classes usually give dramatic reductions in the final table/object file sizes (typically a factor of 2-5) and are pretty cheap performance-wise (one array look-up per character scanned).

−Cf specifies that the *full* scanner tables should be generated - *flex* should not compress the tables by taking advantages of similar transition functions for different states.

−CF specifies that the alternative fast scanner representation (described above under the **−F** flag) should be used. This option cannot be used with **−+.**

−Cm directs *flex* to construct *meta-equivalence classes,* which are sets of equivalence classes (or characters, if equivalence classes are not being used) that are commonly used together. Meta-equivalence classes are often a big win when using compressed tables, but they have a moderate performance impact (one or two "if" tests and one array look-up per character scanned).

−Cr causes the generated scanner to *bypass* use of the standard I/O library (stdio) for input. Instead of calling **fread()** or **getc(),** the scanner will use the **read()** system call, resulting in a performance gain which varies from system to system, but in general is probably negligible unless you are also using **−Cf** or **−CF.** Using **−Cr** can cause strange behavior if, for example, you read from *yyin* using stdio prior to calling the scanner (because the scanner will miss whatever text your previous reads left in the stdio input buffer).

−Cr has no effect if you define **YY_INPUT** (see The Generated Scanner above).

A lone **−C** specifies that the scanner tables should be compressed but neither equivalence classes nor meta-equivalence classes should be used.

The options **−Cf** or **−CF** and **−Cm** do not make sense together - there is no opportunity for meta-equivalence classes if the table is not being compressed. Otherwise the options may be freely mixed, and are cumulative.

The default setting is **−Cem,** which specifies that *flex* should generate equivalence classes and meta-equivalence classes. This setting provides the highest degree of table compression. You can trade off faster-executing scanners at the cost of larger tables with the following generally being true:

slowest & smallest
-Cem
-Cm
-Ce
-C
-C{f,F}e
-C{f,F}
-C{f,F}a
fastest & largest

Note that scanners with the smallest tables are usually generated and compiled the quickest, so during development you will usually want to use the default, maximal compression.

−Cfe is often a good compromise between speed and size for production scanners.

−ooutput

directs flex to write the scanner to the file **output** instead of **lex.yy.c.** If you combine **−o** with the **−t** option, then the scanner is written to *stdout* but its **#line** directives (see the **−L** option above) refer to the file **output.**

−Pprefix

changes the default *yy* prefix used by *flex* for all globally-visible variable and function names to instead be *prefix.* For example, **−Pfoo** changes the name of **yytext** to **footext.** It also changes the name of the default output file from **lex.yy.c** to **lex.foo.c.** Here are all of the names affected:

yy_create_buffer
yy_delete_buffer
yy_flex_debug
yy_init_buffer
yy_flush_buffer
yy_load_buffer_state
yy_switch_to_buffer
yyin
yyleng
yylex
yylineno
yyout
yyrestart
yytext
yywrap

(If you are using a C++ scanner, then only **yywrap** and **yyFlexLexer** are affected.) Within your scanner itself, you can still refer to the global variables and functions using either version of their name; but externally, they have the modified name.

This option lets you easily link together multiple *flex* programs into the same executable. Note, though, that using this option also renames **yywrap(),** so you now *must* either provide your own (appropriately-named) version of the routine for your scanner, or use **%option noyywrap,** as linking with **−lfl** no longer provides one for you by default.

−Sskeleton_file
overrides the default skeleton file from which *flex* constructs its scanners. You'll never need this option unless you are doing *flex* maintenance or development.

flex also provides a mechanism for controlling options within the scanner specification itself, rather than from the flex command-line. This is done by including **%option** directives in the first section of the scanner specification. You can specify multiple options with a single **%option** directive, and multiple directives in the first section of your flex input file.

Most options are given simply as names, optionally preceded by the word "no" (with no intervening white-space) to negate their meaning. A number are equivalent to flex flags or their negation:

```
7bit          -7 option
8bit          -8 option
align         -Ca option
backup         -b option
batch         -B option
c++           -+ option

caseful or
case-sensitive  opposite of -i (default)

case-insensitive or
caseless       -i option

debug          -d option
default        opposite of -s option
ecs           -Ce option
fast          -F option
full          -f option
interactive    -I option
lex-compat     -l option
meta-ecs       -Cm option
perf-report    -p option
read          -Cr option
stdout         -t option
verbose        -v option
warn           opposite of -w option
              (use "%option nowarn" for -w)

array         equivalent to "%array"
pointer        equivalent to "%pointer" (default)
```

Some **%option's** provide features otherwise not available:

always-interactive
instructs flex to generate a scanner which always considers its input "interactive". Normally, on each new input file the scanner calls **isatty()** in an attempt to determine whether the scanner's input source is interactive and thus should be read a character at a time. When this option is used, however, then no such call is made.

main directs flex to provide a default **main()** program for the scanner, which simply calls **yylex().** This option implies **noyywrap** (see below).

never-interactive
instructs flex to generate a scanner which never considers its input "interactive" (again, no call made to **isatty()).** This is the opposite of **always-interactive.**

stack enables the use of start condition stacks (see Start Conditions above).

stdinit if set (i.e., **%option stdinit)** initializes *yyin* and *yyout* to *stdin* and *stdout,* instead of the default of *nil.* Some existing *lex* programs depend on this behavior, even though it is not compliant with ANSI C, which does not require *stdin* and *stdout* to be compile-time constant.

yylineno
directs *flex* to generate a scanner that maintains the number of the current line read from its input in the global variable **yylineno.** This option is implied by **%option lex-compat.**

yywrap
if unset (i.e., **%option noyywrap),** makes the scanner not call **yywrap()** upon an end-of-file, but simply assume that there are no more files to scan (until the user points *yyin* at a new file and calls **yylex()** again).

flex scans your rule actions to determine whether you use the **REJECT** or **yymore()** features. The **reject** and **yymore** options are available to override its decision as to whether you use the options, either by setting them (e.g., **%option reject)** to indicate the feature is indeed used, or unsetting them to indicate it actually is not used (e.g., **%option noyymore).**

Three options take string-delimited values, offset with '=':

```
%option outfile="ABC"
```

is equivalent to **-oABC,** and

```
%option prefix="XYZ"
```

is equivalent to **-PXYZ.** Finally,

```
%option yyclass="foo"
```

only applies when generating a C++ scanner (**−+** option). It informs *flex* that you have derived **foo** as a subclass of **yyFlexLexer,** so *flex* will place your actions in the member function **foo::yylex()** instead of **yyFlexLexer::yylex().** It also generates a **yyFlexLexer::yylex()** member function that emits a run-time error (by invoking **yyFlexLexer::LexerError())** if called. See Generating C++ Scanners, below, for additional information.

A number of options are available for lint purists who want to suppress the appearance of unneeded routines in the generated scanner. Each of the following, if unset (e.g., **%option nounput**), results in the corresponding routine not appearing in the generated scanner:

```
input, unput
yy_push_state, yy_pop_state, yy_top_state
yy_scan_buffer, yy_scan_bytes, yy_scan_string
```

(though **yy_push_state()** and friends won't appear anyway unless you use **%option stack).**

PERFORMANCE CONSIDERATIONS

The main design goal of *flex* is that it generate high-performance scanners. It has been optimized for dealing well with large sets of rules. Aside from the effects on scanner speed of the table compression **−C** options outlined above, there are a number of options/actions which degrade performance. These are, from most expensive to least:

```
REJECT
%option yylineno
arbitrary trailing context

pattern sets that require backing up
%array
%option interactive
%option always-interactive

'^' beginning-of-line operator
yymore()
```

with the first three all being quite expensive and the last two being quite cheap. Note also that **unput()** is implemented as a routine call that potentially does quite a bit of work, while **yyless()** is a quite-cheap macro; so if just putting back some excess text you scanned, use **yyless().**

REJECT should be avoided at all costs when performance is important. It is a particularly expensive option.

Getting rid of backing up is messy and often may be an enormous amount of work for a complicated scanner. In principal, one begins by using the **−b** flag to generate a *lex.backup* file. For example, on the input

```
%%
foo       return TOK_KEYWORD;
foobar    return TOK_KEYWORD;
```

the file looks like:

```
State #6 is non-accepting -
 associated rule line numbers:
       2       3
 out-transitions: [ o ]
 jam-transitions: EOF [ \001-n  p-\177 ]

State #8 is non-accepting -
 associated rule line numbers:
       3
 out-transitions: [ a ]
 jam-transitions: EOF [ \001-`  b-\177 ]

State #9 is non-accepting -
 associated rule line numbers:
       3
 out-transitions: [ r ]
 jam-transitions: EOF [ \001-q  s-\177 ]

Compressed tables always back up.
```

The first few lines tell us that there's a scanner state in which it can make a transition on an 'o' but not on any other character, and that in that state the currently scanned text does not match any rule. The state occurs when trying to match the rules found at lines 2 and 3 in the input file. If the scanner is in that state and then reads something other than an 'o', it will have to back up to find a rule which is matched. With a bit of headscratching one can see that this must be the state it's in when it has seen "fo". When this has happened, if anything other than another 'o' is seen, the scanner will have to back up to simply match the 'f' (by the default rule).

The comment regarding State #8 indicates there's a problem when "foob" has been scanned. Indeed, on any character other than an 'a', the scanner will have to back up to accept "foo". Similarly, the comment for State #9 concerns when "fooba" has been scanned and an 'r' does not follow.

The final comment reminds us that there's no point going to all the trouble of removing backing up from the rules unless we're using **−Cf** or **−CF,** since there's no performance gain doing so with compressed scanners.

The way to remove the backing up is to add "error" rules:

```
%%
foo         return TOK_KEYWORD;
foobar      return TOK_KEYWORD;

fooba       |
foob        |
fo          {
            /* false alarm, not really a keyword */
            return TOK_ID;
            }
```

Eliminating backing up among a list of keywords can also be done using a "catch-all" rule:

```
%%
foo         return TOK_KEYWORD;
foobar      return TOK_KEYWORD;

[a-z]+      return TOK_ID;
```

This is usually the best solution when appropriate.

Backing up messages tend to cascade. With a complicated set of rules it's not uncommon to get hundreds of messages. If one can decipher them, though, it often only takes a dozen or so rules to eliminate the backing up (though it's easy to make a mistake and have an error rule accidentally match a valid token. A possible future *flex* feature will be to automatically add rules to eliminate backing up).

It's important to keep in mind that you gain the benefits of eliminating backing up only if you eliminate *every* instance of backing up. Leaving just one means you gain nothing.

Variable trailing context (where both the leading and trailing parts do not have a fixed length) entails almost the same performance loss as **REJECT** (i.e., substantial). So when possible a rule like:

```
%%
mouse|rat/(cat|dog)   run();
```

is better written:

```
%%
mouse/cat|dog         run();
rat/cat|dog           run();
```

or as

```
%%
mouse|rat/cat         run();
mouse|rat/dog         run();
```

Note that here the special '|' action does *not* provide any savings, and can even make things worse (see Deficiencies / Bugs below).

Another area where the user can increase a scanner's performance (and one that's easier to implement) arises from the fact that the longer the tokens matched, the faster the scanner will run. This is because with long tokens the processing of most input characters takes place in the (short) inner scanning loop, and does not often have to go through the additional work of setting up the scanning environment (e.g., **yytext)** for the action. Recall the scanner for C comments:

```
%x comment
%%
      int line_num = 1;

"/*"        BEGIN(comment);

<comment>[^*\n]*
<comment>"*"+[^*/\n]*
<comment>\n             ++line_num;
<comment>"*"+"/"        BEGIN(INITIAL);
```

This could be sped up by writing it as:

```
%x comment
%%
      int line_num = 1;

"/*"        BEGIN(comment);

<comment>[^*\n]*
<comment>[^*\n]*\n      ++line_num;
<comment>"*"+[^*/\n]*
<comment>"*"+[^*/\n]*\n ++line_num;
<comment>"*"+"/"        BEGIN(INITIAL);
```

Now instead of each newline requiring the processing of another action, recognizing the newlines is "distributed" over the other rules to keep the matched text as long as possible. Note that *adding* rules does *not* slow down the scanner! The speed of the scanner is independent of the number of rules or (modulo the considerations given at the beginning of this section) how complicated the rules are with regard to operators such as '*' and '|'.

A final example in speeding up a scanner: suppose you want to scan through a file containing identifiers and keywords, one per line and with no other extraneous characters, and recognize all the keywords. A natural first approach is:

```
%%
asm      |
auto     |
break    |
... etc ...
volatile |
while    /* it's a keyword */

.|\n     /* it's not a keyword */
```

To eliminate the back-tracking, introduce a catch-all rule:

```
%%
asm      |
auto     |
break    |
... etc ...
volatile |
while    /* it's a keyword */

[a-z]+   |
.|\n     /* it's not a keyword */
```

Now, if it's guaranteed that there's exactly one word per line, then we can reduce the total number of matches by a half by merging in the recognition of newlines with that of the other tokens:

```
%%
asm\n    |
auto\n   |
break\n  |
... etc ...
volatile\n |
while\n  /* it's a keyword */

[a-z]+\n |
.|\n     /* it's not a keyword */
```

One has to be careful here, as we have now reintroduced backing up into the scanner. In particular, while *we* know that there will never be any characters in the input stream other than letters or newlines, *flex* can't figure this out, and it will plan for possibly needing to back up when it has scanned a token like "auto" and then the next character is something other than a newline or a letter. Previously it would then just match the "auto" rule and be done, but now it has no "auto" rule, only a "auto\n" rule. To eliminate the possibility of backing up, we could either duplicate all rules but without final newlines, or, since we never expect to encounter such an input and therefore don't how it's classified, we can introduce one more catch-all rule, this one which doesn't include a newline:

```
%%
asm\n    |
auto\n   |
break\n  |
... etc ...
volatile\n |
while\n  /* it's a keyword */

[a-z]+\n |
[a-z]+   |
.|\n     /* it's not a keyword */
```

Compiled with **−Cf,** this is about as fast as one can get a *flex* scanner to go for this particular problem.

A final note: *flex* is slow when matching NUL's, particularly when a token contains multiple NUL's. It's best to write rules which match *short* amounts of text if it's anticipated that the text will often include NUL's.

Another final note regarding performance: as mentioned above in the section How the Input is Matched, dynamically resizing **yytext** to accommodate huge tokens is a slow process because it presently requires that the (huge) token be rescanned from the beginning. Thus if performance is vital, you should attempt to match "large" quantities of text but not "huge" quantities, where the cutoff between the two is at about 8K

characters/token.

GENERATING C++ SCANNERS

flex provides two different ways to generate scanners for use with C++. The first way is to simply compile a scanner generated by *flex* using a C++ compiler instead of a C compiler. You should not encounter any compilations errors (please report any you find to the email address given in the Author section below). You can then use C++ code in your rule actions instead of C code. Note that the default input source for your scanner remains *yyin*, and default echoing is still done to *yyout*. Both of these remain *FILE* * variables and not C++ *streams*.

You can also use *flex* to generate a C++ scanner class, using the **−+** option (or, equivalently, **%option c++),** which is automatically specified if the name of the flex executable ends in a '+', such as *flex++*. When using this option, flex defaults to generating the scanner to the file **lex.yy.cc** instead of **lex.yy.c.** The generated scanner includes the header file *FlexLexer.h*, which defines the interface to two C++ classes.

The first class, **FlexLexer,** provides an abstract base class defining the general scanner class interface. It provides the following member functions:

const char* YYText()
returns the text of the most recently matched token, the equivalent of **yytext.**

int YYLeng()
returns the length of the most recently matched token, the equivalent of **yyleng.**

int lineno() const
returns the current input line number (see **%option yylineno),** or **1** if **%option yylineno** was not used.

void set_debug(int flag)
sets the debugging flag for the scanner, equivalent to assigning to **yy_flex_debug** (see the Options section above). Note that you must build the scanner using **%option debug** to include debugging information in it.

int debug() const
returns the current setting of the debugging flag.

Also provided are member functions equivalent to **yy_switch_to_buffer(), yy_create_buffer()** (though the first argument is an **std::istream*** object pointer and not a **FILE*), yy_flush_buffer(), yy_delete_buffer(),** and **yyrestart()** (again, the first argument is a **std::istream*** object pointer).

The second class defined in *FlexLexer.h* is **yyFlexLexer,** which is derived from **FlexLexer.** It defines the following additional member functions:

yyFlexLexer(std::istream* arg_yyin = 0, std::ostream* arg_yyout = 0)
constructs a **yyFlexLexer** object using the given streams for input and output. If not specified, the streams default to **cin** and **cout,** respectively.

virtual int yylex()
performs the same role is **yylex()** does for ordinary flex scanners: it scans the input stream, consuming tokens, until a rule's action returns a value. If you derive a subclass **S** from **yyFlexLexer** and want to access the member functions and variables of **S** inside **yylex(),** then you need to use **%option yyclass="S"** to inform *flex* that you will be using that subclass instead of **yyFlexLexer.** In this case, rather than generating **yyFlexLexer::yylex(),** *flex* generates **S::yylex()** (and also generates a dummy **yyFlexLexer::yylex()** that calls **yyFlexLexer::LexerError()** if called).

virtual void switch_streams(std::istream* new_in = 0,
std::ostream* new_out = 0) reassigns **yyin** to **new_in** (if non-nil) and **yyout** to **new_out** (ditto), deleting the previous input buffer if **yyin** is reassigned.

int yylex(std::istream* new_in, std::ostream* new_out = 0)
first switches the input streams via **switch_streams(new_in, new_out)** and then returns the value of **yylex().**

In addition, **yyFlexLexer** defines the following protected virtual functions which you can redefine in derived classes to tailor the scanner:

virtual int LexerInput(char* buf, int max_size)
reads up to **max_size** characters into **buf** and returns the number of characters read. To indicate end-of-input, return 0 characters. Note that "interactive" scanners (see the **−B** and **−I** flags) define the macro **YY_INTERACTIVE.** If you redefine **LexerInput()** and need to take different actions depending on whether or not the scanner might be scanning an interactive input source, you can test for the presence of this name via **#ifdef.**

virtual void LexerOutput(const char* buf, int size)
writes out **size** characters from the buffer **buf,** which, while NUL-terminated, may also contain "internal" NUL's if the scanner's rules can match text with NUL's in them.

virtual void LexerError(const char* msg)
reports a fatal error message. The default version of this function writes the message to the stream **cerr** and exits.

Note that a **yyFlexLexer** object contains its *entire* scanning state. Thus you can use such objects to create reentrant scanners. You can instantiate multiple instances of the same **yyFlexLexer** class, and you can also combine multiple C++ scanner classes together in the same program using the **−P** option discussed above.

Finally, note that the **%array** feature is not available to C++ scanner classes; you must use **%pointer** (the default).

Here is an example of a simple C++ scanner:

```
    // An example of using the flex C++ scanner class.

%{
int mylineno = 0;
%}

string  \"[^\n"]+\"

ws      [ \t]+

alpha   [A-Za-z]
dig     [0-9]
name    ({alpha}|{dig}|\$)({alpha}|{dig}|[_.\-/$])*
num1    [-+]?{dig}+\.?([eE][-+]?{dig}+)?
num2    [-+]?{dig}*\.{dig}+([eE][-+]?{dig}+)?
number  {num1}|{num2}

%%

{ws}    /* skip blanks and tabs */

"/*"    {
        int c;

        while((c = yyinput()) != 0)
            {
            if(c == '\n')
                ++mylineno;

            else if(c == '*')
                {
```

```
            if((c = yyinput()) == '/')
              break;
            else
              unput(c);
            }
          }
        }

{number}  cout << "number " << YYText() << '\n';

\n        mylineno++;

{name}    cout << "name " << YYText() << '\n';

{string}  cout << "string " << YYText() << '\n';

%%

int main( int /* argc */, char** /* argv */ )
  {
  FlexLexer* lexer = new yyFlexLexer;
  while(lexer->yylex() != 0)
    ;
  return 0;
  }
```

If you want to create multiple (different) lexer classes, you use the **−P** flag (or the **prefix=** option) to rename each **yyFlexLexer** to some other **xxFlexLexer.** You then can include **<FlexLexer.h>** in your other sources once per lexer class, first renaming **yyFlexLexer** as follows:

```
#undef yyFlexLexer
#define yyFlexLexer xxFlexLexer
#include <FlexLexer.h>

#undef yyFlexLexer
#define yyFlexLexer zzFlexLexer
#include <FlexLexer.h>
```

if, for example, you used **%option prefix="xx"** for one of your scanners and **%option prefix="zz"** for the other.

IMPORTANT: the present form of the scanning class is *experimental* and may change considerably between major releases.

INCOMPATIBILITIES WITH LEX AND POSIX

flex is a rewrite of the AT&T Unix *lex* tool (the two implementations do not share any code, though), with some extensions and incompatibilities, both of which are of concern to those who wish to write scanners acceptable to either implementation. Flex is fully compliant with the POSIX *lex* specification, except that when using **%pointer** (the default), a call to **unput()** destroys the contents of **yytext,** which is counter to the POSIX specification.

In this section we discuss all of the known areas of incompatibility between flex, AT&T lex, and the POSIX specification.

flex's **−l** option turns on maximum compatibility with the original AT&T *lex* implementation, at the cost of a major loss in the generated scanner's performance. We note below which incompatibilities can be overcome using the **−l** option.

flex is fully compatible with *lex* with the following exceptions:

- The undocumented *lex* scanner internal variable **yylineno** is not supported unless **−l** or **%option yylineno** is used.

  yylineno should be maintained on a per-buffer basis, rather than a per-scanner (single global variable) basis.

  yylineno is not part of the POSIX specification.

- The **input()** routine is not redefinable, though it may be called to read characters following whatever has been matched by a rule. If **input()** encounters an end-of-file the normal **yywrap()** processing is done. A ''real'' end-of-file is returned by **input()** as *EOF.*

  Input is instead controlled by defining the **YY_INPUT** macro.

  The *flex* restriction that **input()** cannot be redefined is in accordance with the POSIX specification, which simply does not specify any way of controlling the scanner's input other than by making an initial assignment to *yyin.*

- The **unput()** routine is not redefinable. This restriction is in accordance with POSIX.

- *flex* scanners are not as reentrant as *lex* scanners. In particular, if you have an interactive scanner and an interrupt handler which long-jumps out of the scanner, and the scanner is subsequently called again, you may get the following message:

  ```
  fatal flex scanner internal error--end of buffer missed
  ```

  To reenter the scanner, first use

  ```
  yyrestart( yyin );
  ```

  Note that this call will throw away any buffered input; usually this isn't a problem with an interactive scanner.

  Also note that flex C++ scanner classes *are* reentrant, so if using C++ is an option for you, you should use them instead. See "Generating C++ Scanners" above for details.

- **output()** is not supported. Output from the **ECHO** macro is done to the file-pointer *yyout* (default *stdout).*

  output() is not part of the POSIX specification.

- *lex* does not support exclusive start conditions (%x), though they are in the POSIX specification.

- When definitions are expanded, *flex* encloses them in parentheses. With lex, the following:

  ```
  NAME    [A-Z][A-Z0-9]*
  %%
  foo{NAME}?      printf( "Found it\n" );
  %%
  ```

  will not match the string "foo" because when the macro is expanded the rule is equivalent to "foo[A-Z][A-Z0-9]*?" and the precedence is such that the '?' is associated with "[A-Z0-9]*". With *flex*, the rule will be expanded to "foo([A-Z][A-Z0-9]*)?" and so the string "foo" will match.

  Note that if the definition begins with ^ or ends with **$** then it is *not* expanded with parentheses, to allow these operators to appear in definitions without losing their special meanings. But the **<s>, /,** and **<<EOF>>** operators cannot be used in a *flex* definition.

  Using **−l** results in the *lex* behavior of no parentheses around the definition.

  The POSIX specification is that the definition be enclosed in parentheses.

- Some implementations of *lex* allow a rule's action to begin on a separate line, if the rule's pattern has trailing whitespace:

```
%%
foo|bar<space here>
  { foobar_action(); }
```

  flex does not support this feature.

- The *lex* **%r** (generate a Ratfor scanner) option is not supported. It is not part of the POSIX specification.

- After a call to **unput(),** *yytext* is undefined until the next token is matched, unless the scanner was built using **%array.** This is not the case with *lex* or the POSIX specification. The **−l** option does away with this incompatibility.

- The precedence of the **{}** (numeric range) operator is different. *lex* interprets "abc{1,3}" as "match one, two, or three occurrences of 'abc'", whereas *flex* interprets it as "match 'ab' followed by one, two, or three occurrences of 'c'". The latter is in agreement with the POSIX specification.

- The precedence of the ^ operator is different. *lex* interprets "^foo|bar" as "match either 'foo' at the beginning of a line, or 'bar' anywhere", whereas *flex* interprets it as "match either 'foo' or 'bar' if they come at the beginning of a line". The latter is in agreement with the POSIX specification.

- The special table-size declarations such as **%a** supported by *lex* are not required by *flex* scanners; *flex* ignores them.

- The name FLEX_SCANNER is #define'd so scanners may be written for use with either *flex* or *lex*. Scanners also include **YY_FLEX_MAJOR_VERSION** and **YY_FLEX_MINOR_VERSION** indicating which version of *flex* generated the scanner (for example, for the 2.5 release, these defines would be 2 and 5 respectively).

The following *flex* features are not included in *lex* or the POSIX specification:

```
C++ scanners
%option
start condition scopes
start condition stacks
interactive/non-interactive scanners
yy_scan_string() and friends
yyterminate()
yy_set_interactive()
yy_set_bol()
YY_AT_BOL()
<<EOF>>
<*>
YY_DECL
YY_START
YY_USER_ACTION
YY_USER_INIT
#line directives
%{}'s around actions
multiple actions on a line
```

plus almost all of the flex flags. The last feature in the list refers to the fact that with *flex* you can put multiple actions on the same line, separated with semi-colons, while with *lex*, the following

```
foo    handle_foo(); ++num_foos_seen;
```

is (rather surprisingly) truncated to

```
foo    handle_foo();
```

flex does not truncate the action. Actions that are not enclosed in braces are simply terminated at the end of the line.

DIAGNOSTICS

warning, rule cannot be matched indicates that the given rule cannot be matched because it follows other rules that will always match the same text as it. For example, in the following "foo" cannot be matched because it comes after an identifier "catch-all" rule:

```
[a-z]+    got_identifier();
foo       got_foo();
```

Using **REJECT** in a scanner suppresses this warning.

warning, **−s** *option given but default rule can be matched* means that it is possible (perhaps only in a particular start condition) that the default rule (match any single character) is the only one that will match a particular input. Since **−s** was given, presumably this is not intended.

reject_used_but_not_detected undefined or *yymore_used_but_not_detected undefined* - These errors can occur at compile time. They indicate that the scanner uses **REJECT** or **yymore()** but that *flex* failed to notice the fact, meaning that *flex* scanned the first two sections looking for occurrences of these actions and failed to find any, but somehow you snuck some in (via a #include file, for example). Use **%option reject** or **%option yymore** to indicate to flex that you really do use these features.

flex scanner jammed - a scanner compiled with **−s** has encountered an input string which wasn't matched by any of its rules. This error can also occur due to internal problems.

token too large, exceeds YYLMAX - your scanner uses **%array** and one of its rules matched a string longer than the **YYLMAX** constant (8K bytes by default). You can increase the value by #define'ing **YYLMAX** in the definitions section of your *flex* input.

scanner requires −8 flag to use the character 'x' - Your scanner specification includes recognizing the 8-bit character *'x'* and you did not specify the −8 flag, and your scanner defaulted to 7-bit because you used the **−Cf** or **−CF** table compression options. See the discussion of the **−7** flag for details.

flex scanner push-back overflow - you used **unput()** to push back so much text that the scanner's buffer could not hold both the pushed-back text and the current token in **yytext.** Ideally the scanner should dynamically resize the buffer in this case, but at present it does not.

input buffer overflow, can't enlarge buffer because scanner uses REJECT - the scanner was working on matching an extremely large token and needed to expand the input buffer. This doesn't work with scanners that use **REJECT.**

fatal flex scanner internal error--end of buffer missed - This can occur in an scanner which is reentered after a long-jump has jumped out (or over) the scanner's activation frame. Before reentering the scanner, use:

```
yyrestart( yyin );
```

or, as noted above, switch to using the C++ scanner class.

too many start conditions in <> construct! - you listed more start conditions in a <> construct than exist (so you must have listed at least one of them twice).

FILES

−lfl library with which scanners must be linked.

lex.yy.c generated scanner (called *lexyy.c* on some systems).

lex.yy.cc
generated C++ scanner class, when using **-+.**

<FlexLexer.h>
header file defining the C++ scanner base class, **FlexLexer,** and its derived class, **yyFlexLexer.**

flex.skl skeleton scanner. This file is only used when building flex, not when flex executes.

lex.backup
backing-up information for **−b** flag (called *lex.bck* on some systems).

DEFICIENCIES / BUGS

Some trailing context patterns cannot be properly matched and generate warning messages ("dangerous trailing context"). These are patterns where the ending of the first part of the rule matches the beginning of the second part, such as "zx*/xy*", where the 'x*' matches the 'x' at the beginning of the trailing context. (Note that the POSIX draft states that the text matched by such patterns is undefined.)

For some trailing context rules, parts which are actually fixed-length are not recognized as such, leading to the above mentioned performance loss. In particular, parts using '|' or {n} (such as "foo{3}") are always considered variable-length.

Combining trailing context with the special '|' action can result in *fixed* trailing context being turned into the more expensive *variable* trailing context. For example, in the following:

```
%%
abc     |
xyz/def
```

Use of **unput()** invalidates yytext and yyleng, unless the **%array** directive or the **−l** option has been used.

Pattern-matching of NUL's is substantially slower than matching other characters.

Dynamic resizing of the input buffer is slow, as it entails rescanning all the text matched so far by the current (generally huge) token.

Due to both buffering of input and read-ahead, you cannot intermix calls to <stdio.h> routines, such as, for example, **getchar(),** with *flex* rules and expect it to work. Call **input()** instead.

The total table entries listed by the **−v** flag excludes the number of table entries needed to determine what rule has been matched. The number of entries is equal to the number of DFA states if the scanner does not use **REJECT,** and somewhat greater than the number of states if it does.

REJECT cannot be used with the **−f** or **−F** options.

The *flex* internal algorithms need documentation.

SEE ALSO

lex(1), yacc(1), sed(1), awk(1).

John Levine, Tony Mason, and Doug Brown, *Lex & Yacc*, O'Reilly and Associates. Be sure to get the 2nd edition.

M. E. Lesk and E. Schmidt, *LEX − Lexical Analyzer Generator*

Alfred Aho, Ravi Sethi and Jeffrey Ullman, *Compilers: Principles, Techniques and Tools*, Addison-Wesley (1986). Describes the pattern-matching techniques used by *flex* (deterministic finite automata).

AUTHOR

Vern Paxson, with the help of many ideas and much inspiration from Van Jacobson. Original version by Jef Poskanzer. The fast table representation is a partial implementation of a design done by Van Jacobson. The implementation was done by Kevin Gong and Vern Paxson.

Thanks to the many *flex* beta-testers, feedbackers, and contributors, especially Francois Pinard, Casey Leedom, Robert Abramovitz, Stan Adermann, Terry Allen, David Barker-Plummer, John Basrai, Neal Becker, Nelson H.F. Beebe, benson@odi.com, Karl Berry, Peter A. Bigot, Simon Blanchard, Keith Bostic, Frederic

Brehm, Ian Brockbank, Kin Cho, Nick Christopher, Brian Clapper, J.T. Conklin, Jason Coughlin, Bill Cox, Nick Cropper, Dave Curtis, Scott David Daniels, Chris G. Demetriou, Theo de Raadt, Mike Donahue, Chuck Doucette, Tom Epperly, Leo Eskin, Chris Faylor, Chris Flatters, Jon Forrest, Jeffrey Friedl, Joe Gayda, Kaveh R. Ghazi, Wolfgang Glunz, Eric Goldman, Christopher M. Gould, Ulrich Grepel, Peer Griebel, Jan Hajic, Charles Hemphill, NORO Hideo, Jarkko Hietaniemi, Scott Hofmann, Jeff Honig, Dana Hudes, Eric Hughes, John Interrante, Ceriel Jacobs, Michal Jaegermann, Sakari Jalovaara, Jeffrey R. Jones, Henry Juengst, Klaus Kaempf, Jonathan I. Kamens, Terrence O Kane, Amir Katz, ken@ken.hilco.com, Kevin B. Kenny, Steve Kirsch, Winfried Koenig, Marq Kole, Ronald Lamprecht, Greg Lee, Rohan Lenard, Craig Leres, John Levine, Steve Liddle, David Loffredo, Mike Long, Mohamed el Lozy, Brian Madsen, Malte, Joe Marshall, Bengt Martensson, Chris Metcalf, Luke Mewburn, Jim Meyering, R. Alexander Milowski, Erik Naggum, G.T. Nicol, Landon Noll, James Nordby, Marc Nozell, Richard Ohnemus, Karsten Pahnke, Sven Panne, Roland Pesch, Walter Pelissero, Gaumond Pierre, Esmond Pitt, Jef Poskanzer, Joe Rahmeh, Jarmo Raiha, Frederic Raimbault, Pat Rankin, Rick Richardson, Kevin Rodgers, Kai Uwe Rommel, Jim Roskind, Alberto Santini, Andreas Scherer, Darrell Schiebel, Raf Schietekat, Doug Schmidt, Philippe Schnoebelen, Andreas Schwab, Larry Schwimmer, Alex Siegel, Eckehard Stolz, Jan-Erik Strvmquist, Mike Stump, Paul Stuart, Dave Tallman, Ian Lance Taylor, Chris Thewalt, Richard M. Timoney, Jodi Tsai, Paul Tuinenga, Gary Weik, Frank Whaley, Gerhard Wilhelms, Kent Williams, Ken Yap, Ron Zellar, Nathan Zelle, David Zuhn, and those whose names have slipped my marginal mail-archiving skills but whose contributions are appreciated all the same.

Thanks to Keith Bostic, Jon Forrest, Noah Friedman, John Gilmore, Craig Leres, John Levine, Bob Mulcahy, G.T. Nicol, Francois Pinard, Rich Salz, and Richard Stallman for help with various distribution headaches.

Thanks to Esmond Pitt and Earle Horton for 8-bit character support; to Benson Margulies and Fred Burke for C++ support; to Kent Williams and Tom Epperly for C++ class support; to Ove Ewerlid for support of NUL's; and to Eric Hughes for support of multiple buffers.

This work was primarily done when I was with the Real Time Systems Group at the Lawrence Berkeley Laboratory in Berkeley, CA. Many thanks to all there for the support I received.

Send comments to vern@ee.lbl.gov.

NAME

fmt — simple text formatter

SYNOPSIS

fmt [**-Cr**] [*goal* [*maximum*]] [name ...]
fmt [**-Cr**] [**-g** *goal*] [**-m** *maximum*] [name ...]

DESCRIPTION

fmt is a simple text formatter which reads the concatenation of input files (or standard input if none are given) and produces on standard output a version of its input with lines as close to the *goal* length as possible without exceeding the *maximum*. The *goal* length defaults to 65 and the *maximum* to 75. The spacing at the beginning of the input lines is preserved in the output, as are blank lines and interword spacing. In non raw mode, lines that look like mail headers or begin with a period are not formatted.

-C instructs **fmt** to center the text.

-g *goal* New way to set the goal length.

-m *maximum*
New way to set the maximum length.

-r Raw mode; formats all lines and does not make exceptions for lines that start with a period or look like mail headers.

fmt is meant to format mail messages prior to sending, but may also be useful for other simple tasks. For instance, within visual mode of the ex(1) editor (e.g., vi(1)) the command

```
!}fmt
```

will reformat a paragraph, evening the lines.

SEE ALSO

mail(1), nroff(1)

HISTORY

The **fmt** command appeared in 3BSD.

BUGS

The program was designed to be simple and fast − for more complex operations, the standard text processors are likely to be more appropriate.

NAME

fold — fold long lines for finite width output device

SYNOPSIS

fold [**-bs**] [**-w** *width*] *file . . .*

DESCRIPTION

fold is a filter which folds the contents of the specified files, or the standard input if no files are specified, breaking the lines to have maximum of 80 characters.

The options are as follows:

-b Count *width* in bytes rather than column positions.

-s Fold line after the last blank character within the first *width* column positions (or bytes).

-w Specifies a line width to use instead of the default 80 characters.

The **fold** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

expand(1)

STANDARDS

The **fold** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

BUGS

If underlining is present it may be messed up by folding.

Width should be a multiple of 8 if tabs are present, or the tabs should be expanded using expand(1) before using **fold**.

NAME

fpr — print Fortran file

SYNOPSIS

fpr

DESCRIPTION

fpr is a filter that transforms files formatted according to Fortran's carriage control conventions into files formatted according to UNIX line printer conventions.

fpr copies its input onto its output, replacing the carriage control characters with characters that will produce the intended effects when printed using `lpr`(1). The first character of each line determines the vertical spacing as follows:

Blank	One line
0	Two lines
1	To first line of next page
+	No advance

A blank line is treated as if its first character is a blank. A blank that appears as a carriage control character is deleted. A zero is changed to a newline. A one is changed to a form feed. The effects of a '+' are simulated using backspaces.

EXAMPLES

```
a.out | fpr | lpr
```

```
fpr < f77.output | lpr
```

HISTORY

The **fpr** command appeared in 4.2BSD.

BUGS

Results are undefined for input lines longer than 170 characters.

NAME

from — print names of those who have sent mail

SYNOPSIS

from [**-s** *sender*] [**-f** *file*] [*user*]

DESCRIPTION

from prints out the mail header lines from the invoker's mailbox.

Options:

-f *file* The supplied file is examined instead of the invoker's mailbox. If the **-f** option is used, the *user* argument should not be used.

-s *sender*
Only mail from addresses containing the supplied string are printed.

If *user* is given, the *user*'s mailbox, is examined instead of the invoker's own mailbox. (Privileges are required.)

FILES

/var/mail/*

SEE ALSO

biff(1), mail(1)

HISTORY

The **from** command appeared in 3.0 BSD.

NAME

fsplit — split a multi-routine Fortran file into individual files

SYNOPSIS

fsplit [**-e** *efile*] ... [*file*]

DESCRIPTION

fsplit takes as input either a file or standard input containing Fortran source code. It attempts to split the input into separate routine files of the form *name.f*, where *name* is the name of the program unit (e.g. function, subroutine, block data or program). The name for unnamed block data subprograms has the form *blkdtaNNN.f* where NNN is three digits and a file of this name does not already exist. For unnamed main programs the name has the form *mainNNN.f*. If there is an error in classifying a program unit, or if *name.f* already exists, the program unit will be put in a file of the form *zzzNNN.f* where *zzzNNN.f* does not already exist.

-e *efile* Normally each subprogram unit is split into a separate file. When the **-e** option is used, only the specified subprogram units are split into separate files. E.g.:

```
fsplit -e readit -e doit prog.f
```

will split readit and doit into separate files.

DIAGNOSTICS

If names specified via the **-e** option are not found, a diagnostic is written to standard error.

HISTORY

The **fsplit** command appeared in 4.2BSD.

AUTHORS

Asa Romberger and Jerry Berkman

BUGS

fsplit assumes the subprogram name is on the first noncomment line of the subprogram unit. Nonstandard source formats may confuse **fsplit**.

It is hard to use **-e** for unnamed main programs and block data subprograms since you must predict the created file name.

NAME

fstat — display status of open files

SYNOPSIS

fstat [**-fnv**] [**-M** *core*] [**-N** *system*] [**-p** *pid*] [**-u** *user*] [*file ...*]

DESCRIPTION

fstat identifies open files. A file is considered open by a process if it was explicitly opened, is the working directory, root directory, active pure text, or kernel trace file for that process. If no options are specified, **fstat** reports on all open files in the system.

Options:

-f Restrict examination to files open in the same filesystems as the named file arguments, or to the filesystem containing the current directory if there are no additional filename arguments. For example, to find all files open in the filesystem where the directory `/usr/src` resides, type "`fstat -f /usr/src`".

-M Extract values associated with the name list from the specified core instead of the default `/dev/kmem`.

-N Extract the name list from the specified system instead of the default `/netbsd`.

-n Numerical format. Print the device number (maj,min) of the filesystem the file resides in rather than the mount point name; for special files, print the device number that the special device refers to rather than the filename in `/dev`; and print the mode of the file in octal instead of symbolic form.

-p Report all files open by the specified process.

-u Report all files open by the specified user.

-v Verbose mode. Print error messages upon failures to locate particular system data structures rather than silently ignoring them. Most of these data structures are dynamically created or deleted and it is possible for them to disappear while **fstat** is running. This is normal and unavoidable since the rest of the system is running while **fstat** itself is running.

file ...
Restrict reports to the specified files.

The following fields are printed:

`USER` The username of the owner of the process (effective UID).

`CMD` The command name of the process.

`PID` The process ID.

`FD` The file number in the per-process open file table or one of the following special names:

`text`	pure text inode
`wd`	current working directory
`root`	root inode
`tr`	kernel trace file

If the file number is followed by an asterisk ("*"), the file is not an inode, but rather a socket, FIFO, or there is an error. In this case the remainder of the line doesn't correspond to the remaining headers -- the format of the line is described later under **SOCKETS**.

MOUNT If the **-n** flag wasn't specified, this header is present and is the pathname that the filesystem the file resides in is mounted on.

DEV If the **-n** flag is specified, this header is present and is the major/minor number of the device that this file resides in.

INUM The inode number of the file.

MODE The mode of the file. If the **-n** flag isn't specified, the mode is printed using a symbolic format (see `strmode`(3)); otherwise, the mode is printed as an octal number.

SZ|DV If the file is not a character or block special file, prints the size of the file in bytes. Otherwise, if the **-n** flag is not specified, prints the name of the special file as located in `/dev`. If that cannot be located, or the **-n** flag is specified, prints the major/minor device number that the special device refers to.

R/W This column describes the access mode that the file allows. The letter "r" indicates open for reading; the letter "w" indicates open for writing. This field is useful when trying to find the processes that are preventing a filesystem from being downgraded to read-only.

NAME If filename arguments are specified and the **-f** flag is not, then this field is present and is the name associated with the given file. Normally the name cannot be determined since there is no mapping from an open file back to the directory entry that was used to open that file. Also, since different directory entries may reference the same file (via `ln`(1)), the name printed may not be the actual name that the process originally used to open that file.

SOCKETS

The formatting of open sockets depends on the protocol domain. In all cases the first field is the domain name, the second field is the socket type (stream, dgram, etc), and the third is the socket flags field (in hex). The remaining fields are protocol dependent. For TCP, it is the address of the tcpcb, and for UDP, the inpcb (socket pcb). For UNIX domain sockets, its the address of the socket pcb and the address of the connected pcb (if connected). Otherwise the protocol number and address of the socket itself are printed. The attempt is to make enough information available to permit further analysis without duplicating `netstat`(1).

For example, the addresses mentioned above are the addresses which the "`netstat -A`" command would print for TCP, UDP, and UNIX domain. Note that since pipes are implemented using sockets, a pipe appears as a connected UNIX domain stream socket. A unidirectional UNIX domain socket indicates the direction of flow with an arrow ("<-" or "->"), and a full duplex socket shows a double arrow ("<->").

For internet sockets **fstat** also attempts to print the internet address and port for the local end of a connection. If the socket is connected, it also prints the remote internet address and port. An asterisk ("*") is used to indicate an INADDR_ANY binding.

SEE ALSO

`netstat`(1), `nfsstat`(1), `ps`(1), `sockstat`(1), `systat`(1), `vmstat`(1), `iostat`(8), `pstat`(8)

HISTORY

The **fstat** command appeared in 4.3BSD−Tahoe.

BUGS

Since **fstat** takes a snapshot of the system, it is only correct for a very short period of time.

Moreover, because DNS resolution and YP lookups cause many file descriptor changes, **fstat** does not attempt to translate the internet address and port numbers into symbolic names.

NAME

ftp — ARPANET file transfer program

SYNOPSIS

ftp [**-K**] [**-d**] [**-g**] [**-i**] [**-l**] [**-n**] [**-p**] [**-t**] [**-v**] [**-x**] [**--no-gss-bindings**] [**--no-gss-delegate**] [*host*]

DESCRIPTION

ftp is the user interface to the ARPANET standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

Modifications have been made so that it almost follows the FTP Security Extensions, RFC 2228.

Options may be specified at the command line, or to the command interpreter.

-K Disable Kerberos authentication.

-t Enables packet tracing.

-v Verbose option forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics.

-n Restrains **ftp** from attempting "auto-login" upon initial connection. If auto-login is enabled, **ftp** will check the `.netrc` (see below) file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.

-i Turns off interactive prompting during multiple file transfers.

-p Turn on passive mode.

-d Enables debugging.

-g Disables file name globbing.
.It Fl -no-gss-bindings Don't use GSS-API bindings when talking to peer. IP addresses will not be checked to ensure they match.

--no-gss-delegate
Disable delegation of GSSAPI credentials.

-l Disables command line editing.

-x Encrypt command and data channel.

The client host with which **ftp** is to communicate may be specified on the command line. If this is done, **ftp** will immediately attempt to establish a connection to an FTP server on that host; otherwise, **ftp** will enter its command interpreter and await instructions from the user. When **ftp** is awaiting commands from the user the prompt `ftp>` is provided to the user. The following commands are recognized by **ftp**:

! [*command* [*args*]]
Invoke an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any **ntrans** or **nmap** setting. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

ascii Set the file transfer **type** to network ASCII. This is the default type.

bell Arrange that a bell be sounded after each file transfer command is completed.

binary Set the file transfer **type** to support binary image transfer.

bye Terminate the FTP session with the remote server and exit **ftp**. An end of file will also terminate the session and exit.

case Toggle remote computer file name case mapping during **mget** commands. When **case** is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.

cd *remote-directory*
Change the working directory on the remote machine to *remote-directory*.

cdup Change the remote machine working directory to the parent of the current remote machine working directory.

chmod *mode file-name*
Change the permission modes of the file *file-name* on the remote sytem to *mode*.

close Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

cr Toggle carriage return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non−UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete the file *remote-file* on the remote machine.

debug [*debug-value*]
Toggle debugging mode. If an optional *debug-value* is specified it is used to set the debugging level. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string `−−>`

dir [*remote-directory*] [*local-file*]
Print a listing of the directory contents in the directory, *remote-directory*, and, optionally, placing the output in *local-file*. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **dir** output. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, or *local-file* is **−**, output comes to the terminal.

disconnect
A synonym for *close*.

form *format*
Set the file transfer **form** to *format*. The default format is "file".

get *remote-file* [*local-file*]
Retrieve the *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. The current settings for **type**, **form**, **mode**, and **structure** are used while transferring the file.

glob Toggle filename expansion for **mdelete**, **mget** and **mput**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in csh(1). For **mdelete** and **mget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file: the exact result depends on the foreign operating system and ftp server, and can be previewed by doing `mls remote-files -`. As a security measure, remotely globbed files that starts with '/' or contains '../', will not be automatically received. If you have interactive prompting turned off, these filenames will be ignored. Note: **mget** and **mput** are not meant to transfer entire directory subtrees of files. That can be done by transferring a tar(1) archive of the subtree (in binary mode).

hash Toggle hash-sign ("#") printing for each data block transferred. The size of a data block is 1024 bytes.

help [*command*]
Print an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of the known commands.

idle [*seconds*]
Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, the current inactivity timer is printed.

lcd [*directory*]
Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

ls [*remote-directory*] [*local-file*]
Print a listing of the contents of a directory on the remote machine. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems will produce output from the command `ls -l`. (See also **nlist**.) If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **ls** output. If no local file is specified, or if *local-file* is '**-**', the output is sent to the terminal.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets '$' and '\' as special characters. A '$' followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A '$' followed by an 'i' signals that macro processor that the executing macro is to be looped. On the first pass '$i' is replaced by the first argument on the macro invocation command line, on the second pass it is replaced

by the second argument, and so on. A '\' followed by any character is replaced by that character. Use the '\' to prevent special treatment of the '$'.

mdelete [*remote-files*]
Delete the *remote-files* on the remote machine.

mdir *remote-files local-file*
Like **dir**, except multiple remote files may be specified. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget *remote-files*
Expand the *remote-files* on the remote machine and do a **get** for each file name thus produced. See **glob** for details on the filename expansion. Resulting file names will then be processed according to **case**, **ntrans**, and **nmap** settings. Files are transferred into the local working directory, which can be changed with `lcd directory`; new local directories can be created with `! mkdir directory`.

mkdir *directory-name*
Make a directory on the remote machine.

mls *remote-files local-file*
Like **nlist**, except multiple remote files may be specified, and the *local-file* must be specified. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mls** output.

mode [*mode-name*]
Set the file transfer **mode** to *mode-name*. The default mode is "stream" mode.

modtime *file-name*
Show the last modification time of the file on the remote machine.

mput *local-files*
Expand wild cards in the list of local files given as arguments and do a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting file names will then be processed according to **ntrans** and **nmap** settings.

newer *file-name*
Get the file only if the modification time of the remote file is more recent that the file on the current system. If the file does not exist on the current system, the remote file is considered **newer**. Otherwise, this command is identical to *get*.

nlist [*remote-directory*] [*local-file*]
Print a list of the files in a directory on the remote machine. If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **nlist** output. If no local file is specified, or if *local-file* is **-**, the output is sent to the terminal.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non−UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. [*Inpattern*] is a tem-

plate for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences '$1', '$2', ..., '$9' in *inpattern*. Use '\' to prevent this special treatment of the '$' character. All other characters are treated literally, and are used to determine the **nmap** [*inpattern*] variable values. For example, given *inpattern* $1.$2 and the remote file name "mydata.data", $1 would have the value "mydata", and $2 would have the value "data". The *outpattern* determines the resulting mapped filename. The sequences '$1', '$2', ..., '$9' are replaced by any value resulting from the *inpattern* template. The sequence '$0' is replace by the original filename. Additionally, the sequence '[*seq1*, *seq2*]' is replaced by [*seq1*] if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command

```
nmap $1.$2.$3 [$1,$2].[$2,file]
```

would yield the output filename "myfile.data" for input filenames "myfile.data" and "myfile.data.old", "myfile.file" for the input filename "myfile", and "myfile.myfile" for the input filename ".myfile". Spaces may be included in *outpattern*, as in the example: 'nmap $1 sed "s/ *$//" > $1' . Use the '\' character to prevent special treatment of the '$','[','[', and ',' characters.

ntrans [*inchars* [*outchars*]]

Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non−UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

open *host* [*port*]

Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, **ftp** will attempt to contact an FTP server at that port. If the **auto-login** option is on (default), **ftp** will also attempt to automatically log the user in to the FTP server (see below).

passive Toggle passive mode. If passive mode is turned on (default is off), the ftp client will send a PASV command for all data connections instead of the usual PORT command. The PASV command requests that the remote server open a port for the data connection and return the address of that port. The remote server listens on that port and the client connects to it. When using the more traditional PORT command, the client listens on a port and sends that address to the remote server, who connects back to it. Passive mode is useful when using **ftp** through a gateway router or host that controls the directionality of traffic. (Note that though ftp servers are required to support the PASV command by RFC 1123, some do not.)

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** will transfer all files, and any **mdelete** will delete all files.

proxy *ftp-command*

Execute an ftp command on a secondary control connection. This command allows simultaneous connection to two remote ftp servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection.

Enter the command "proxy ?" to see other ftp commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** will not define new macros during the auto-login process, **close** will not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the ftp protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any **ntrans** or **nmap** settings in naming the remote file. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

pwd Print the name of the current working directory on the remote machine.

quit A synonym for **bye**.

quote *arg1 arg2 ...*
The arguments specified are sent, verbatim, to the remote FTP server.

recv *remote-file* [*local-file*]
A synonym for get.

reget *remote-file* [*local-file*]
Reget acts like get, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that are prone to dropping connections.

remotehelp [*command-name*]
Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

remotestatus [*file-name*]
With no arguments, show status of remote machine. If *file-name* is specified, show status of *file-name* on remote machine.

rename [*from*] [*to*]
Rename the file *from* on the remote machine, to the file *to*.

reset Clear reply queue. This command re-synchronizes command/reply sequencing with the remote ftp server. Resynchronization may be necessary following a violation of the ftp protocol by the remote server.

restart *marker*
Restart the immediately following **get** or **put** at the indicated *marker*. On UNIX systems, marker is usually a byte offset into the file.

rmdir *directory-name*
Delete a directory on the remote machine.

runique Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a ".1" is appended to the name. If the resulting name matches another existing file, a ".2" is appended to the original name. If this process continues up to ".99", an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that **runique** will not

affect local files generated from a shell command (see below). The default value is off.

send *local-file* [*remote-file*]
A synonym for put.

sendport Toggle the use of PORT commands. By default, **ftp** will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, **ftp** will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer. This is useful for certain FTP implementations which do ignore PORT commands but, incorrectly, indicate they've been accepted.

site *arg1 arg2 ...*
The arguments specified are sent, verbatim, to the remote FTP server as a SITE command.

size *file-name*
Return size of *file-name* on remote machine.

status Show the current status of **ftp**.

struct [*struct-name*]
Set the file transfer *structure* to *struct-name*. By default "stream" structure is used.

sunique Toggle storing of files on remote machine under unique file names. Remote ftp server must support ftp protocol STOU command for successful completion. The remote server will report unique name. Default value is off.

system Show the type of operating system running on the remote machine.

tenex Set the file transfer type to that needed to talk to TENEX machines.

trace Toggle packet tracing.

type [*type-name*]
Set the file transfer **type** to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

umask [*newmask*]
Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

user *user-name* [*password*] [*account*]
Identify yourself to the remote FTP server. If the *password* is not specified and the server requires it, **ftp** will prompt the user for it (after disabling local echo). If an *account* field is not specified, and the FTP server requires it, the user will be prompted for it. If an *account* field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless **ftp** is invoked with "auto-login" disabled, this process is done automatically on initial connection to the FTP server.

verbose Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

? [*command*]
A synonym for help.

The following command can be used with ftpsec-aware servers.

prot *clear* | *safe* | *confidential* | *private*
Set the data protection level to the requested level.

The following command can be used with ftp servers that has implemented the KAUTH site command.

kauth [*principal*]
Obtain remote tickets.

Command arguments which have embedded spaces may be quoted with quote '"' marks.

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending a ftp protocol ABOR command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for ABOR processing. If the remote server does not support the ABOR command, an `ftp>` prompt will not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence will be ignored when **ftp** has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described above, or from unexpected behavior by the remote server, including violations of the ftp protocol. If the delay results from unexpected remote server behavior, the local **ftp** program must be killed by hand.

FILE NAMING CONVENTIONS

Files specified as arguments to **ftp** commands are processed according to the following rules.

1. If the file name '**-**' is specified, the *stdin* (for reading) or *stdout* (for writing) is used.
2. If the first character of the file name is '|', the remainder of the argument is interpreted as a shell command. **Ftp** then forks a shell, using popen(3) with the argument supplied, and reads (writes) from the stdout (stdin). If the shell command includes spaces, the argument must be quoted; e.g. "" ls -lt"". A particularly useful example of this mechanism is: "dir more".
3. Failing the above checks, if ''globbing'' is enabled, local file names are expanded according to the rules used in the csh(1); c.f. the **glob** command. If the **ftp** command expects a single local file (.e.g. **put**), only the first filename generated by the "globbing" operation is used.
4. For **mget** commands and **get** commands with unspecified local file names, the local filename is the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.
5. For **mput** commands and **put** commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters which may affect a file transfer. The **type** may be one of "ascii", "image" (binary), "ebcdic", and "local byte size" (for PDP-10's and PDP-20's mostly). **Ftp** supports the ascii and image types of file transfer, plus local byte size 8 for **tenex** mode transfers.

Ftp supports only the default values for the remaining file transfer parameters: **mode**, **form**, and **struct**.

THE .netrc FILE

The `.netrc` file contains login and initialization information used by the auto-login process. It resides in the user's home directory. The following tokens are recognized; they may be separated by spaces, tabs, or new-lines:

machine *name*
Identify a remote machine *name*. The auto-login process searches the `.netrc` file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent `.netrc` tokens are processed, stopping when the end of file is reached or another **machine** or a **default** token is encountered.

default This is the same as **machine** *name* except that **default** matches any name. There can be only one **default** token, and it must be after all **machine** tokens. This is normally used as:

```
default login anonymous password user@site
```

thereby giving the user *automatic* anonymous ftp login to machines not specified in `.netrc`. This can be overridden by using the **-n** flag to disable auto-login.

login *name*
Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified *name*.

password *string*
Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the `.netrc` file for any user other than *anonymous*, **ftp** will abort the auto-login process if the `.netrc` is readable by anyone besides the user.

account *string*
Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an `ACCT` command if it does not.

macdef *name*
Define a macro. This token functions like the **ftp macdef** command functions. A macro is defined with the specified name; its contents begin with the next `.netrc` line and continue until a null line (consecutive new-line characters) is encountered. If a macro named **init** is defined, it is automatically executed as the last step in the auto-login process.

ENVIRONMENT

Ftp uses the following environment variables.

`HOME` For default location of a `.netrc` file, if one exists.

`SHELL` For default shell.

SEE ALSO

`ftpd`(8)

RFC2228.

HISTORY

The **ftp** command appeared in 4.2BSD.

BUGS

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

NAME

ftp — Internet file transfer program

SYNOPSIS

ftp [**-46AadefginpRtVv**] [**-N** *netrc*] [**-o** *output*] [**-P** *port*] [**-q** *quittime*]
[**-r** *retry*] [**-s** *srcaddr*] [**-T** *dir*,*max*[,*inc*]] [[*user*@]*host* [*port*]]
[[*user*@]*host*:[*path*][/]] [file:///*path*]
[ftp://[*user*[:*password*]@]*host*[:*port*]/*path*[/][;type=*X*]]
[http://[*user*[:*password*]@]*host*[:*port*]/*path*] [...]
ftp -u *URL file* [...]

DESCRIPTION

ftp is the user interface to the Internet standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

The last five arguments will fetch a file using the FTP or HTTP protocols, or by direct copying, into the current directory. This is ideal for scripts. Refer to **AUTO-FETCHING FILES** below for more information.

Options may be specified at the command line, or to the command interpreter.

-4 Forces **ftp** to only use IPv4 addresses.

-6 Forces **ftp** to only use IPv6 addresses.

-A Force active mode ftp. By default, **ftp** will try to use passive mode ftp and fall back to active mode if passive is not supported by the server. This option causes **ftp** to always use an active connection. It is only useful for connecting to very old servers that do not implement passive mode properly.

-a Causes **ftp** to bypass normal login procedure, and use an anonymous login instead.

-d Enables debugging.

-e Disables command line editing. This is useful for Emacs ange-ftp mode.

-f Forces a cache reload for transfers that go through the FTP or HTTP proxies.

-g Disables file name globbing.

-i Turns off interactive prompting during multiple file transfers.

-N *netrc* Use *netrc* instead of ~/.netrc. Refer to **THE .netrc FILE** for more information.

-n Restrains **ftp** from attempting "auto-login" upon initial connection for non auto-fetch transfers. If auto-login is enabled, **ftp** will check the .netrc (see below) file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login. To override the auto-login for auto-fetch transfers, specify the username (and optionally, password) as appropriate.

-o *output*
When auto-fetching files, save the contents in *output*. *output* is parsed according to the **FILE NAMING CONVENTIONS** below. If *output* is not '-' or doesn't start with '|', then only the first file specified will be retrieved into *output*; all other files will be retrieved into the basename of their remote name.

-P *port* Sets the port number to *port*.

-p Enable passive mode operation for use behind connection filtering firewalls. This option has been deprecated as **ftp** now tries to use passive mode by default, falling back to active mode if the server does not support passive connections.

-q *quittime*
Quit if the connection has stalled for *quittime* seconds.

-R Restart all non-proxied auto-fetches.

-r *wait* Retry the connection attempt if it failed, pausing for *wait* seconds.

-s *srcaddr*
Uses *srcaddr* as the local IP address for all connections.

-t Enables packet tracing.

-T *direction*,*maximum*[,*increment*]
Set the maximum transfer rate for *direction* to *maximum* bytes/second, and if specified, the increment to *increment* bytes/second. Refer to **rate** for more information.

-u *URL file* [...]
Upload files on the command line to *URL* where *URL* is one of the ftp URL types as supported by auto-fetch (with an optional target filename for single file uploads), and *file* is one or more local files to be uploaded.

-V Disable **verbose** and **progress**, overriding the default of enabled when output is to a terminal.

-v Enable **verbose** and **progress**. This is the default if output is to a terminal (and in the case of **progress**, **ftp** is the foreground process). Forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics.

The client host with which **ftp** is to communicate may be specified on the command line. If this is done, **ftp** will immediately attempt to establish a connection to an FTP server on that host; otherwise, **ftp** will enter its command interpreter and await instructions from the user. When **ftp** is awaiting commands from the user the prompt `ftp>` is provided to the user. The following commands are recognized by **ftp**:

! [*command* [*args*]]
Invoke an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any **ntrans** or **nmap** setting. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

ascii Set the file transfer **type** to network ASCII. This is the default type.

bell Arrange that a bell be sounded after each file transfer command is completed.

binary Set the file transfer **type** to support binary image transfer.

bye Terminate the FTP session with the remote server and exit **ftp**. An end of file will also terminate the session and exit.

case Toggle remote computer file name case mapping during **get**, **mget** and **mput** commands. When **case** is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.

cd *remote-directory*
Change the working directory on the remote machine to *remote-directory*.

cdup Change the remote machine working directory to the parent of the current remote machine working directory.

chmod *mode remote-file*
Change the permission modes of the file *remote-file* on the remote system to *mode*.

close Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

cr Toggle carriage return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non−UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete the file *remote-file* on the remote machine.

dir [*remote-path* [*local-file*]]
Print a listing of the contents of a directory on the remote machine. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems will produce output from the command `ls -l`. If *remote-path* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **dir** output. If no local file is specified, or if *local-file* is '**-**', the output is sent to the terminal.

disconnect
A synonym for **close**.

edit Toggle command line editing, and context sensitive command and file completion. This is automatically enabled if input is from a terminal, and disabled otherwise.

epsv epsv4 epsv6
Toggle the use of the extended EPSV and EPRT commands on all IP, IPv4, and IPv6 connections respectively. First try EPSV / EPRT, and then PASV / PORT. This is enabled by default. If an extended command fails then this option will be temporarily disabled for the duration of the current connection, or until **epsv**, **epsv4**, or **epsv6** is executed again.

exit A synonym for **bye**.

features Display what features the remote server supports (using the FEAT command).

fget *localfile*
Retrieve the files listed in *localfile*, which has one line per filename.

form *format*
Set the file transfer **form** to *format*. The default (and only supported) format is “non-print”.

ftp *host* [*port*]
A synonym for **open**.

ftp_debug [*ftp_debug-value*]
Toggle debugging mode. If an optional *ftp_debug-value* is specified it is used to set the debugging level. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string `-->`.

gate [*host* [*port*]]
Toggle gate-ftp mode, which used to connect through the TIS FWTK and Gauntlet ftp proxies. This will not be permitted if the gate-ftp server hasn't been set (either explicitly by the user, or from the FTPSERVER environment variable). If *host* is given, then gate-ftp mode will be enabled, and the gate-ftp server will be set to *host*. If *port* is also given, that will be used as the port to connect to on the gate-ftp server.

get *remote-file* [*local-file*]
Retrieve the *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. The current settings for **type**, **form**, **mode**, and **structure** are used while transferring the file.

glob Toggle filename expansion for **mdelete**, **mget**, **mput**, and **mreget**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in csh(1). For **mdelete**, **mget**, and **mreget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file: the exact result depends on the foreign operating system and ftp server, and can be previewed by doing `mls remote-files -` Note: **mget**, **mput** and **mreget** are not meant to transfer entire directory subtrees of files. That can be done by transferring a tar(1) archive of the subtree (in binary mode).

hash [*size*]
Toggle hash-sign (‘#’) printing for each data block transferred. The size of a data block defaults to 1024 bytes. This can be changed by specifying *size* in bytes. Enabling **hash** disables **progress**.

help [*command*]
Print an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of the known commands.

idle [*seconds*]
Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, the current inactivity timer is printed.

image A synonym for **binary**.

lcd [*directory*]
Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

less *file*
A synonym for **page**.

lpage *local-file*
Display *local-file* with the program specified by the **set pager** option.

lpwd Print the working directory on the local machine.

ls [*remote-path* [*local-file*]]
A synonym for **dir**.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macro names can be a maximum of 8 characters. Macros are only applicable to the current session they are defined within (or if defined outside a session, to the session invoked with the next **open** command), and remain defined until a **close** command is executed. To invoke a macro, use the **$** command (see above).

The macro processor interprets '$' and '\' as special characters. A '$' followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A '$' followed by an 'i' signals the macro processor that the executing macro is to be looped. On the first pass "$i" is replaced by the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A '\' followed by any character is replaced by that character. Use the '\' to prevent special treatment of the '$'.

mdelete [*remote-files*]
Delete the *remote-files* on the remote machine.

mdir *remote-files local-file*
Like **dir**, except multiple remote files may be specified. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget *remote-files*
Expand the *remote-files* on the remote machine and do a **get** for each file name thus produced. See **glob** for details on the filename expansion. Resulting file names will then be processed according to **case**, **ntrans**, and **nmap** settings. Files are transferred into the local working directory, which can be changed with `lcd directory`; new local directories can be created with `! mkdir directory`.

mkdir *directory-name*
Make a directory on the remote machine.

mls *remote-files local-file*
Like **ls**, except multiple remote files may be specified, and the *local-file* must be specified. If interactive prompting is on, **ftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mls** output.

mlsd [*remote-path*]
Display the contents of *remote-path* (which should default to the current directory if not given) in a machine-parsable form, using `MLSD`. The format of display can be changed with 'remopts mlst ...'.

mlst [*remote-path*]
Display the details about *remote-path* (which should default to the current directory if not given) in a machine-parsable form, using `MLST`. The format of display can be changed with 'remopts mlst ...'.

mode *mode-name*
Set the file transfer **mode** to *mode-name*. The default (and only supported) mode is “stream”.

modtime *remote-file*
Show the last modification time of the file on the remote machine, in RFC2822 format.

more *file*
A synonym for **page**.

mput *local-files*
Expand wild cards in the list of local files given as arguments and do a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting file names will then be processed according to **ntrans** and **nmap** settings.

mreget *remote-files*
As per **mget**, but performs a **reget** instead of **get**.

msend *local-files*
A synonym for **mput**.

newer *remote-file* [*local-file*]
Get the file only if the modification time of the remote file is more recent that the file on the current system. If the file does not exist on the current system, the remote file is considered **newer**. Otherwise, this command is identical to *get*.

nlist [*remote-path* [*local-file*]]
A synonym for **ls**.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non−UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. [*Inpattern*] is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences “$1”, “$2”, ... “$9” in *inpattern*. Use ‘\’ to prevent this special treatment of the ‘$’ character. All other characters are treated literally, and are used to determine the **nmap** [*inpattern*] variable values. For example, given *inpattern* $1.$2 and the remote file name "mydata.data", $1 would have the value "mydata", and $2 would have the value "data". The *outpattern* determines the resulting mapped filename. The sequences “$1”, “$2”, ... “$9” are replaced by any value resulting from the *inpattern* template. The sequence “$0” is replaced by the original filename. Additionally, the sequence “[*seq1*, *seq2*]” is replaced by [*seq1*] if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command

```
nmap $1.$2.$3 [$1,$2].[$2,file]
```

would yield the output filename "myfile.data" for input filenames "myfile.data" and "myfile.data.old", "myfile.file" for the input filename "myfile", and "myfile.myfile" for the input filename ".myfile". Spaces may be included in *outpattern*, as in the example:

```
nmap $1 sed s/  *$// > $1
```

Use the ‘\’ character to prevent special treatment of the ‘$’, ‘[’, ‘]’, and ‘,’ characters.

ntrans [*inchars* [*outchars*]]
Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non−UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

open *host* [*port*]
Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, **ftp** will attempt to contact an FTP server at that port. If the **set autologin** option is on (default), **ftp** will also attempt to automatically log the user in to the FTP server (see below).

page *file*
Retrieve **file** and display with the program specified by the **set pager** option.

passive [**auto**]
Toggle passive mode (if no arguments are given). If **auto** is given, act as if FTPMODE is set to 'auto'. If passive mode is turned on (default), **ftp** will send a PASV command for all data connections instead of a PORT command. The PASV command requests that the remote server open a port for the data connection and return the address of that port. The remote server listens on that port and the client connects to it. When using the more traditional PORT command, the client listens on a port and sends that address to the remote server, who connects back to it. Passive mode is useful when using **ftp** through a gateway router or host that controls the directionality of traffic. (Note that though FTP servers are required to support the PASV command by RFC1123, some do not.)

pdir [*remote-path*]
Perform **dir** [*remote-path*], and display the result with the program specified by the **set pager** option.

pls [*remote-path*]
Perform **ls** [*remote-path*], and display the result with the program specified by the **set pager** option.

pmlsd [*remote-path*]
Perform **mlsd** [*remote-path*], and display the result with the program specified by the **set pager** option.

preserve Toggle preservation of modification times on retrieved files.

progress Toggle display of transfer progress bar. The progress bar will be disabled for a transfer that has *local-file* as '**−**' or a command that starts with '|'. Refer to **FILE NAMING CONVENTIONS** for more information. Enabling **progress** disables **hash**.

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** will transfer all files, and any **mdelete** will delete all files.

When prompting is on, the following commands are available at a prompt:

a Answer ‘yes’ to the current file, and automatically answer ‘yes’ to any remaining files for the current command.

n Answer ‘no’, and do not transfer the file.

p Answer ‘yes’ to the current file, and turn off prompt mode (as is “prompt off” had been given).

q Terminate the current operation.

y Answer ‘yes’, and transfer the file.

? Display a help message.

Any other response will answer ‘yes’ to the current file.

proxy *ftp-command*
Execute an ftp command on a secondary control connection. This command allows simultaneous connection to two remote FTP servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command "proxy ?" to see other FTP commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** will not define new macros during the auto-login process, **close** will not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the FTP protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any **ntrans** or **nmap** settings in naming the remote file. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

pwd Print the name of the current working directory on the remote machine.

quit A synonym for **bye**.

quote *arg1 arg2 ...*
The arguments specified are sent, verbatim, to the remote FTP server.

rate *direction* [*maximum* [*increment*]]
Throttle the maximum transfer rate to *maximum* bytes/second. If *maximum* is 0, disable the throttle.

direction may be one of:
all Both directions.
get Incoming transfers.
put Outgoing transfers.

maximum can be modified on the fly by *increment* bytes (default: 1024) each time a given signal is received:

SIGUSR1 Increment *maximum* by *increment* bytes.

SIGUSR2 Decrement *maximum* by *increment* bytes. The result must be a positive number.

If *maximum* is not supplied, the current throttle rates are displayed.

Note: **rate** is not yet implemented for ascii mode transfers.

rcvbuf *size*
Set the size of the socket receive buffer to *size*.

recv *remote-file* [*local-file*]
A synonym for **get**.

reget *remote-file* [*local-file*]
reget acts like **get**, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that are prone to dropping connections.

remopts *command* [*command-options*]
Set options on the remote FTP server for *command* to *command-options* (whose absence is handled on a command-specific basis). Remote FTP commands known to support options include: 'MLST' (used for MLSD and MLST).

rename [*from* [*to*]]
Rename the file *from* on the remote machine, to the file *to*.

reset Clear reply queue. This command re-synchronizes command/reply sequencing with the remote FTP server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

restart *marker*
Restart the immediately following **get** or **put** at the indicated *marker*. On UNIX systems, marker is usually a byte offset into the file.

rhelp [*command-name*]
Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

rmdir *directory-name*
Delete a directory on the remote machine.

rstatus [*remote-file*]
With no arguments, show status of remote machine. If *remote-file* is specified, show status of *remote-file* on remote machine.

runique Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a ".1" is appended to the name. If the resulting name matches another existing file, a ".2" is appended to the original name. If this process continues up to ".99", an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that **runique** will not affect local files generated from a shell command (see below). The default value is off.

send *local-file* [*remote-file*]
A synonym for **put**.

sendport Toggle the use of PORT commands. By default, **ftp** will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, **ftp** will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer. This is useful for certain FTP implementations which do ignore PORT commands but, incorrectly, indicate they've been accepted.

set [*option value*]
Set *option* to *value*. If *option* and *value* are not given, display all of the options and their values. The currently supported options are:

anonpass Defaults to $FTPANONPASS

ftp_proxy Defaults to $ftp_proxy.

http_proxy Defaults to $http_proxy.

no_proxy Defaults to $no_proxy.

pager Defaults to $PAGER.

prompt Defaults to $FTPPROMPT.

rprompt Defaults to $FTPRPROMPT.

site *arg1 arg2 ...*
The arguments specified are sent, verbatim, to the remote FTP server as a SITE command.

size *remote-file*
Return size of *remote-file* on remote machine.

sndbuf *size*
Set the size of the socket send buffer to *size*.

status Show the current status of **ftp**.

struct *struct-name*
Set the file transfer *structure* to *struct-name*. The default (and only supported) structure is "file".

sunique Toggle storing of files on remote machine under unique file names. The remote FTP server must support FTP protocol STOU command for successful completion. The remote server will report unique name. Default value is off.

system Show the type of operating system running on the remote machine.

tenex Set the file transfer type to that needed to talk to TENEX machines.

throttle A synonym for **rate**.

trace Toggle packet tracing.

type [*type-name*]
Set the file transfer **type** to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

umask [*newmask*]
Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

unset *option*
Unset *option*. Refer to **set** for more information.

usage *command*
Print the usage message for *command*.

user *user-name* [*password* [*account*]]
Identify yourself to the remote FTP server. If the *password* is not specified and the server requires it, **ftp** will prompt the user for it (after disabling local echo). If an *account* field is not specified, and the FTP server requires it, the user will be prompted for it. If an *account*

field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless **ftp** is invoked with "auto-login" disabled, this process is done automatically on initial connection to the FTP server.

verbose Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

xferbuf *size*
Set the size of the socket send and receive buffers to *size*.

? [*command*]
A synonym for **help**.

Command arguments which have embedded spaces may be quoted with quote '"' marks.

Commands which toggle settings can take an explicit **on** or **off** argument to force the setting appropriately.

Commands which take a byte count as an argument (e.g., **hash**, **rate**, and **xferbuf**) support an optional suffix on the argument which changes the interpretation of the argument. Supported suffixes are:

- b Causes no modification. (Optional)
- k Kilo; multiply the argument by 1024
- m Mega; multiply the argument by 1048576
- g Giga; multiply the argument by 1073741824

If **ftp** receives a SIGINFO (see the "status" argument of stty(1)) or SIGQUIT signal whilst a transfer is in progress, the current transfer rate statistics will be written to the standard error output, in the same format as the standard completion message.

AUTO-FETCHING FILES

In addition to standard commands, this version of **ftp** supports an auto-fetch feature. To enable auto-fetch, simply pass the list of hostnames/files on the command line.

The following formats are valid syntax for an auto-fetch element:

[*user*@]*host*:[*path*][/]
"Classic" FTP format.

If *path* contains a glob character and globbing is enabled, (see **glob**), then the equivalent of mget path is performed.

If the directory component of *path* contains no globbing characters, it is stored locally with the name basename (see basename(1)) of **path**, in the current directory. Otherwise, the full remote name is used as the local name, relative to the local root directory.

ftp://[*user*[:*password*]@]*host*[:*port*]/*path*[/][;type=*X*]
An FTP URL, retrieved using the FTP protocol if **set ftp_proxy** isn't defined. Otherwise, transfer the URL using HTTP via the proxy defined in **set ftp_proxy**. If **set ftp_proxy** isn't defined and *user* is given, login as *user*. In this case, use *password* if supplied, otherwise prompt the user for one.

If a suffix of ';type=A' or ';type=I' is supplied, then the transfer type will take place as ascii or binary (respectively). The default transfer type is binary.

In order to be compliant with RFC3986, **ftp** interprets the *path* part of an "ftp://" auto-fetch URL as follows:

- The '/' immediately after the *host*[:*port*] is interpreted as a separator before the *path*, and not as part of the *path* itself.
- The *path* is interpreted as a '/'-separated list of name components. For all but the last such component, **ftp** performs the equivalent of a **cd** command. For the last path component, **ftp** performs the equivalent of a **get** command.
- Empty name components, which result from '//' within the *path*, or from an extra '/' at the beginning of the *path*, will cause the equivalent of a **cd** command without a directory name. This is unlikely to be useful.
- Any '%*XX*' codes (per RFC3986) within the path components are decoded, with *XX* representing a character code in hexadecimal. This decoding takes place after the *path* has been split into components, but before each component is used in the equivalent of a **cd** or **get** command. Some often-used codes are '%2F' (which represents '/') and '%7E' (which represents '~').

The above interpretation has the following consequences:

- The path is interpreted relative to the default login directory of the specified user or of the 'anonymous' user. If the / directory is required, use a leading path of "%2F". If a user's home directory is required (and the remote server supports the syntax), use a leading path of "%7Euser/". For example, to retrieve /etc/motd from 'localhost' as the user 'myname' with the password 'mypass', use "ftp://myname:mypass@localhost/%2fetc/motd"
- The exact **cd** and **get** commands can be controlled by careful choice of where to use '/' and where to use '%2F' (or '%2f'). For example, the following URLs correspond to the equivalents of the indicated commands:

ftp://host/dir1/dir2/file	"cd dir1", "cd dir2", "get file".
ftp://host/%2Fdir1/dir2/file	"cd /dir1", "cd dir2", "get file".
ftp://host/dir1%2Fdir2/file	"cd dir1/dir2", "get file".
ftp://host/%2Fdir1%2Fdir2/file	"cd /dir1/dir2", "get file".
ftp://host/dir1%2Fdir2%2Ffile	"get dir1/dir2/file".
ftp://host/%2Fdir1%2Fdir2%2Ffile	"get /dir1/dir2/file".

- You must have appropriate access permission for each of the intermediate directories that is used in the equivalent of a **cd** command.

http://[*user*[:*password*]@]*host*[:*port*]/*path*
: An HTTP URL, retrieved using the HTTP protocol. If **set http_proxy** is defined, it is used as a URL to an HTTP proxy server. If HTTP authorization is required to retrieve *path*, and 'user' (and optionally 'password') is in the URL, use them for the first attempt to authenticate.

file:///*path*
: A local URL, copied from /*path* on the local host.

about:*topic*
: Display information regarding *topic*; no file is retrieved for this auto-fetched element. Supported values include:

  about:ftp Information about **ftp**.

  about:version The version of **ftp**. Useful to provide when reporting problems.

Unless noted otherwise above, and **-o** *output* is not given, the file is stored in the current directory as the basename(1) of *path*. Note that if a HTTP redirect is received, the fetch is retried using the new target

URL supplied by the server, with a corresponding new *path*. Using an explicit **-o** *output* is recommended, to avoid writing to unexpected file names.

If a classic format or an FTP URL format has a trailing '/' or an empty *path* component, then **ftp** will connect to the site and **cd** to the directory given as the path, and leave the user in interactive mode ready for further input. This will not work if **set ftp_proxy** is being used.

Direct HTTP transfers use HTTP 1.1. Proxied FTP and HTTP transfers use HTTP 1.0.

If **-R** is given, all auto-fetches that don't go via the FTP or HTTP proxies will be restarted. For FTP, this is implemented by using **reget** instead of **get**. For HTTP, this is implemented by using the 'Range: bytes=' HTTP/1.1 directive.

If WWW or proxy WWW authentication is required, you will be prompted to enter a username and password to authenticate with.

When specifying IPv6 numeric addresses in a URL, you need to surround the address in square brackets. E.g.: "ftp://[::1]:21/". This is because colons are used in IPv6 numeric address as well as being the separator for the port number.

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending an FTP protocol ABOR command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for ABOR processing. If the remote server does not support the ABOR command, the prompt will not appear until the remote server has completed sending the requested file.

If the terminal interrupt key sequence is used whilst **ftp** is awaiting a reply from the remote server for the ABOR processing, then the connection will be closed. This is different from the traditional behaviour (which ignores the terminal interrupt during this phase), but is considered more useful.

FILE NAMING CONVENTIONS

Files specified as arguments to **ftp** commands are processed according to the following rules.

1. If the file name '**-**' is specified, the *stdin* (for reading) or *stdout* (for writing) is used.
2. If the first character of the file name is '|', the remainder of the argument is interpreted as a shell command. **ftp** then forks a shell, using popen(3) with the argument supplied, and reads (writes) from the stdout (stdin). If the shell command includes spaces, the argument must be quoted; e.g. "`"| ls -lt"`". A particularly useful example of this mechanism is: "`dir "" |more`".
3. Failing the above checks, if "globbing" is enabled, local file names are expanded according to the rules used in the csh(1); see the **glob** command. If the **ftp** command expects a single local file (e.g. **put**), only the first filename generated by the "globbing" operation is used.
4. For **mget** commands and **get** commands with unspecified local file names, the local filename is the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.
5. For **mput** commands and **put** commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters which may affect a file transfer. The **type** may be one of "ascii", "image" (binary), "ebcdic", and "local byte size" (for PDP-10's and PDP-20's mostly). **ftp** supports the ascii and image types of file transfer, plus local byte size 8 for **tenex** mode transfers.

ftp supports only the default values for the remaining file transfer parameters: **mode**, **form**, and **struct**.

THE .netrc FILE

The .netrc file contains login and initialization information used by the auto-login process. It resides in the user's home directory, unless overridden with the **-N** *netrc* option, or specified in the NETRC environment variable. The following tokens are recognized; they may be separated by spaces, tabs, or new-lines:

machine *name*
: Identify a remote machine *name*. The auto-login process searches the .netrc file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent .netrc tokens are processed, stopping when the end of file is reached or another **machine** or a **default** token is encountered.

default
: This is the same as **machine** *name* except that **default** matches any name. There can be only one **default** token, and it must be after all **machine** tokens. This is normally used as:

```
default login anonymous password user@site
```

thereby giving the user an automatic anonymous FTP login to machines not specified in .netrc. This can be overridden by using the **-n** flag to disable auto-login.

login *name*
: Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified *name*.

password *string*
: Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the .netrc file for any user other than *anonymous*, **ftp** will abort the auto-login process if the .netrc is readable by anyone besides the user.

account *string*
: Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

macdef *name*
: Define a macro. This token functions like the **ftp** **macdef** command functions. A macro is defined with the specified name; its contents begin with the next .netrc line and continue until a blank line (consecutive new-line characters) is encountered. Like the other tokens in the .netrc file, a **macdef** is applicable only to the **machine** definition preceding it. A **macdef** entry cannot be used by multiple **machine** definitions; rather, it must be defined following each **machine** it is intended to be used with. If a macro named **init** is defined, it is automatically executed as the last step in the auto-login process. For example,

```
default
macdef init
epsv4 off
```

followed by a blank line.

COMMAND LINE EDITING

ftp supports interactive command line editing, via the editline(3) library. It is enabled with the **edit** command, and is enabled by default if input is from a tty. Previous lines can be recalled and edited with the arrow keys, and other GNU Emacs-style editing keys may be used as well.

The editline(3) library is configured with a .editrc file - refer to editrc(5) for more information.

An extra key binding is available to **ftp** to provide context sensitive command and filename completion (including remote file completion). To use this, bind a key to the editline(3) command **ftp-complete**. By default, this is bound to the TAB key.

COMMAND LINE PROMPT

By default, **ftp** displays a command line prompt of "ftp> " to the user. This can be changed with the **set prompt** command.

A prompt can be displayed on the right side of the screen (after the command input) with the **set rprompt** command.

The following formatting sequences are replaced by the given information:

`%/` The current remote working directory.

`%c[[0]`*n*`],%.[[0]`*n*`]`
The trailing component of the current remote working directory, or *n* trailing components if a digit *n* is given. If *n* begins with '0', the number of skipped components precede the trailing component(s) in the format "`/<`*number*`>`*trailing*" (for '%c') or "`...`*trailing*" (for '%.').

`%M` The remote host name.

`%m` The remote host name, up to the first '.'.

`%n` The remote user name.

`%%` A single '%'.

ENVIRONMENT

ftp uses the following environment variables.

FTPANONPASS	Password to send in an anonymous FTP transfer. Defaults to "`'whoami'@`".
FTPMODE	Overrides the default operation mode. Support values are:
	active active mode FTP only
	auto automatic determination of passive or active (this is the default)
	gate gate-ftp mode
	passive passive mode FTP only
FTPPROMPT	Command-line prompt to use. Defaults to "ftp> ". Refer to **COMMAND LINE PROMPT** for more information.
FTPRPROMPT	Command-line right side prompt to use. Defaults to "". Refer to **COMMAND LINE PROMPT** for more information.
FTPSERVER	Host to use as gate-ftp server when **gate** is enabled.
FTPSERVERPORT	Port to use when connecting to gate-ftp server when **gate** is enabled. Default is port returned by a **getservbyname**() lookup of "ftpgate/tcp".
FTPUSERAGENT	The value to send for the HTTP User-Agent header.
HOME	For default location of a .netrc file, if one exists.
NETRC	An alternate location of the .netrc file.

`PAGER` Used by various commands to display files. Defaults to `more`(1) if empty or not set.

`SHELL` For default shell.

`ftp_proxy` URL of FTP proxy to use when making FTP URL requests (if not defined, use the standard FTP protocol).

See `http_proxy` for further notes about proxy use.

`http_proxy` URL of HTTP proxy to use when making HTTP URL requests. If proxy authentication is required and there is a username and password in this URL, they will automatically be used in the first attempt to authenticate to the proxy.

If "unsafe" URL characters are required in the username or password (for example '@' or '/'), encode them with `RFC3986` '*%XX*' encoding.

Note that the use of a username and password in `ftp_proxy` and `http_proxy` may be incompatible with other programs that use it (such as `lynx`(1)).

NOTE: this is not used for interactive sessions, only for command-line fetches.

`no_proxy` A space or comma separated list of hosts (or domains) for which proxying is not to be used. Each entry may have an optional trailing ":port", which restricts the matching to connections to that port.

EXTENDED PASSIVE MODE AND FIREWALLS

Some firewall configurations do not allow **`ftp`** to use extended passive mode. If you find that even a simple **`ls`** appears to hang after printing a message such as this:

```
229 Entering Extended Passive Mode (|||58551|)
```

then you will need to disable extended passive mode with **`epsv4 off`**. See the above section **The .netrc File** for an example of how to make this automatic.

SEE ALSO

`getservbyname`(3), `editrc`(5), `services`(5), `ftpd`(8)

STANDARDS

`ftp` attempts to be compliant with:

`RFC0959`
File Transfer Protocol

`RFC1123`
Requirements for Internet Hosts - Application and Support

`RFC1635`
How to Use Anonymous FTP

`RFC2389`
Feature negotiation mechanism for the File Transfer Protocol

`RFC2428`
FTP Extensions for IPv6 and NATs

`RFC2616`
Hypertext Transfer Protocol -- HTTP/1.1

RFC2822
Internet Message Format

RFC3659
Extensions to FTP

RFC3986
Uniform Resource Identifier (URI)

HISTORY

The **ftp** command appeared in 4.2BSD.

Various features such as command line editing, context sensitive command and file completion, dynamic progress bar, automatic fetching of files and URLs, modification time preservation, transfer rate throttling, configurable command line prompt, and other enhancements over the standard BSD **ftp** were implemented in NetBSD 1.3 and later releases by Luke Mewburn ⟨lukem@NetBSD.org⟩.

IPv6 support was added by the WIDE/KAME project (but may not be present in all non-NetBSD versions of this program, depending if the operating system supports IPv6 in a similar manner to KAME).

BUGS

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

ftp assumes that all IPv4 mapped addresses (IPv6 addresses with a form like `::ffff:10.1.1.1`) indicate IPv4 destinations which can be handled by `AF_INET` sockets. However, in certain IPv6 network configurations, this assumption is not true. In such an environment, IPv4 mapped addresses must be passed to `AF_INET6` sockets directly. For example, if your site uses a SIIT translator for IPv6-to-IPv4 translation, **ftp** is unable to support your configuration.

NAME

gcc − GNU project C and C++ compiler

SYNOPSIS

gcc [**−c**|**−S**|**−E**] [**−std=***standard*]
[**−g**] [**−pg**] [**−O***level*]
[**−W***warn*...] [**−pedantic**]
[**−I***dir*...] [**−L***dir*...]
[**−D***macro*[*=defn*]...] [**−U***macro*]
[**−f***option*...] [**−m***machine-option*...]
[**−o** *outfile*] *infile*...

Only the most useful options are listed here; see below for the remainder. **g++** accepts mostly the same options as **gcc**.

DESCRIPTION

When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The ‘‘overall options’’ allow you to stop this process at an intermediate stage. For example, the **−c** option says not to run the linker. Then the output consists of object files output by the assembler.

Other options are passed on to one stage of processing. Some options control the preprocessor and others the compiler itself. Yet other options control the assembler and linker; most of these are not documented here, since you rarely need to use any of them.

Most of the command line options that you can use with GCC are useful for C programs; when an option is only useful with another language (usually C++), the explanation says so explicitly. If the description for a particular option does not mention a source language, you can use that option with all supported languages.

The **gcc** program accepts options and file names as operands. Many options have multi-letter names; therefore multiple single-letter options may *not* be grouped: **−dr** is very different from **−d −r**.

You can mix options and other arguments. For the most part, the order you use doesn’t matter. Order does matter when you use several options of the same kind; for example, if you specify **−L** more than once, the directories are searched in the order specified.

Many options have long names starting with **−f** or with **−W**−−−for example, **−fstrength−reduce**, **−Wformat** and so on. Most of these have both positive and negative forms; the negative form of **−ffoo** would be **−fno−foo**. This manual documents only one of these two forms, whichever one is not the default.

OPTIONS

Option Summary

Here is a summary of all the options, grouped by type. Explanations are in the following sections.

Overall Options

−c −S −E −o *file* **−combine −pipe −pass−exit−codes −x** *language* **−v −### −−help −−target−help −−version**

C Language Options

−ansi −std=*standard* **−aux−info** *filename* **−fno−asm −fno−builtin −fno−builtin−***function* **−fhosted −ffreestanding −fms−extensions −trigraphs −no−integrated−cpp −traditional −traditional−cpp −fallow−single−precision −fcond−mismatch −fsigned−bitfields −fsigned−char −funsigned−bitfields −funsigned−char**

C++ Language Options

−fabi−version=*n* **−fno−access−control −fcheck−new −fconserve−space −ffriend−injection −fno−const−strings −fno−elide−constructors −fno−enforce−eh−specs −ffor−scope −fno−for−scope −fno−gnu−keywords −fno−implicit−templates −fno−implicit−inline−templates −fno−implement−inlines −fms−extensions −fno−nonansi−builtins −fno−operator−names −fno−optional−diags −fpermissive −frepo −fno−rtti −fstats −ftemplate−depth−***n* **−fno−threadsafe−statics −fuse−cxa−atexit −fno−weak −nostdinc++ −fno−default−inline −fvisibility−inlines−hidden −Wabi −Wctor−dtor−privacy −Wnon−virtual−dtor −Wreorder −Weffc++**

−Wno−deprecated −Wstrict−null−sentinel −Wno−non−template−friend −Wold−style−cast −Woverloaded−virtual −Wno−pmf−conversions −Wsign−promo

Objective-C and Objective−C++ Language Options

−fconstant−string−class=*class-name* **−fgnu−runtime −fnext−runtime −fno−nil−receivers −fobjc−call−cxx−cdtors −fobjc−direct−dispatch −fobjc−exceptions −fobjc−gc −freplace−objc−classes −fzero−link −gen−decls −Wassign−intercept −Wno−protocol −Wselector −Wstrict−selector−match −Wundeclared−selector**

Language Independent Options

−fmessage−length=*n* **−fdiagnostics−show−location=**[**once**|**every-line**] −fdiagnostics−show−options

Warning Options

−fsyntax−only −pedantic −pedantic−errors −w −Wextra −Wall −Waggregate−return −Wno−attributes −Wc++−compat −Wcast−align −Wcast−qual −Wchar−subscripts −Wcomment −Wconversion −Wno−deprecated−declarations −Wdisabled−optimization −Wno−div−by−zero −Wno−endif−labels −Werror −Werror−implicit−function−declaration −Wfatal−errors −Wfloat−equal −Wformat −Wformat=2 −Wno−format−extra−args −Wformat−nonliteral −Wformat−security −Wformat−y2k −Wimplicit −Wimplicit−function−declaration −Wimplicit−int −Wimport −Wno−import −Winit−self −Winline −Wno−int−to−pointer−cast −Wno−invalid−offsetof −Winvalid−pch −Wlarger−than−*len* **−Wunsafe−loop−optimizations −Wlong−long −Wmain −Wmissing−braces −Wmissing−field−initializers −Wmissing−format−attribute −Wmissing−include−dirs −Wmissing−noreturn −Wno−multichar −Wnonnull −Wpacked −Wpadded −Wparentheses −Wpointer−arith −Wno−pointer−to−int−cast −Wredundant−decls −Wreturn−type −Wsequence−point −Wshadow −Wsign−compare −Wstack−protector −Wstrict−aliasing −Wstrict−aliasing=2 −Wswitch −Wswitch−default −Wswitch−enum −Wsystem−headers −Wtrigraphs −Wundef −Wuninitialized −Wunknown−pragmas −Wno−pragmas −Wunreachable−code −Wunused −Wunused−function −Wunused−label −Wunused−parameter −Wunused−value −Wunused−variable −Wvariadic−macros −Wvolatile−register−var −Wwrite−strings**

C−only Warning Options

−Wbad−function−cast −Wmissing−declarations −Wmissing−prototypes −Wnested−externs −Wold−style−definition −Wstrict−prototypes −Wtraditional −Wdeclaration−after−statement −Wpointer−sign

Debugging Options

−d[*letters*] **−dumpspecs −dumpmachine −dumpversion −fdump−unnumbered −fdump−translation−unit**[*−n*] **−fdump−class−hierarchy**[*−n*] **−fdump−ipa−all −fdump−ipa−cgraph −fdump−tree−all −fdump−tree−original**[*−n*] **−fdump−tree−optimized**[*−n*] **−fdump−tree−inlined**[*−n*] **−fdump−tree−cfg −fdump−tree−vcg −fdump−tree−alias −fdump−tree−ch −fdump−tree−ssa**[*−n*] **−fdump−tree−pre**[*−n*] **−fdump−tree−ccp**[*−n*] **−fdump−tree−dce**[*−n*] **−fdump−tree−gimple**[**−raw**] **−fdump−tree−mudflap**[*−n*] **−fdump−tree−dom**[*−n*] **−fdump−tree−dse**[*−n*] **−fdump−tree−phiopt**[*−n*] **−fdump−tree−forwprop**[*−n*] **−fdump−tree−copyrename**[*−n*] **−fdump−tree−nrv −fdump−tree−vect −fdump−tree−sink −fdump−tree−sra**[*−n*] **−fdump−tree−salias −fdump−tree−fre**[*−n*] **−fdump−tree−vrp**[*−n*] **−ftree−vectorizer−verbose=***n* **−fdump−tree−storeccp**[*−n*] **−feliminate−dwarf2−dups −feliminate−unused−debug−types −feliminate−unused−debug−symbols −fmem−report −fprofile−arcs −frandom−seed=***string* **−fsched−verbose=***n* **−ftest−coverage −ftime−report −fvar−tracking −g −g***level* **−gcoff −gdwarf−2 −ggdb −gstabs −gstabs+ −gvms −gxcoff −gxcoff+ −p −pg −print−file−name=***library* **−print−libgcc−file−name −print−multi−directory −print−multi−lib −print−prog−name=***program* **−print−search−dirs −Q −save−temps −time**

Optimization Options

−falign−functions=*n* **−falign−jumps=***n* **−falign−labels=***n* **−falign−loops=***n* **−fbounds−check −fmudflap −fmudflapth −fmudflapir −fbranch−probabilities −fprofile−values −fvpt −fbranch−target−load−optimize −fbranch−target−load−optimize2 −fbtr−bb−exclusive**

−fcaller−saves −fcprop−registers −fcse−follow−jumps −fcse−skip−blocks −fcx−limited−range −fdata−sections −fdelayed−branch −fdelete−null−pointer−checks −fearly−inlining −fexpensive−optimizations −ffast−math −ffloat−store −fforce−addr −ffunction−sections −fgcse −fgcse−lm −fgcse−sm −fgcse−las −fgcse−after−reload −floop−optimize −fcrossjumping −fif−conversion −fif−conversion2 −finline−functions −finline−functions−called−once −finline−limit=*n*** −fkeep−inline−functions −fkeep−static−consts −fmerge−constants −fmerge−all−constants −fmodulo−sched −fno−branch−count−reg −fno−default−inline −fno−defer−pop −floop−optimize2 −fmove−loop−invariants −fno−function−cse −fno−guess−branch−probability −fno−inline −fno−math−errno −fno−peephole −fno−peephole2 −funsafe−math−optimizations −funsafe−loop−optimizations −ffinite−math−only −fno−trapping−math −fno−zero−initialized−in−bss −fomit−frame−pointer −foptimize−register−move −foptimize−sibling−calls −fprefetch−loop−arrays −fprofile−generate −fprofile−use −fregmove −frename−registers −freorder−blocks −freorder−blocks−and−partition −freorder−functions −frerun−cse−after−loop −frerun−loop−opt −frounding−math −fschedule−insns −fschedule−insns2 −fno−sched−interblock −fno−sched−spec −fsched−spec−load −fsched−spec−load−dangerous −fsched−stalled−insns=***n*** −fsched−stalled−insns−dep=***n*** −fsched2−use−superblocks −fsched2−use−traces −freschedule−modulo−scheduled−loops −fsignaling−nans −fsingle−precision−constant −fstack−protector −fstack−protector−all −fstrength−reduce −fstrict−aliasing −ftracer −fthread−jumps −funroll−all−loops −funroll−loops −fpeel−loops −fsplit−ivs−in−unroller −funswitch−loops −fvariable−expansion−in−unroller −ftree−pre −ftree−ccp −ftree−dce −ftree−loop−optimize −ftree−loop−linear −ftree−loop−im −ftree−loop−ivcanon −fivopts −ftree−dominator−opts −ftree−dse −ftree−copyrename −ftree−sink −ftree−ch −ftree−sra −ftree−ter −ftree−lrs −ftree−fre −ftree−vectorize −ftree−vect−loop−version −ftree−salias −fweb −ftree−copy−prop −ftree−store−ccp −ftree−store−copy−prop −fwhole−program −−param** *name=value* **−O −O0 −O1 −O2 −O3 −Os**

Preprocessor Options

−A*question=answer* **−A−***question*[*=answer*] **−C −dD −dI −dM −dN −D***macro*[*=defn*] **−E −H −idirafter** *dir* **−include** *file* **−imacros** *file* **−iprefix** *file* **−iwithprefix** *dir* **−iwithprefixbefore** *dir* **−isystem** *dir* **−isysroot** *dir* **−M −MM −MF −MG −MP −MQ −MT −nostdinc −P −fworking−directory −remap −trigraphs −undef −U***macro* **−Wp,***option* **−Xpreprocessor** *option*

Assembler Option

−Wa,*option* **−Xassembler** *option*

Linker Options

object−file−name **−l***library* **−nostartfiles −nodefaultlibs −nostdlib −pie −rdynamic −s −static −static−libgcc −shared −shared−libgcc −symbolic −Wl,***option* **−Xlinker** *option* **−u** *symbol*

Directory Options

−B*prefix* **−I***dir* **−iquote***dir* **−L***dir* **−specs=***file* **−I− −−sysroot=***dir*

Target Options

−V *version* **−b** *machine*

Machine Dependent Options

ARC Options **−EB −EL −mmangle−cpu −mcpu=***cpu* **−mtext=***text−section* **−mdata=***data−section* **−mrodata=***readonly−data−section*

ARM Options **−mapcs−frame −mno−apcs−frame −mabi=***name* **−mapcs−stack−check −mno−apcs−stack−check −mapcs−float −mno−apcs−float −mapcs−reentrant −mno−apcs−reentrant −msched−prolog −mno−sched−prolog −mlittle−endian −mbig−endian −mwords−little−endian −mfloat−abi=***name* **−msoft−float −mhard−float −mfpe −mthumb−interwork −mno−thumb−interwork −mcpu=***name* **−march=***name* **−mfpu=***name* **−mstructure−size−boundary=***n* **−mabort−on−noreturn −mlong−calls −mno−long−calls −msingle−pic−base −mno−single−pic−base −mpic−register=***reg* **−mnop−fun−dllimport −mcirrus−fix−invalid−insns −mno−cirrus−fix−invalid−insns −mpoke−function−name −mthumb −marm −mtpcs−frame −mtpcs−leaf−frame −mcaller−super−interworking −mcallee−super−interworking −mtp=***name***

AVR Options **−mmcu=***mcu* **−msize −minit−stack=***n* **−mno−interrupts −mcall−prologues −mno−tablejump −mtiny−stack −mint8**

Blackfin Options **−momit−leaf−frame−pointer −mno−omit−leaf−frame−pointer −mspecld−anomaly −mno−specld−anomaly −mcsync−anomaly −mno−csync−anomaly −mlow−64k −mno−low64k −mid−shared−library −mno−id−shared−library −mshared−library−id=***n* **−mlong−calls −mno−long−calls**

CRIS Options **−mcpu=***cpu* **−march=***cpu* **−mtune=***cpu* **−mmax−stack−frame=***n* **−melinux−stacksize=***n* **−metrax4 −metrax100 −mpdebug −mcc−init −mno−side−effects −mstack−align −mdata−align −mconst−align −m32−bit −m16−bit −m8−bit −mno−prologue−epilogue −mno−gotplt −melf −maout −melinux −mlinux −sim −sim2 −mmul−bug−workaround −mno−mul−bug−workaround**

CRX Options **−mmac −mpush−args**

Darwin Options **−all_load −allowable_client −arch −arch_errors_fatal −arch_only −bind_at_load −bundle −bundle_loader −client_name −compatibility_version −current_version −dead_strip −dependency−file −dylib_file −dylinker_install_name −dynamic −dynamiclib −exported_symbols_list −filelist −flat_namespace −force_cpusubtype_ALL −force_flat_namespace −headerpad_max_install_names −image_base −init −install_name −keep_private_externs −multi_module −multiply_defined −multiply_defined_unused −noall_load −no_dead_strip_inits_and_terms −nofixprebinding −nomultidefs −noprebind −noseglinkedit −pagezero_size −prebind −prebind_all_twolevel_modules −private_bundle −read_only_relocs −sectalign −sectobjectsymbols −whyload −seg1addr −sectcreate −sectobjectsymbols −sectorder −segaddr −segs_read_only_addr −segs_read_write_addr −seg_addr_table −seg_addr_table_filename −seglinkedit −segprot −segs_read_only_addr −segs_read_write_addr −single_module −static −sub_library −sub_umbrella −twolevel_namespace −umbrella −undefined −unexported_symbols_list −weak_reference_mismatches −whatsloaded −F −gused −gfull −mmacosx−version−min=***version* **−mone−byte−bool**

DEC Alpha Options **−mno−fp−regs −msoft−float −malpha−as −mgas −mieee −mieee−with−inexact −mieee−conformant −mfp−trap−mode=***mode* **−mfp−rounding−mode=***mode* **−mtrap−precision=***mode* **−mbuild−constants −mcpu=***cpu-type* **−mtune=***cpu-type* **−mbwx −mmax −mfix −mcix −mfloat−vax −mfloat−ieee −mexplicit−relocs −msmall−data −mlarge−data −msmall−text −mlarge−text −mmemory−latency=***time*

DEC Alpha/VMS Options **−mvms−return−codes**

FRV Options **−mgpr−32 −mgpr−64 −mfpr−32 −mfpr−64 −mhard−float −msoft−float −malloc−cc −mfixed−cc −mdword −mno−dword −mdouble −mno−double −mmedia −mno−media −mmuladd −mno−muladd −mfdpic −minline−plt −mgprel−ro −multilib−library−pic −mlinked−fp −mlong−calls −malign−labels −mlibrary−pic −macc−4 −macc−8 −mpack −mno−pack −mno−eflags −mcond−move −mno−cond−move −moptimize−membar −mno−optimize−membar −mscc −mno−scc −mcond−exec −mno−cond−exec −mvliw−branch −mno−vliw−branch −mmulti−cond−exec −mno−multi−cond−exec −mnested−cond−exec −mno−nested−cond−exec −mtomcat−stats −mTLS −mtls −mcpu=***cpu*

H8/300 Options **−mrelax −mh −ms −mn −mint32 −malign−300**

HPPA Options **−march=***architecture-type* **−mbig−switch −mdisable−fpregs −mdisable−indexing −mfast−indirect−calls −mgas −mgnu−ld −mhp−ld −mfixed−range=***register-range* **−mjump−in−delay −mlinker−opt −mlong−calls −mlong−load−store −mno−big−switch −mno−disable−fpregs −mno−disable−indexing −mno−fast−indirect−calls −mno−gas −mno−jump−in−delay −mno−long−load−store −mno−portable−runtime −mno−soft−float −mno−space−regs −msoft−float −mpa−risc−1−0 −mpa−risc−1−1 −mpa−risc−2−0 −mportable−runtime −mschedule=***cpu-type* **−mspace−regs −msio −mwsio −munix=***unix-std* **−nolibdld −static −threads**

i386 and x86−64 Options **−mtune=***cpu-type* **−march=***cpu-type* **−mfpmath=***unit* **−masm=***dialect* **−mno−fancy−math−387 −mno−fp−ret−in−387 −msoft−float −msvr3−shlib −mno−wide−multiply −mrtd −malign−double −mpreferred−stack−boundary=***num* **−mmmx −msse −msse2 −msse3 −m3dnow −mthreads −mno−align−stringops −minline−all−stringops −mpush−args −maccumulate−outgoing−args −m128bit−long−double −m96bit−long−double −mregparm=***num* **−msseregparm −momit−leaf−frame−pointer −mno−red−zone −mno−tls−direct−seg−refs −mcmodel=***code-model* **−m32 −m64 −mlarge−data−threshold=***num*

IA−64 Options **−mbig−endian −mlittle−endian −mgnu−as −mgnu−ld −mno−pic −mvolatile−asm−stop −mregister−names −mno−sdata −mconstant−gp −mauto−pic −minline−float−divide−min−latency −minline−float−divide−max−throughput −minline−int−divide−min−latency −minline−int−divide−max−throughput −minline−sqrt−min−latency −minline−sqrt−max−throughput −mno−dwarf2−asm −mearly−stop−bits −mfixed−range=***register-range* **−mtls−size=***tls-size* **−mtune=***cpu-type* **−mt −pthread −milp32 −mlp64**

M32R/D Options **−m32r2 −m32rx −m32r −mdebug −malign−loops −mno−align−loops −missue−rate=***number* **−mbranch−cost=***number* **−mmodel=***code-size-model-type* **−msdata=***sdata-type* **−mno−flush−func −mflush−func=***name* **−mno−flush−trap −mflush−trap=***number* **−G** *num*

M32C Options **−mcpu=***cpu* **−msim −memregs=***number*

M680x0 Options **−m68000 −m68020 −m68020−40 −m68020−60 −m68030 −m68040 −m68060 −mcpu32 −m5200 −m68881 −mbitfield −mc68000 −mc68020 −mnobitfield −mrtd −mshort −msoft−float −mpcrel −malign−int −mstrict−align −msep−data −mno−sep−data −mshared−library−id=n −mid−shared−library −mno−id−shared−library**

M68hc1x Options **−m6811 −m6812 −m68hc11 −m68hc12 −m68hcs12 −mauto−incdec −minmax −mlong−calls −mshort −msoft−reg−count=***count*

MCore Options **−mhardlit −mno−hardlit −mdiv −mno−div −mrelax−immediates −mno−relax−immediates −mwide−bitfields −mno−wide−bitfields −m4byte−functions −mno−4byte−functions −mcallgraph−data −mno−callgraph−data −mslow−bytes −mno−slow−bytes −mno−lsim −mlittle−endian −mbig−endian −m210 −m340 −mstack−increment**

MIPS Options **−EL −EB −march=***arch* **−mtune=***arch* **−mips1 −mips2 −mips3 −mips4 −mips32 −mips32r2 −mips64 −mips16 −mno−mips16 −mabi=***abi* **−mabicalls −mno−abicalls −mxgot −mno−xgot −mgp32 −mgp64 −mfp32 −mfp64 −mhard−float −msoft−float −msingle−float −mdouble−float −mdsp −mpaired−single −mips3d −mlong64 −mlong32 −msym32 −mno−sym32 −G***num* **−membedded−data −mno−embedded−data −muninit−const−in−rodata −mno−uninit−const−in−rodata −msplit−addresses −mno−split−addresses −mexplicit−relocs −mno−explicit−relocs −mcheck−zero−division −mno−check−zero−division −mdivide−traps −mdivide−breaks −mmemcpy −mno−memcpy −mlong−calls −mno−long−calls −mmad −mno−mad −mfused−madd −mno−fused−madd −nocpp −mfix−r4000 −mno−fix−r4000 −mfix−r4400 −mno−fix−r4400 −mfix−vr4120 −mno−fix−vr4120 −mfix−vr4130 −mfix−sb1 −mno−fix−sb1 −mflush−func=***func* **−mno−flush−func −mbranch−likely −mno−branch−likely −mfp−exceptions −mno−fp−exceptions −mvr4130−align −mno−vr4130−align**

MMIX Options **−mlibfuncs −mno−libfuncs −mepsilon −mno−epsilon −mabi=gnu −mabi=mmixware −mzero−extend −mknuthdiv −mtoplevel−symbols −melf −mbranch−predict −mno−branch−predict −mbase−addresses −mno−base−addresses −msingle−exit −mno−single−exit**

MN10300 Options **−mmult−bug −mno−mult−bug −mam33 −mno−am33 −mam33−2 −mno−am33−2 −mreturn−pointer−on−d0 −mno−crt0 −mrelax**

MT Options **−mno−crt0 −mbacc −msim −march=***cpu-type*

PDP−11 Options **−mfpu −msoft−float −mac0 −mno−ac0 −m40 −m45 −m10 −mbcopy**

−mbcopy−builtin −mint32 −mno−int16 −mint16 −mno−int32 −mfloat32 −mno−float64 −mfloat64 −mno−float32 −mabshi −mno−abshi −mbranch−expensive −mbranch−cheap −msplit −mno−split −munix−asm −mdec−asm

PowerPC Options See RS/6000 and PowerPC Options.

RS/6000 and PowerPC Options **−mcpu=***cpu-type* **−mtune=***cpu-type* **−mpower −mno−power −mpower2 −mno−power2 −mpowerpc −mpowerpc64 −mno−powerpc −maltivec −mno−altivec −mpowerpc−gpopt −mno−powerpc−gpopt −mpowerpc−gfxopt −mno−powerpc−gfxopt −mmfcrf −mno−mfcrf −mpopcntb −mno−popcntb −mfprnd −mno−fprnd −mnew−mnemonics −mold−mnemonics −mfull−toc −mminimal−toc −mno−fp−in−toc −mno−sum−in−toc −m64 −m32 −mxl−compat −mno−xl−compat −mpe −malign−power −malign−natural −msoft−float −mhard−float −mmultiple −mno−multiple −mstring −mno−string −mupdate −mno−update −mfused−madd −mno−fused−madd −mbit−align −mno−bit−align −mstrict−align −mno−strict−align −mrelocatable −mno−relocatable −mrelocatable−lib −mno−relocatable−lib −mtoc −mno−toc −mlittle −mlittle−endian −mbig −mbig−endian −mdynamic−no−pic −maltivec −mswdiv −mprioritize−restricted−insns=***priority** **−msched−costly−dep=***dependence_type* **−minsert−sched−nops=***scheme* **−mcall−sysv −mcall−netbsd −maix−struct−return −msvr4−struct−return −mabi=***abi-type* **−msecure−plt −mbss−plt −misel −mno−isel −misel=yes −misel=no −mspe −mno−spe −mspe=yes −mspe=no −mvrsave −mno−vrsave −mfloat−gprs=yes −mfloat−gprs=no −mfloat−gprs=single −mfloat−gprs=double −mprototype −mno−prototype −msim −mmvme −mads −myellowknife −memb −msdata −msdata=***opt** **−mvxworks −mwindiss −G** *num* **−pthread**

S/390 and zSeries Options **−mtune=***cpu-type* **−march=***cpu-type* **−mhard−float −msoft−float −mlong−double−64 −mlong−double−128 −mbackchain −mno−backchain −mpacked−stack −mno−packed−stack −msmall−exec −mno−small−exec −mmvcle −mno−mvcle −m64 −m31 −mdebug −mno−debug −mesa −mzarch −mtpf−trace −mno−tpf−trace −mfused−madd −mno−fused−madd −mwarn−framesize −mwarn−dynamicstack −mstack−size −mstack−guard**

SH Options **−m1 −m2 −m2e −m3 −m3e −m4−nofpu −m4−single−only −m4−single −m4 −m4a−nofpu −m4a−single−only −m4a−single −m4a −m4al −m5−64media −m5−64media−nofpu −m5−32media −m5−32media−nofpu −m5−compact −m5−compact−nofpu −mb −ml −mdalign −mrelax −mbigtable −mfmovd −mhitachi −mrenesas −mno−renesas −mnomacsave −mieee −misize −mpadstruct −mspace −mprefergot −musermode −multcost=***number** **−mdiv=***strategy* **−mdivsi3_libfunc=***name* **−madjust−unroll −mindexed−addressing −mgettrcost=***number* **−mpt−fixed**
−minvalid−symbols

SPARC Options **−mcpu=***cpu-type* **−mtune=***cpu-type* **−mcmodel=***code-model* **−m32 −m64 −mapp−regs −mno−app−regs −mfaster−structs −mno−faster−structs −mfpu −mno−fpu −mhard−float −msoft−float −mhard−quad−float −msoft−quad−float −mimpure−text −mno−impure−text −mlittle−endian −mstack−bias −mno−stack−bias −munaligned−doubles −mno−unaligned−doubles −mv8plus −mno−v8plus −mvis −mno−vis −threads −pthreads −pthread**

System V Options **−Qy −Qn −YP,***paths* **−Ym,***dir*

TMS320C3x/C4x Options **−mcpu=***cpu* **−mbig −msmall −mregparm −mmemparm −mfast−fix −mmpyi −mbk −mti −mdp−isr−reload −mrpts=***count* **−mrptb −mdb −mloop−unsigned −mparallel−insns −mparallel−mpy −mpreserve−float**

V850 Options **−mlong−calls −mno−long−calls −mep −mno−ep −mprolog−function −mno−prolog−function −mspace −mtda=***n* **−msda=***n* **−mzda=***n* **−mapp−regs −mno−app−regs −mdisable−callt −mno−disable−callt −mv850e1 −mv850e −mv850 −mbig−switch**

VAX Options **−mg −mgnu −munix**

x86−64 Options See i386 and x86−64 Options.

Xstormy16 Options **−msim**

Xtensa Options **−mconst16 −mno−const16 −mfused−madd −mno−fused−madd −mtext−section−literals −mno−text−section−literals −mtarget−align −mno−target−align −mlongcalls −mno−longcalls**

zSeries Options See S/390 and zSeries Options.

Code Generation Options

−fcall−saved−*reg* **−fcall−used−***reg* **−ffixed−***reg* **−fexceptions −fnon−call−exceptions −funwind−tables −fasynchronous−unwind−tables −finhibit−size−directive −finstrument−functions −fno−common −fno−ident −fpcc−struct−return −fpic −fPIC −fpie −fPIE −fno−jump−tables −freg−struct−return −fshared−data −fshort−enums −fshort−double −fshort−wchar −fverbose−asm −fpack−struct[=***n***] −fstack−check −fstack−limit−register=***reg* **−fstack−limit−symbol=***sym* **−fargument−alias −fargument−noalias −fargument−noalias−global −fleading−underscore −ftls−model=***model* **−ftrapv −fwrapv −fbounds−check −fvisibility**

Options Controlling the Kind of Output

Compilation can involve up to four stages: preprocessing, compilation proper, assembly and linking, always in that order. GCC is capable of preprocessing and compiling several files either into several assembler input files, or into one assembler input file; then each assembler input file produces an object file, and linking combines all the object files (those newly compiled, and those specified as input) into an executable file.

For any given input file, the file name suffix determines what kind of compilation is done:

*file***.c**
C source code which must be preprocessed.

*file***.i**
C source code which should not be preprocessed.

*file***.ii**
C++ source code which should not be preprocessed.

*file***.m**
Objective-C source code. Note that you must link with the *libobjc* library to make an Objective-C program work.

*file***.mi**
Objective-C source code which should not be preprocessed.

*file***.mm**
*file***.M**
Objective−C++ source code. Note that you must link with the *libobjc* library to make an Objective−C++ program work. Note that **.M** refers to a literal capital M.

*file***.mii**
Objective−C++ source code which should not be preprocessed.

*file***.h**
C, C++, Objective-C or Objective−C++ header file to be turned into a precompiled header.

*file***.cc**
*file***.cp**
*file***.cxx**
*file***.cpp**
*file***.CPP**
*file***.c++**
*file***.C**
C++ source code which must be preprocessed. Note that in **.cxx**, the last two letters must both be literally **x**. Likewise, **.C** refers to a literal capital C.

file.**mm**
file.**M**
Objective−C++ source code which must be preprocessed.

file.**mii**
Objective−C++ source code which should not be preprocessed.

file.**hh**
file.**H**
C++ header file to be turned into a precompiled header.

file.**f**
file.**for**
file.**FOR**
Fixed form Fortran source code which should not be preprocessed.

file.**F**
file.**fpp**
file.**FPP**
Fixed form Fortran source code which must be preprocessed (with the traditional preprocessor).

file.**f90**
file.**f95**
Free form Fortran source code which should not be preprocessed.

file.**F90**
file.**F95**
Free form Fortran source code which must be preprocessed (with the traditional preprocessor).

file.**ads**
Ada source code file which contains a library unit declaration (a declaration of a package, subprogram, or generic, or a generic instantiation), or a library unit renaming declaration (a package, generic, or subprogram renaming declaration). Such files are also called *specs*.

file.**adb**
Ada source code file containing a library unit body (a subprogram or package body). Such files are also called *bodies*.

file.**s**
Assembler code.

file.**S**
Assembler code which must be preprocessed.

other
An object file to be fed straight into linking. Any file name with no recognized suffix is treated this way.

You can specify the input language explicitly with the **−x** option:

−x *language*
Specify explicitly the *language* for the following input files (rather than letting the compiler choose a default based on the file name suffix). This option applies to all following input files until the next **−x** option. Possible values for *language* are:

```
c  c-header  c-cpp-output
c++  c++-header  c++-cpp-output
objective-c  objective-c-header  objective-c-cpp-output
objective-c++ objective-c++-header objective-c++-cpp-output
assembler  assembler-with-cpp
ada
f77  f77-cpp-input
f95  f95-cpp-input
java
treelang
```

−x none
Turn off any specification of a language, so that subsequent files are handled according to their file name suffixes (as they are if **−x** has not been used at all).

−pass−exit−codes
Normally the **gcc** program will exit with the code of 1 if any phase of the compiler returns a non-success return code. If you specify **−pass−exit−codes**, the **gcc** program will instead return with numerically highest error produced by any phase that returned an error indication.

If you only want some of the stages of compilation, you can use **−x** (or filename suffixes) to tell **gcc** where to start, and one of the options **−c**, **−S**, or **−E** to say where **gcc** is to stop. Note that some combinations (for example, **−x cpp-output −E**) instruct **gcc** to do nothing at all.

−c Compile or assemble the source files, but do not link. The linking stage simply is not done. The ultimate output is in the form of an object file for each source file.

By default, the object file name for a source file is made by replacing the suffix **.c**, **.i**, **.s**, etc., with **.o**.

Unrecognized input files, not requiring compilation or assembly, are ignored.

−S Stop after the stage of compilation proper; do not assemble. The output is in the form of an assembler code file for each non-assembler input file specified.

By default, the assembler file name for a source file is made by replacing the suffix **.c**, **.i**, etc., with **.s**.

Input files that don't require compilation are ignored.

−E Stop after the preprocessing stage; do not run the compiler proper. The output is in the form of preprocessed source code, which is sent to the standard output.

Input files which don't require preprocessing are ignored.

−o *file*
Place output in file *file*. This applies regardless to whatever sort of output is being produced, whether it be an executable file, an object file, an assembler file or preprocessed C code.

If **−o** is not specified, the default is to put an executable file in *a.out*, the object file for *source.suffix* in *source.o*, its assembler file in *source.s*, a precompiled header file in *source.suffix.gch*, and all preprocessed C source on standard output.

−v Print (on standard error output) the commands executed to run the stages of compilation. Also print the version number of the compiler driver program and of the preprocessor and the compiler proper.

−###
Like **−v** except the commands are not executed and all command arguments are quoted. This is useful for shell scripts to capture the driver-generated command lines.

−pipe
Use pipes rather than temporary files for communication between the various stages of compilation. This fails to work on some systems where the assembler is unable to read from a pipe; but the GNU assembler has no trouble.

−combine
If you are compiling multiple source files, this option tells the driver to pass all the source files to the compiler at once (for those languages for which the compiler can handle this). This will allow inter-module analysis (IMA) to be performed by the compiler. Currently the only language for which this is supported is C. If you pass source files for multiple languages to the driver, using this option, the driver will invoke the compiler(s) that support IMA once each, passing each compiler all the source files appropriate for it. For those languages that do not support IMA this option will be ignored, and the compiler will be invoked once for each source file in that language. If you use this option in conjunction with **−save−temps**, the compiler will generate multiple pre-processed files (one for each source file), but only one (combined) *.o* or *.s* file.

−−help
Print (on the standard output) a description of the command line options understood by **gcc**. If the **−v** option is also specified then **−−help** will also be passed on to the various processes invoked by **gcc**, so that they can display the command line options they accept. If the **−Wextra** option is also specified then command line options which have no documentation associated with them will also be displayed.

−−target−help
Print (on the standard output) a description of target specific command line options for each tool.

−−version
Display the version number and copyrights of the invoked GCC.

Compiling C++ Programs

C++ source files conventionally use one of the suffixes **.C**, **.cc**, **.cpp**, **.CPP**, **.c++**, **.cp**, or **.cxx**; C++ header files often use **.hh** or **.H**; and preprocessed C++ files use the suffix **.ii**. GCC recognizes files with these names and compiles them as C++ programs even if you call the compiler the same way as for compiling C programs (usually with the name **gcc**).

However, C++ programs often require class libraries as well as a compiler that understands the C++ language−−−and under some circumstances, you might want to compile programs or header files from standard input, or otherwise without a suffix that flags them as C++ programs. You might also like to precompile a C header file with a **.h** extension to be used in C++ compilations. **g++** is a program that calls GCC with the default language set to C++, and automatically specifies linking against the C++ library. On many systems, **g++** is also installed with the name **c++**.

When you compile C++ programs, you may specify many of the same command-line options that you use for compiling programs in any language; or command-line options meaningful for C and related languages; or options that are meaningful only for C++ programs.

Options Controlling C Dialect

The following options control the dialect of C (or languages derived from C, such as C++, Objective-C and Objective−C++) that the compiler accepts:

−ansi
In C mode, support all ISO C90 programs. In C++ mode, remove GNU extensions that conflict with ISO C++.

This turns off certain features of GCC that are incompatible with ISO C90 (when compiling C code), or of standard C++ (when compiling C++ code), such as the `asm` and `typeof` keywords, and predefined macros such as `unix` and `vax` that identify the type of system you are using. It also enables the undesirable and rarely used ISO trigraph feature. For the C compiler, it disables recognition of C++ style // comments as well as the `inline` keyword.

The alternate keywords `__asm__`, `__extension__`, `__inline__` and `__typeof__` continue to work despite **−ansi**. You would not want to use them in an ISO C program, of course, but it is useful to put them in header files that might be included in compilations done with **−ansi**. Alternate predefined macros such as `__unix__` and `__vax__` are also available, with or without **−ansi**.

The **−ansi** option does not cause non-ISO programs to be rejected gratuitously. For that, **−pedantic** is required in addition to **−ansi**.

The macro `__STRICT_ANSI__` is predefined when the **−ansi** option is used. Some header files may notice this macro and refrain from declaring certain functions or defining certain macros that the ISO standard doesn't call for; this is to avoid interfering with any programs that might use these names for other things.

Functions which would normally be built in but do not have semantics defined by ISO C (such as `alloca` and `ffs`) are not built-in functions with **−ansi** is used.

−std=

Determine the language standard. This option is currently only supported when compiling C or C++. A value for this option must be provided; possible values are

c89
iso9899:1990
ISO C90 (same as **−ansi**).

iso9899:199409
ISO C90 as modified in amendment 1.

c99
c9x
iso9899:1999
iso9899:199x
ISO C99. Note that this standard is not yet fully supported; see <**http://gcc.gnu.org/gcc−4.1/c99status.html**> for more information. The names **c9x** and **iso9899:199x** are deprecated.

gnu89
Default, ISO C90 plus GNU extensions (including some C99 features).

gnu99
gnu9x
ISO C99 plus GNU extensions. When ISO C99 is fully implemented in GCC, this will become the default. The name **gnu9x** is deprecated.

c++98
The 1998 ISO C++ standard plus amendments.

gnu++98
The same as **−std=c++98** plus GNU extensions. This is the default for C++ code.

Even when this option is not specified, you can still use some of the features of newer standards in so far as they do not conflict with previous C standards. For example, you may use `__restrict__` even when **−std=c99** is not specified.

The **−std** options specifying some version of ISO C have the same effects as **−ansi**, except that features that were not in ISO C90 but are in the specified version (for example, // comments and the `inline` keyword in ISO C99) are not disabled.

−aux−info *filename*

Output to the given filename prototyped declarations for all functions declared and/or defined in a translation unit, including those in header files. This option is silently ignored in any language other than C.

Besides declarations, the file indicates, in comments, the origin of each declaration (source file and line), whether the declaration was implicit, prototyped or unprototyped (**I**, **N** for new or **O** for old, respectively, in the first character after the line number and the colon), and whether it came from a declaration or a definition (**C** or **F**, respectively, in the following character). In the case of function definitions, a K&R−style list of arguments followed by their declarations is also provided, inside

comments, after the declaration.

−fno−asm
Do not recognize `asm`, `inline` or `typeof` as a keyword, so that code can use these words as identifiers. You can use the keywords `__asm__`, `__inline__` and `__typeof__` instead. **−ansi** implies **−fno−asm**.

In C++, this switch only affects the `typeof` keyword, since `asm` and `inline` are standard keywords. You may want to use the **−fno−gnu−keywords** flag instead, which has the same effect. In C99 mode (**−std=c99** or **−std=gnu99**), this switch only affects the `asm` and `typeof` keywords, since `inline` is a standard keyword in ISO C99.

−fno−builtin
−fno−builtin−*function*
Don't recognize built-in functions that do not begin with **__builtin_** as prefix.

GCC normally generates special code to handle certain built-in functions more efficiently; for instance, calls to `alloca` may become single instructions that adjust the stack directly, and calls to `memcpy` may become inline copy loops. The resulting code is often both smaller and faster, but since the function calls no longer appear as such, you cannot set a breakpoint on those calls, nor can you change the behavior of the functions by linking with a different library. In addition, when a function is recognized as a built-in function, GCC may use information about that function to warn about problems with calls to that function, or to generate more efficient code, even if the resulting code still contains calls to that function. For example, warnings are given with **−Wformat** for bad calls to `printf`, when `printf` is built in, and `strlen` is known not to modify global memory.

With the **−fno−builtin−***function* option only the built-in function *function* is disabled. *function* must not begin with **__builtin_**. If a function is named this is not built-in in this version of GCC, this option is ignored. There is no corresponding **−fbuiltin−***function* option; if you wish to enable built-in functions selectively when using **−fno−builtin** or **−ffreestanding**, you may define macros such as:

```
#define abs(n)          __builtin_abs ((n))
#define strcpy(d, s)    __builtin_strcpy ((d), (s))
```

−fhosted
Assert that compilation takes place in a hosted environment. This implies **−fbuiltin**. A hosted environment is one in which the entire standard library is available, and in which `main` has a return type of `int`. Examples are nearly everything except a kernel. This is equivalent to **−fno−freestanding**.

−ffreestanding
Assert that compilation takes place in a freestanding environment. This implies **−fno−builtin**. A freestanding environment is one in which the standard library may not exist, and program startup may not necessarily be at `main`. The most obvious example is an OS kernel. This is equivalent to **−fno−hosted**.

−fms−extensions
Accept some non-standard constructs used in Microsoft header files.

Some cases of unnamed fields in structures and unions are only accepted with this option.

−trigraphs
Support ISO C trigraphs. The **−ansi** option (and **−std** options for strict ISO C conformance) implies **−trigraphs**.

−no−integrated−cpp
Performs a compilation in two passes: preprocessing and compiling. This option allows a user supplied "cc1", "cc1plus", or "cc1obj" via the **−B** option. The user supplied compilation step can then add in an additional preprocessing step after normal preprocessing but before compiling. The default is to use the integrated cpp (internal cpp)

The semantics of this option will change if "cc1", "cc1plus", and "cc1obj" are merged.

−traditional
−traditional−cpp
Formerly, these options caused GCC to attempt to emulate a pre-standard C compiler. They are now only supported with the **−E** switch. The preprocessor continues to support a pre-standard mode. See the GNU CPP manual for details.

−fcond−mismatch
Allow conditional expressions with mismatched types in the second and third arguments. The value of such an expression is void. This option is not supported for C++.

−funsigned−char
Let the type `char` be unsigned, like `unsigned char`.

Each kind of machine has a default for what `char` should be. It is either like `unsigned char` by default or like `signed char` by default.

Ideally, a portable program should always use `signed char` or `unsigned char` when it depends on the signedness of an object. But many programs have been written to use plain `char` and expect it to be signed, or expect it to be unsigned, depending on the machines they were written for. This option, and its inverse, let you make such a program work with the opposite default.

The type `char` is always a distinct type from each of `signed char` or `unsigned char`, even though its behavior is always just like one of those two.

−fsigned−char
Let the type `char` be signed, like `signed char`.

Note that this is equivalent to **−fno−unsigned−char**, which is the negative form of **−funsigned−char**. Likewise, the option **−fno−signed−char** is equivalent to **−funsigned−char**.

−fsigned−bitfields
−funsigned−bitfields
−fno−signed−bitfields
−fno−unsigned−bitfields
These options control whether a bit-field is signed or unsigned, when the declaration does not use either `signed` or `unsigned`. By default, such a bit-field is signed, because this is consistent: the basic integer types such as `int` are signed types.

Options Controlling C++ Dialect

This section describes the command-line options that are only meaningful for C++ programs; but you can also use most of the GNU compiler options regardless of what language your program is in. For example, you might compile a file `firstClass.C` like this:

```
g++ -g -frepo -O -c firstClass.C
```

In this example, only **−frepo** is an option meant only for C++ programs; you can use the other options with any language supported by GCC.

Here is a list of options that are *only* for compiling C++ programs:

−fabi−version=*n*
Use version *n* of the C++ ABI. Version 2 is the version of the C++ ABI that first appeared in G++ 3.4. Version 1 is the version of the C++ ABI that first appeared in G++ 3.2. Version 0 will always be the version that conforms most closely to the C++ ABI specification. Therefore, the ABI obtained using version 0 will change as ABI bugs are fixed.

The default is version 2.

−fno−access−control
Turn off all access checking. This switch is mainly useful for working around bugs in the access control code.

−fcheck−new
Check that the pointer returned by `operator new` is non-null before attempting to modify the storage allocated. This check is normally unnecessary because the C++ standard specifies that `operator new` will only return `0` if it is declared ***throw()***, in which case the compiler will always check the return value even without this option. In all other cases, when `operator new` has a non-empty exception specification, memory exhaustion is signalled by throwing `std::bad_alloc`. See also **new (nothrow)**.

−fconserve−space
Put uninitialized or runtime-initialized global variables into the common segment, as C does. This saves space in the executable at the cost of not diagnosing duplicate definitions. If you compile with this flag and your program mysteriously crashes after `main()` has completed, you may have an object that is being destroyed twice because two definitions were merged.

This option is no longer useful on most targets, now that support has been added for putting variables into BSS without making them common.

−ffriend−injection
Inject friend functions into the enclosing namespace, so that they are visible outside the scope of the class in which they are declared. Friend functions were documented to work this way in the old Annotated C++ Reference Manual, and versions of G++ before 4.1 always worked that way. However, in ISO C++ a friend function which is not declared in an enclosing scope can only be found using argument dependent lookup. This option causes friends to be injected as they were in earlier releases.

This option is for compatibility, and may be removed in a future release of G++.

−fno−const−strings
Give string constants type `char *` instead of type `const char *`. By default, G++ uses type `const char *` as required by the standard. Even if you use **−fno−const−strings**, you cannot actually modify the value of a string constant.

This option might be removed in a future release of G++. For maximum portability, you should structure your code so that it works with string constants that have type `const char *`.

−fno−elide−constructors
The C++ standard allows an implementation to omit creating a temporary which is only used to initialize another object of the same type. Specifying this option disables that optimization, and forces G++ to call the copy constructor in all cases.

−fno−enforce−eh−specs
Don't generate code to check for violation of exception specifications at runtime. This option violates the C++ standard, but may be useful for reducing code size in production builds, much like defining **NDEBUG**. This does not give user code permission to throw exceptions in violation of the exception specifications; the compiler will still optimize based on the specifications, so throwing an unexpected exception will result in undefined behavior.

−ffor−scope
−fno−for−scope
If **−ffor−scope** is specified, the scope of variables declared in a *for-init-statement* is limited to the **for** loop itself, as specified by the C++ standard. If **−fno−for−scope** is specified, the scope of variables declared in a *for-init-statement* extends to the end of the enclosing scope, as was the case in old versions of G++, and other (traditional) implementations of C++.

The default if neither flag is given to follow the standard, but to allow and give a warning for old-style code that would otherwise be invalid, or have different behavior.

−fno−gnu−keywords
Do not recognize `typeof` as a keyword, so that code can use this word as an identifier. You can use the keyword `__typeof__` instead. **−ansi** implies **−fno−gnu−keywords**.

−fno−implicit−templates
Never emit code for non-inline templates which are instantiated implicitly (i.e. by use); only emit code for explicit instantiations.

−fno−implicit−inline−templates
Don't emit code for implicit instantiations of inline templates, either. The default is to handle inlines differently so that compiles with and without optimization will need the same set of explicit instantiations.

−fno−implement−inlines
To save space, do not emit out-of-line copies of inline functions controlled by **#pragma implementation**. This will cause linker errors if these functions are not inlined everywhere they are called.

−fms−extensions
Disable pedantic warnings about constructs used in MFC, such as implicit int and getting a pointer to member function via non-standard syntax.

−fno−nonansi−builtins
Disable built-in declarations of functions that are not mandated by ANSI/ISO C. These include `ffs`, `alloca`, `_exit`, `index`, `bzero`, `conjf`, and other related functions.

−fno−operator−names
Do not treat the operator name keywords `and`, `bitand`, `bitor`, `compl`, `not`, `or` and `xor` as synonyms as keywords.

−fno−optional−diags
Disable diagnostics that the standard says a compiler does not need to issue. Currently, the only such diagnostic issued by G++ is the one for a name having multiple meanings within a class.

−fpermissive
Downgrade some diagnostics about nonconformant code from errors to warnings. Thus, using **−fpermissive** will allow some nonconforming code to compile.

−frepo
Enable automatic template instantiation at link time. This option also implies **−fno−implicit−templates**.

−fno−rtti
Disable generation of information about every class with virtual functions for use by the C++ runtime type identification features (**dynamic_cast** and **typeid**). If you don't use those parts of the language, you can save some space by using this flag. Note that exception handling uses the same information, but it will generate it as needed.

−fstats
Emit statistics about front-end processing at the end of the compilation. This information is generally only useful to the G++ development team.

−ftemplate−depth−*n*
Set the maximum instantiation depth for template classes to *n*. A limit on the template instantiation depth is needed to detect endless recursions during template class instantiation. ANSI/ISO C++ conforming programs must not rely on a maximum depth greater than 17.

−fno−threadsafe−statics
Do not emit the extra code to use the routines specified in the C++ ABI for thread-safe initialization of local statics. You can use this option to reduce code size slightly in code that doesn't need to be thread−safe.

−fuse−cxa−atexit
Register destructors for objects with static storage duration with the `__cxa_atexit` function rather than the `atexit` function. This option is required for fully standards-compliant handling of static destructors, but will only work if your C library supports `__cxa_atexit`.

−fvisibility−inlines−hidden

Causes all inlined methods to be marked with `__attribute__ ((visibility ("hidden")))` so that they do not appear in the export table of a DSO and do not require a PLT indirection when used within the DSO. Enabling this option can have a dramatic effect on load and link times of a DSO as it massively reduces the size of the dynamic export table when the library makes heavy use of templates. While it can cause bloating through duplication of code within each DSO where it is used, often the wastage is less than the considerable space occupied by a long symbol name in the export table which is typical when using templates and namespaces. For even more savings, combine with the **−fvisibility=hidden** switch.

−fno−weak

Do not use weak symbol support, even if it is provided by the linker. By default, G++ will use weak symbols if they are available. This option exists only for testing, and should not be used by end−users; it will result in inferior code and has no benefits. This option may be removed in a future release of G++.

−nostdinc++

Do not search for header files in the standard directories specific to C++, but do still search the other standard directories. (This option is used when building the C++ library.)

In addition, these optimization, warning, and code generation options have meanings only for C++ programs:

−fno−default−inline

Do not assume **inline** for functions defined inside a class scope.

Note that these functions will have linkage like inline functions; they just won't be inlined by default.

−Wabi (C++ only)

Warn when G++ generates code that is probably not compatible with the vendor-neutral C++ ABI. Although an effort has been made to warn about all such cases, there are probably some cases that are not warned about, even though G++ is generating incompatible code. There may also be cases where warnings are emitted even though the code that is generated will be compatible.

You should rewrite your code to avoid these warnings if you are concerned about the fact that code generated by G++ may not be binary compatible with code generated by other compilers.

The known incompatibilities at this point include:

* Incorrect handling of tail-padding for bit−fields. G++ may attempt to pack data into the same byte as a base class. For example:

```
struct A { virtual void f(); int f1 : 1; };
struct B : public A { int f2 : 1; };
```

In this case, G++ will place `B::f2` into the same byte as`A::f1`; other compilers will not. You can avoid this problem by explicitly padding `A` so that its size is a multiple of the byte size on your platform; that will cause G++ and other compilers to layout `B` identically.

* Incorrect handling of tail-padding for virtual bases. G++ does not use tail padding when laying out virtual bases. For example:

```
struct A { virtual void f(); char c1; };
struct B { B(); char c2; };
struct C : public A, public virtual B {};
```

In this case, G++ will not place `B` into the tail-padding for `A`; other compilers will. You can avoid this problem by explicitly padding `A` so that its size is a multiple of its alignment (ignoring virtual base classes); that will cause G++ and other compilers to layout `C` identically.

* Incorrect handling of bit-fields with declared widths greater than that of their underlying types, when the bit-fields appear in a union. For example:

```
union U { int i : 4096; };
```

Assuming that an `int` does not have 4096 bits, G++ will make the union too small by the number of bits in an `int`.

* Empty classes can be placed at incorrect offsets. For example:

```
struct A {};

struct B {
  A a;
  virtual void f ();
};

struct C : public B, public A {};
```

G++ will place the `A` base class of `C` at a nonzero offset; it should be placed at offset zero. G++ mistakenly believes that the `A` data member of `B` is already at offset zero.

* Names of template functions whose types involve `typename` or template template parameters can be mangled incorrectly.

```
template <typename Q>
void f(typename Q::X) {}

template <template <typename> class Q>
void f(typename Q<int>::X) {}
```

Instantiations of these templates may be mangled incorrectly.

−Wctor−dtor−privacy (C++ only)
Warn when a class seems unusable because all the constructors or destructors in that class are private, and it has neither friends nor public static member functions.

−Wnon−virtual−dtor (C++ only)
Warn when a class appears to be polymorphic, thereby requiring a virtual destructor, yet it declares a non-virtual one. This warning is enabled by **−Wall**.

−Wreorder (C++ only)
Warn when the order of member initializers given in the code does not match the order in which they must be executed. For instance:

```
struct A {
  int i;
  int j;
  A(): j (0), i (1) { }
};
```

The compiler will rearrange the member initializers for **i** and **j** to match the declaration order of the members, emitting a warning to that effect. This warning is enabled by **−Wall**.

The following **−W...** options are not affected by **−Wall**.

−Weffc++ (C++ only)
Warn about violations of the following style guidelines from Scott Meyers' *Effective C++* book:

* Item 11: Define a copy constructor and an assignment operator for classes with dynamically allocated memory.
* Item 12: Prefer initialization to assignment in constructors.
* Item 14: Make destructors virtual in base classes.
* Item 15: Have `operator=` return a reference to `*this`.
* Item 23: Don't try to return a reference when you must return an object.

Also warn about violations of the following style guidelines from Scott Meyers' *More Effective C++* book:

* Item 6: Distinguish between prefix and postfix forms of increment and decrement operators.
* Item 7: Never overload `&&`, `||`, or `,`.

When selecting this option, be aware that the standard library headers do not obey all of these guidelines; use **grep -v** to filter out those warnings.

-Wno-deprecated (C++ only)
Do not warn about usage of deprecated features.

-Wstrict-null-sentinel (C++ only)
Warn also about the use of an uncasted `NULL` as sentinel. When compiling only with GCC this is a valid sentinel, as `NULL` is defined to `__null`. Although it is a null pointer constant not a null pointer, it is guaranteed to of the same size as a pointer. But this use is not portable across different compilers.

-Wno-non-template-friend (C++ only)
Disable warnings when non-templatized friend functions are declared within a template. Since the advent of explicit template specification support in G++, if the name of the friend is an unqualified-id (i.e., **friend foo(int)**), the C++ language specification demands that the friend declare or define an ordinary, nontemplate function. (Section 14.5.3). Before G++ implemented explicit specification, unqualified-ids could be interpreted as a particular specialization of a templatized function. Because this non-conforming behavior is no longer the default behavior for G++, **-Wnon-template-friend** allows the compiler to check existing code for potential trouble spots and is on by default. This new compiler behavior can be turned off with **-Wno-non-template-friend** which keeps the conformant compiler code but disables the helpful warning.

-Wold-style-cast (C++ only)
Warn if an old-style (C-style) cast to a non-void type is used within a C++ program. The new-style casts (**dynamic_cast**, **static_cast**, **reinterpret_cast**, and **const_cast**) are less vulnerable to unintended effects and much easier to search for.

-Woverloaded-virtual (C++ only)
Warn when a function declaration hides virtual functions from a base class. For example, in:

```
struct A {
  virtual void f();
};

struct B: public A {
  void f(int);
};
```

the `A` class version of `f` is hidden in `B`, and code like:

```
B* b;
b->f();
```

will fail to compile.

-Wno-pmf-conversions (C++ only)
Disable the diagnostic for converting a bound pointer to member function to a plain pointer.

-Wsign-promo (C++ only)
Warn when overload resolution chooses a promotion from unsigned or enumerated type to a signed type, over a conversion to an unsigned type of the same size. Previous versions of G++ would try to preserve unsignedness, but the standard mandates the current behavior.

```
struct A {
  operator int ();
  A& operator = (int);
};
```

```
main ()
{
  A a,b;
  a = b;
}
```

In this example, G++ will synthesize a default **A& operator = (const A&);**, while cfront will use the user-defined **operator =**.

Options Controlling Objective-C and Objective−C++ Dialects

(NOTE: This manual does not describe the Objective-C and Objective−C++ languages themselves. See

This section describes the command-line options that are only meaningful for Objective-C and Objective−C++ programs, but you can also use most of the language-independent GNU compiler options. For example, you might compile a file `some_class.m` like this:

```
gcc -g -fgnu-runtime -O -c some_class.m
```

In this example, **−fgnu−runtime** is an option meant only for Objective-C and Objective−C++ programs; you can use the other options with any language supported by GCC.

Note that since Objective-C is an extension of the C language, Objective-C compilations may also use options specific to the C front-end (e.g., **−Wtraditional**). Similarly, Objective−C++ compilations may use C++−specific options (e.g., **−Wabi**).

Here is a list of options that are *only* for compiling Objective-C and Objective−C++ programs:

−fconstant−string−class=*class-name*
Use *class-name* as the name of the class to instantiate for each literal string specified with the syntax `@"..."`. The default class name is `NXConstantString` if the GNU runtime is being used, and `NSConstantString` if the NeXT runtime is being used (see below). The **−fconstant−cfstrings** option, if also present, will override the **−fconstant−string−class** setting and cause `@"..."` literals to be laid out as constant CoreFoundation strings.

−fgnu−runtime
Generate object code compatible with the standard GNU Objective-C runtime. This is the default for most types of systems.

−fnext−runtime
Generate output compatible with the NeXT runtime. This is the default for NeXT-based systems, including Darwin and Mac OS X. The macro `__NEXT_RUNTIME__` is predefined if (and only if) this option is used.

−fno−nil−receivers
Assume that all Objective-C message dispatches (e.g., `[receiver message:arg]`) in this translation unit ensure that the receiver is not `nil`. This allows for more efficient entry points in the runtime to be used. Currently, this option is only available in conjunction with the NeXT runtime on Mac OS X 10.3 and later.

−fobjc−call−cxx−cdtors
For each Objective-C class, check if any of its instance variables is a C++ object with a non-trivial default constructor. If so, synthesize a special `- (id) .cxx_construct` instance method that will run non-trivial default constructors on any such instance variables, in order, and then return `self`. Similarly, check if any instance variable is a C++ object with a non-trivial destructor, and if so, synthesize a special `- (void) .cxx_destruct` method that will run all such default destructors, in reverse order.

The `- (id) .cxx_construct` and/or `- (void) .cxx_destruct` methods thusly generated will only operate on instance variables declared in the current Objective-C class, and not those inherited from superclasses. It is the responsibility of the Objective-C runtime to invoke all such methods in an object's inheritance hierarchy. The `- (id) .cxx_construct` methods will be invoked by

the runtime immediately after a new object instance is allocated; the `- (void) .cxx_destruct` methods will be invoked immediately before the runtime deallocates an object instance.

As of this writing, only the NeXT runtime on Mac OS X 10.4 and later has support for invoking the `- (id) .cxx_construct` and `- (void) .cxx_destruct` methods.

-fobjc-direct-dispatch
Allow fast jumps to the message dispatcher. On Darwin this is accomplished via the comm page.

-fobjc-exceptions
Enable syntactic support for structured exception handling in Objective-C, similar to what is offered by C++ and Java. Currently, this option is only available in conjunction with the NeXT runtime on Mac OS X 10.3 and later.

```
@try {
  ...
     @throw expr;
  ...
}
@catch (AnObjCClass *exc) {
  ...
    @throw expr;
  ...
    @throw;
  ...
}
@catch (AnotherClass *exc) {
  ...
}
@catch (id allOthers) {
  ...
}
@finally {
  ...
    @throw expr;
  ...
}
```

The `@throw` statement may appear anywhere in an Objective-C or Objective-C++ program; when used inside of a `@catch` block, the `@throw` may appear without an argument (as shown above), in which case the object caught by the `@catch` will be rethrown.

Note that only (pointers to) Objective-C objects may be thrown and caught using this scheme. When an object is thrown, it will be caught by the nearest `@catch` clause capable of handling objects of that type, analogously to how `catch` blocks work in C++ and Java. A `@catch(id ...)` clause (as shown above) may also be provided to catch any and all Objective-C exceptions not caught by previous `@catch` clauses (if any).

The `@finally` clause, if present, will be executed upon exit from the immediately preceding `@try ... @catch` section. This will happen regardless of whether any exceptions are thrown, caught or rethrown inside the `@try ... @catch` section, analogously to the behavior of the `finally` clause in Java.

There are several caveats to using the new exception mechanism:

* Although currently designed to be binary compatible with `NS_HANDLER`-style idioms provided by the `NSException` class, the new exceptions can only be used on Mac OS X 10.3 (Panther) and later systems, due to additional functionality needed in the (NeXT) Objective-C runtime.

* As mentioned above, the new exceptions do not support handling types other than Objective-C objects. Furthermore, when used from Objective−C++, the Objective-C exception model does not interoperate with C++ exceptions at this time. This means you cannot `@throw` an exception from Objective-C and `catch` it in C++, or vice versa (i.e., `throw ... @catch`).

The **−fobjc−exceptions** switch also enables the use of synchronization blocks for thread-safe execution:

```
@synchronized (ObjCClass *guard) {
  ...
}
```

Upon entering the `@synchronized` block, a thread of execution shall first check whether a lock has been placed on the corresponding `guard` object by another thread. If it has, the current thread shall wait until the other thread relinquishes its lock. Once `guard` becomes available, the current thread will place its own lock on it, execute the code contained in the `@synchronized` block, and finally relinquish the lock (thereby making `guard` available to other threads).

Unlike Java, Objective-C does not allow for entire methods to be marked `@synchronized`. Note that throwing exceptions out of `@synchronized` blocks is allowed, and will cause the guarding object to be unlocked properly.

−fobjc−gc
Enable garbage collection (GC) in Objective-C and Objective−C++ programs.

−freplace−objc−classes
Emit a special marker instructing ***ld*** **(1)** not to statically link in the resulting object file, and allow ***dyld*** **(1)** to load it in at run time instead. This is used in conjunction with the Fix-and-Continue debugging mode, where the object file in question may be recompiled and dynamically reloaded in the course of program execution, without the need to restart the program itself. Currently, Fix-and-Continue functionality is only available in conjunction with the NeXT runtime on Mac OS X 10.3 and later.

−fzero−link
When compiling for the NeXT runtime, the compiler ordinarily replaces calls to `objc_getClass("...")` (when the name of the class is known at compile time) with static class references that get initialized at load time, which improves run-time performance. Specifying the **−fzero−link** flag suppresses this behavior and causes calls to `objc_getClass("...")` to be retained. This is useful in Zero-Link debugging mode, since it allows for individual class implementations to be modified during program execution.

−gen−decls
Dump interface declarations for all classes seen in the source file to a file named *sourcename.decl*.

−Wassign−intercept
Warn whenever an Objective-C assignment is being intercepted by the garbage collector.

−Wno−protocol
If a class is declared to implement a protocol, a warning is issued for every method in the protocol that is not implemented by the class. The default behavior is to issue a warning for every method not explicitly implemented in the class, even if a method implementation is inherited from the superclass. If you use the **−Wno−protocol** option, then methods inherited from the superclass are considered to be implemented, and no warning is issued for them.

−Wselector
Warn if multiple methods of different types for the same selector are found during compilation. The check is performed on the list of methods in the final stage of compilation. Additionally, a check is performed for each selector appearing in a `@selector(...)` expression, and a corresponding method for that selector has been found during compilation. Because these checks scan the method table only at the end of compilation, these warnings are not produced if the final stage of compilation is not reached, for example because an error is found during compilation, or because the **−fsyntax−only**

option is being used.

−Wstrict−selector−match
Warn if multiple methods with differing argument and/or return types are found for a given selector when attempting to send a message using this selector to a receiver of type `id` or `Class`. When this flag is off (which is the default behavior), the compiler will omit such warnings if any differences found are confined to types which share the same size and alignment.

−Wundeclared−selector
Warn if a `@selector(...)` expression referring to an undeclared selector is found. A selector is considered undeclared if no method with that name has been declared before the `@selector(...)` expression, either explicitly in an `@interface` or `@protocol` declaration, or implicitly in an `@implementation` section. This option always performs its checks as soon as a `@selector(...)` expression is found, while **−Wselector** only performs its checks in the final stage of compilation. This also enforces the coding style convention that methods and selectors must be declared before being used.

−print−objc−runtime−info
Generate C header describing the largest structure that is passed by value, if any.

Options to Control Diagnostic Messages Formatting

Traditionally, diagnostic messages have been formatted irrespective of the output device's aspect (e.g. its width, ...). The options described below can be used to control the diagnostic messages formatting algorithm, e.g. how many characters per line, how often source location information should be reported. Right now, only the C++ front end can honor these options. However it is expected, in the near future, that the remaining front ends would be able to digest them correctly.

−fmessage−length=*n*
Try to format error messages so that they fit on lines of about *n* characters. The default is 72 characters for **g++** and 0 for the rest of the front ends supported by GCC. If *n* is zero, then no line-wrapping will be done; each error message will appear on a single line.

−fdiagnostics−show−location=once
Only meaningful in line-wrapping mode. Instructs the diagnostic messages reporter to emit *once* source location information; that is, in case the message is too long to fit on a single physical line and has to be wrapped, the source location won't be emitted (as prefix) again, over and over, in subsequent continuation lines. This is the default behavior.

−fdiagnostics−show−location=every−line
Only meaningful in line-wrapping mode. Instructs the diagnostic messages reporter to emit the same source location information (as prefix) for physical lines that result from the process of breaking a message which is too long to fit on a single line.

−fdiagnostics−show−options
This option instructs the diagnostic machinery to add text to each diagnostic emitted, which indicates which command line option directly controls that diagnostic, when such an option is known to the diagnostic machinery.

Options to Request or Suppress Warnings

Warnings are diagnostic messages that report constructions which are not inherently erroneous but which are risky or suggest there may have been an error.

You can request many specific warnings with options beginning **−W**, for example **−Wimplicit** to request warnings on implicit declarations. Each of these specific warning options also has a negative form beginning **−Wno−** to turn off warnings; for example, **−Wno−implicit**. This manual lists only one of the two forms, whichever is not the default.

The following options control the amount and kinds of warnings produced by GCC; for further, language-specific options also refer to **C++ Dialect Options** and **Objective-C and Objective−C++ Dialect Options**.

−fsyntax−only
Check the code for syntax errors, but don't do anything beyond that.

−pedantic
Issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any **−std** option used.

Valid ISO C and ISO C++ programs should compile properly with or without this option (though a rare few will require **−ansi** or a **−std** option specifying the required version of ISO C). However, without this option, certain GNU extensions and traditional C and C++ features are supported as well. With this option, they are rejected.

−pedantic does not cause warning messages for use of the alternate keywords whose names begin and end with `__`. Pedantic warnings are also disabled in the expression that follows `__extension__`. However, only system header files should use these escape routes; application programs should avoid them.

Some users try to use **−pedantic** to check programs for strict ISO C conformance. They soon find that it does not do quite what they want: it finds some non-ISO practices, but not all−−−only those for which ISO C *requires* a diagnostic, and some others for which diagnostics have been added.

A feature to report any failure to conform to ISO C might be useful in some instances, but would require considerable additional work and would be quite different from **−pedantic**. We don't have plans to support such a feature in the near future.

Where the standard specified with **−std** represents a GNU extended dialect of C, such as **gnu89** or **gnu99**, there is a corresponding *base standard*, the version of ISO C on which the GNU extended dialect is based. Warnings from **−pedantic** are given where they are required by the base standard. (It would not make sense for such warnings to be given only for features not in the specified GNU C dialect, since by definition the GNU dialects of C include all features the compiler supports with the given option, and there would be nothing to warn about.)

−pedantic−errors
Like **−pedantic**, except that errors are produced rather than warnings.

−w Inhibit all warning messages.

−Wno−import
Inhibit warning messages about the use of **#import**.

−Wchar−subscripts
Warn if an array subscript has type `char`. This is a common cause of error, as programmers often forget that this type is signed on some machines. This warning is enabled by **−Wall**.

−Wcomment
Warn whenever a comment-start sequence **/*** appears in a **/*** comment, or whenever a Backslash-Newline appears in a // comment. This warning is enabled by **−Wall**.

−Wfatal−errors
This option causes the compiler to abort compilation on the first error occurred rather than trying to keep going and printing further error messages.

−Wformat
Check calls to `printf` and `scanf`, etc., to make sure that the arguments supplied have types appropriate to the format string specified, and that the conversions specified in the format string make sense. This includes standard functions, and others specified by format attributes, in the `printf`, `scanf`, `strftime` and `strfmon` (an X/Open extension, not in the C standard) families (or other target-specific families). Which functions are checked without format attributes having been specified depends on the standard version selected, and such checks of functions without the attribute specified are disabled by **−ffreestanding** or **−fno−builtin**.

The formats are checked against the format features supported by GNU libc version 2.2. These include all ISO C90 and C99 features, as well as features from the Single Unix Specification and some BSD and GNU extensions. Other library implementations may not support all these features; GCC does not support warning about features that go beyond a particular library's limitations. However, if **-pedantic** is used with **-Wformat**, warnings will be given about format features not in the selected standard version (but not for `strfmon` formats, since those are not in any version of the C standard).

Since **-Wformat** also checks for null format arguments for several functions, **-Wformat** also implies **-Wnonnull**.

-Wformat is included in **-Wall**. For more control over some aspects of format checking, the options **-Wformat-y2k**, **-Wno-format-extra-args**, **-Wno-format-zero-length**, **-Wformat-nonliteral**, **-Wformat-security**, and **-Wformat=2** are available, but are not included in **-Wall**.

-Wformat-y2k
If **-Wformat** is specified, also warn about `strftime` formats which may yield only a two-digit year.

-Wno-format-extra-args
If **-Wformat** is specified, do not warn about excess arguments to a `printf` or `scanf` format function. The C standard specifies that such arguments are ignored.

Where the unused arguments lie between used arguments that are specified with **$** operand number specifications, normally warnings are still given, since the implementation could not know what type to pass to `va_arg` to skip the unused arguments. However, in the case of `scanf` formats, this option will suppress the warning if the unused arguments are all pointers, since the Single Unix Specification says that such unused arguments are allowed.

-Wno-format-zero-length
If **-Wformat** is specified, do not warn about zero-length formats. The C standard specifies that zero-length formats are allowed.

-Wformat-nonliteral
If **-Wformat** is specified, also warn if the format string is not a string literal and so cannot be checked, unless the format function takes its format arguments as a `va_list`.

-Wformat-security
If **-Wformat** is specified, also warn about uses of format functions that represent possible security problems. At present, this warns about calls to `printf` and `scanf` functions where the format string is not a string literal and there are no format arguments, as in `printf (foo);`. This may be a security hole if the format string came from untrusted input and contains **%n**. (This is currently a subset of what **-Wformat-nonliteral** warns about, but in future warnings may be added to **-Wformat-security** that are not included in **-Wformat-nonliteral**.)

-Wformat=2
Enable **-Wformat** plus format checks not included in **-Wformat**. Currently equivalent to **-Wformat -Wformat-nonliteral -Wformat-security -Wformat-y2k**.

-Wnonnull
Warn about passing a null pointer for arguments marked as requiring a non-null value by the `nonnull` function attribute.

-Wnonnull is included in **-Wall** and **-Wformat**. It can be disabled with the **-Wno-nonnull** option.

-Winit-self (C, C++, Objective-C and Objective-C++ only)
Warn about uninitialized variables which are initialized with themselves. Note this option can only be used with the **-Wuninitialized** option, which in turn only works with **-O1** and above.

For example, GCC will warn about `i` being uninitialized in the following snippet only when **-Winit-self** has been specified:

```
int f()
{
  int i = i;
  return i;
}
```

−Wimplicit−int
Warn when a declaration does not specify a type. This warning is enabled by **−Wall**.

−Wimplicit−function−declaration
−Werror−implicit−function−declaration
Give a warning (or error) whenever a function is used before being declared. The form **−Wno−error−implicit−function−declaration** is not supported. This warning is enabled by **−Wall** (as a warning, not an error).

−Wimplicit
Same as **−Wimplicit−int** and **−Wimplicit−function−declaration**. This warning is enabled by **−Wall**.

−Wmain
Warn if the type of **main** is suspicious. **main** should be a function with external linkage, returning int, taking either zero arguments, two, or three arguments of appropriate types. This warning is enabled by **−Wall**.

−Wmissing−braces
Warn if an aggregate or union initializer is not fully bracketed. In the following example, the initializer for **a** is not fully bracketed, but that for **b** is fully bracketed.

```
int a[2][2] = { 0, 1, 2, 3 };
int b[2][2] = { { 0, 1 }, { 2, 3 } };
```

This warning is enabled by **−Wall**.

−Wmissing−include−dirs (C, C++, Objective-C and Objective−C++ only)
Warn if a user-supplied include directory does not exist.

−Wparentheses
Warn if parentheses are omitted in certain contexts, such as when there is an assignment in a context where a truth value is expected, or when operators are nested whose precedence people often get confused about. Only the warning for an assignment used as a truth value is supported when compiling C++; the other warnings are only supported when compiling C.

Also warn if a comparison like **x<=y<=z** appears; this is equivalent to **(x<=y ? 1 : 0) <= z**, which is a different interpretation from that of ordinary mathematical notation.

Also warn about constructions where there may be confusion to which `if` statement an `else` branch belongs. Here is an example of such a case:

```
{
  if (a)
    if (b)
      foo ();
  else
    bar ();
}
```

In C, every `else` branch belongs to the innermost possible `if` statement, which in this example is `if (b)`. This is often not what the programmer expected, as illustrated in the above example by indentation the programmer chose. When there is the potential for this confusion, GCC will issue a warning when this flag is specified. To eliminate the warning, add explicit braces around the innermost `if` statement so there is no way the `else` could belong to the enclosing `if`. The resulting code would look like this:

```
{
  if (a)
    {
      if (b)
        foo ();
      else
        bar ();
    }
}
```

This warning is enabled by **-Wall**.

-Wsequence-point

Warn about code that may have undefined semantics because of violations of sequence point rules in the C standard.

The C standard defines the order in which expressions in a C program are evaluated in terms of *sequence points*, which represent a partial ordering between the execution of parts of the program: those executed before the sequence point, and those executed after it. These occur after the evaluation of a full expression (one which is not part of a larger expression), after the evaluation of the first operand of a `&&`, `||`, `? :` or `,` (comma) operator, before a function is called (but after the evaluation of its arguments and the expression denoting the called function), and in certain other places. Other than as expressed by the sequence point rules, the order of evaluation of subexpressions of an expression is not specified. All these rules describe only a partial order rather than a total order, since, for example, if two functions are called within one expression with no sequence point between them, the order in which the functions are called is not specified. However, the standards committee have ruled that function calls do not overlap.

It is not specified when between sequence points modifications to the values of objects take effect. Programs whose behavior depends on this have undefined behavior; the C standard specifies that ‘‘Between the previous and next sequence point an object shall have its stored value modified at most once by the evaluation of an expression. Furthermore, the prior value shall be read only to determine the value to be stored.’’. If a program breaks these rules, the results on any particular implementation are entirely unpredictable.

Examples of code with undefined behavior are `a = a++;`, `a[n] = b[n++]` and `a[i++] = i;`. Some more complicated cases are not diagnosed by this option, and it may give an occasional false positive result, but in general it has been found fairly effective at detecting this sort of problem in programs.

The present implementation of this option only works for C programs. A future implementation may also work for C++ programs.

The C standard is worded confusingly, therefore there is some debate over the precise meaning of the sequence point rules in subtle cases. Links to discussions of the problem, including proposed formal definitions, may be found on the GCC readings page, at <**http://gcc.gnu.org/readings.html**>.

This warning is enabled by **-Wall**.

-Wreturn-type

Warn whenever a function is defined with a return-type that defaults to `int`. Also warn about any `return` statement with no return-value in a function whose return-type is not `void`.

For C, also warn if the return type of a function has a type qualifier such as `const`. Such a type qualifier has no effect, since the value returned by a function is not an lvalue. ISO C prohibits qualified `void` return types on function definitions, so such return types always receive a warning even without this option.

For C++, a function without return type always produces a diagnostic message, even when **-Wno-return-type** is specified. The only exceptions are **main** and functions defined in system

headers.

This warning is enabled by **−Wall**.

−Wswitch
Warn whenever a `switch` statement has an index of enumerated type and lacks a `case` for one or more of the named codes of that enumeration. (The presence of a `default` label prevents this warning.) `case` labels outside the enumeration range also provoke warnings when this option is used. This warning is enabled by **−Wall**.

−Wswitch−default
Warn whenever a `switch` statement does not have a `default` case.

−Wswitch−enum
Warn whenever a `switch` statement has an index of enumerated type and lacks a `case` for one or more of the named codes of that enumeration. `case` labels outside the enumeration range also provoke warnings when this option is used.

−Wtrigraphs
Warn if any trigraphs are encountered that might change the meaning of the program (trigraphs within comments are not warned about). This warning is enabled by **−Wall**.

−Wunused−function
Warn whenever a static function is declared but not defined or a non-inline static function is unused. This warning is enabled by **−Wall**.

−Wunused−label
Warn whenever a label is declared but not used. This warning is enabled by **−Wall**.

To suppress this warning use the **unused** attribute.

−Wunused−parameter
Warn whenever a function parameter is unused aside from its declaration.

To suppress this warning use the **unused** attribute.

−Wunused−variable
Warn whenever a local variable or non-constant static variable is unused aside from its declaration This warning is enabled by **−Wall**.

To suppress this warning use the **unused** attribute.

−Wunused−value
Warn whenever a statement computes a result that is explicitly not used. This warning is enabled by **−Wall**.

To suppress this warning cast the expression to **void**.

−Wunused
All the above **−Wunused** options combined.

In order to get a warning about an unused function parameter, you must either specify **−Wextra −Wunused** (note that **−Wall** implies **−Wunused**), or separately specify **−Wunused−parameter**.

−Wuninitialized
Warn if an automatic variable is used without first being initialized or if a variable may be clobbered by a `setjmp` call.

These warnings are possible only in optimizing compilation, because they require data flow information that is computed only when optimizing. If you don't specify **−O**, you simply won't get these warnings.

If you want to warn about code which uses the uninitialized value of the variable in its own initializer, use the **−Winit−self** option.

These warnings occur for individual uninitialized or clobbered elements of structure, union or array

variables as well as for variables which are uninitialized or clobbered as a whole. They do not occur for variables or elements declared `volatile`. Because these warnings depend on optimization, the exact variables or elements for which there are warnings will depend on the precise optimization options and version of GCC used.

Note that there may be no warning about a variable that is used only to compute a value that itself is never used, because such computations may be deleted by data flow analysis before the warnings are printed.

These warnings are made optional because GCC is not smart enough to see all the reasons why the code might be correct despite appearing to have an error. Here is one example of how this can happen:

```
{
  int x;
  switch (y)
    {
    case 1: x = 1;
      break;
    case 2: x = 4;
      break;
    case 3: x = 5;
    }
  foo (x);
}
```

If the value of `y` is always 1, 2 or 3, then `x` is always initialized, but GCC doesn't know this. Here is another common case:

```
{
  int save_y;
  if (change_y) save_y = y, y = new_y;
  ...
  if (change_y) y = save_y;
}
```

This has no bug because `save_y` is used only if it is set.

This option also warns when a non-volatile automatic variable might be changed by a call to `longjmp`. These warnings as well are possible only in optimizing compilation.

The compiler sees only the calls to `setjmp`. It cannot know where `longjmp` will be called; in fact, a signal handler could call it at any point in the code. As a result, you may get a warning even when there is in fact no problem because `longjmp` cannot in fact be called at the place which would cause a problem.

Some spurious warnings can be avoided if you declare all the functions you use that never return as `noreturn`.

This warning is enabled by **-Wall**.

-Wunknown-pragmas

Warn when a #pragma directive is encountered which is not understood by GCC. If this command line option is used, warnings will even be issued for unknown pragmas in system header files. This is not the case if the warnings were only enabled by the **-Wall** command line option.

-Wno-pragmas

Do not warn about misuses of pragmas, such as incorrect parameters, invalid syntax, or conflicts between pragmas. See also **-Wunknown-pragmas**.

−Wstrict−aliasing
This option is only active when **−fstrict−aliasing** is active. It warns about code which might break the strict aliasing rules that the compiler is using for optimization. The warning does not catch all cases, but does attempt to catch the more common pitfalls. It is included in **−Wall**.

−Wstrict−aliasing=2
This option is only active when **−fstrict−aliasing** is active. It warns about code which might break the strict aliasing rules that the compiler is using for optimization. This warning catches more cases than **−Wstrict−aliasing**, but it will also give a warning for some ambiguous cases that are safe.

−Wall
All of the above **−W** options combined. This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros. This also enables some language-specific warnings described in **C++ Dialect Options** and **Objective-C and Objective−C++ Dialect Options**.

The following **−W...** options are not implied by **−Wall**. Some of them warn about constructions that users generally do not consider questionable, but which occasionally you might wish to check for; others warn about constructions that are necessary or hard to avoid in some cases, and there is no simple way to modify the code to suppress the warning.

−Wextra
(This option used to be called **−W**. The older name is still supported, but the newer name is more descriptive.) Print extra warning messages for these events:

* A function can return either with or without a value. (Falling off the end of the function body is considered returning without a value.) For example, this function would evoke such a warning:

```
foo (a)
{
  if (a > 0)
    return a;
}
```

* An expression-statement or the left-hand side of a comma expression contains no side effects. To suppress the warning, cast the unused expression to void. For example, an expression such as **x[i,j]** will cause a warning, but **x[(void)i,j]** will not.
* An unsigned value is compared against zero with < or >=.
* Storage-class specifiers like `static` are not the first things in a declaration. According to the C Standard, this usage is obsolescent.
* If **−Wall** or **−Wunused** is also specified, warn about unused arguments.
* A comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned. (But don't warn if **−Wno−sign−compare** is also specified.)
* An aggregate has an initializer which does not initialize all members. This warning can be independently controlled by **−Wmissing−field−initializers**.
* A function parameter is declared without a type specifier in K&R−style functions:

```
void foo(bar) { }
```

* An empty body occurs in an **if** or **else** statement.
* A pointer is compared against integer zero with <, <=, >, or >=.
* A variable might be changed by **longjmp** or **vfork**.
* Any of several floating-point events that often indicate errors, such as overflow, underflow, loss of precision, etc.

*<(C++ only)>
An enumerator and a non-enumerator both appear in a conditional expression.

*<(C++ only)>
A non-static reference or non-static **const** member appears in a class without constructors.

*<(C++ only)>
Ambiguous virtual bases.

*<(C++ only)>
Subscripting an array which has been declared **register**.

*<(C++ only)>
Taking the address of a variable which has been declared **register**.

*<(C++ only)>
A base class is not initialized in a derived class' copy constructor.

−Wno−div−by−zero
Do not warn about compile-time integer division by zero. Floating point division by zero is not warned about, as it can be a legitimate way of obtaining infinities and NaNs.

−Wsystem−headers
Print warning messages for constructs found in system header files. Warnings from system headers are normally suppressed, on the assumption that they usually do not indicate real problems and would only make the compiler output harder to read. Using this command line option tells GCC to emit warnings from system headers as if they occurred in user code. However, note that using **−Wall** in conjunction with this option will *not* warn about unknown pragmas in system headers−−−for that, **−Wunknown−pragmas** must also be used.

−Wfloat−equal
Warn if floating point values are used in equality comparisons.

The idea behind this is that sometimes it is convenient (for the programmer) to consider floating-point values as approximations to infinitely precise real numbers. If you are doing this, then you need to compute (by analyzing the code, or in some other way) the maximum or likely maximum error that the computation introduces, and allow for it when performing comparisons (and when producing output, but that's a different problem). In particular, instead of testing for equality, you would check to see whether the two values have ranges that overlap; and this is done with the relational operators, so equality comparisons are probably mistaken.

−Wtraditional (C only)
Warn about certain constructs that behave differently in traditional and ISO C. Also warn about ISO C constructs that have no traditional C equivalent, and/or problematic constructs which should be avoided.

* Macro parameters that appear within string literals in the macro body. In traditional C macro replacement takes place within string literals, but does not in ISO C.

* In traditional C, some preprocessor directives did not exist. Traditional preprocessors would only consider a line to be a directive if the **#** appeared in column 1 on the line. Therefore **−Wtraditional** warns about directives that traditional C understands but would ignore because the **#** does not appear as the first character on the line. It also suggests you hide directives like **#pragma** not understood by traditional C by indenting them. Some traditional implementations would not recognize **#elif**, so it suggests avoiding it altogether.

* A function-like macro that appears without arguments.

* The unary plus operator.

* The **U** integer constant suffix, or the **F** or **L** floating point constant suffixes. (Traditional C does support the **L** suffix on integer constants.) Note, these suffixes appear in macros defined in the system headers of most modern systems, e.g. the **_MIN**/**_MAX** macros in `<limits.h>`. Use

of these macros in user code might normally lead to spurious warnings, however GCC's integrated preprocessor has enough context to avoid warning in these cases.

* A function declared external in one block and then used after the end of the block.
* A `switch` statement has an operand of type `long`.
* A non-`static` function declaration follows a `static` one. This construct is not accepted by some traditional C compilers.
* The ISO type of an integer constant has a different width or signedness from its traditional type. This warning is only issued if the base of the constant is ten. I.e. hexadecimal or octal values, which typically represent bit patterns, are not warned about.
* Usage of ISO string concatenation is detected.
* Initialization of automatic aggregates.
* Identifier conflicts with labels. Traditional C lacks a separate namespace for labels.
* Initialization of unions. If the initializer is zero, the warning is omitted. This is done under the assumption that the zero initializer in user code appears conditioned on e.g. `__STDC__` to avoid missing initializer warnings and relies on default initialization to zero in the traditional C case.
* Conversions by prototypes between fixed/floating point values and vice versa. The absence of these prototypes when compiling with traditional C would cause serious problems. This is a subset of the possible conversion warnings, for the full set use **-Wconversion**.
* Use of ISO C style function definitions. This warning intentionally is *not* issued for prototype declarations or variadic functions because these ISO C features will appear in your code when using libiberty's traditional C compatibility macros, `PARAMS` and `VPARAMS`. This warning is also bypassed for nested functions because that feature is already a GCC extension and thus not relevant to traditional C compatibility.

-Wdeclaration-after-statement (C only)
Warn when a declaration is found after a statement in a block. This construct, known from C++, was introduced with ISO C99 and is by default allowed in GCC. It is not supported by ISO C90 and was not supported by GCC versions before GCC 3.0.

-Wundef
Warn if an undefined identifier is evaluated in an **#if** directive.

-Wno-endif-labels
Do not warn whenever an **#else** or an **#endif** are followed by text.

-Wshadow
Warn whenever a local variable shadows another local variable, parameter or global variable or whenever a built-in function is shadowed.

-Wlarger-than-*len*
Warn whenever an object of larger than *len* bytes is defined.

-Wunsafe-loop-optimizations
Warn if the loop cannot be optimized because the compiler could not assume anything on the bounds of the loop indices. With **-funsafe-loop-optimizations** warn if the compiler made such assumptions.

-Wpointer-arith
Warn about anything that depends on the ''size of'' a function type or of `void`. GNU C assigns these types a size of 1, for convenience in calculations with `void *` pointers and pointers to functions.

-Wbad-function-cast (C only)
Warn whenever a function call is cast to a non-matching type. For example, warn if `int malloc()` is cast to `anything *`.

−Wc++−compat
Warn about ISO C constructs that are outside of the common subset of ISO C and ISO C++, e.g. request for implicit conversion from `void *` to a pointer to non−`void` type.

−Wcast−qual
Warn whenever a pointer is cast so as to remove a type qualifier from the target type. For example, warn if a `const char *` is cast to an ordinary `char *`.

−Wcast−align
Warn whenever a pointer is cast such that the required alignment of the target is increased. For example, warn if a `char *` is cast to an `int *` on machines where integers can only be accessed at two− or four-byte boundaries.

−Wwrite−strings
When compiling C, give string constants the type `const char[`*`length`*`]` so that copying the address of one into a non−`const char *` pointer will get a warning; when compiling C++, warn about the deprecated conversion from string constants to `char *`. These warnings will help you find at compile time code that can try to write into a string constant, but only if you have been very careful about using `const` in declarations and prototypes. Otherwise, it will just be a nuisance; this is why we did not make **−Wall** request these warnings.

−Wconversion
Warn if a prototype causes a type conversion that is different from what would happen to the same argument in the absence of a prototype. This includes conversions of fixed point to floating and vice versa, and conversions changing the width or signedness of a fixed point argument except when the same as the default promotion.

Also, warn if a negative integer constant expression is implicitly converted to an unsigned type. For example, warn about the assignment `x = −1` if `x` is unsigned. But do not warn about explicit casts like `(unsigned) −1`.

−Wsign−compare
Warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned. This warning is also enabled by **−Wextra**; to get the other warnings of **−Wextra** without this warning, use **−Wextra −Wno−sign−compare**.

−Waggregate−return
Warn if any functions that return structures or unions are defined or called. (In languages where you can return an array, this also elicits a warning.)

−Wno−attributes
Do not warn if an unexpected `__attribute__` is used, such as unrecognized attributes, function attributes applied to variables, etc. This will not stop errors for incorrect use of supported attributes.

−Wstrict−prototypes (C only)
Warn if a function is declared or defined without specifying the argument types. (An old-style function definition is permitted without a warning if preceded by a declaration which specifies the argument types.)

−Wold−style−definition (C only)
Warn if an old-style function definition is used. A warning is given even if there is a previous prototype.

−Wmissing−prototypes (C only)
Warn if a global function is defined without a previous prototype declaration. This warning is issued even if the definition itself provides a prototype. The aim is to detect global functions that fail to be declared in header files.

−Wmissing−declarations (C only)
Warn if a global function is defined without a previous declaration. Do so even if the definition itself provides a prototype. Use this option to detect global functions that are not declared in header files.

−Wmissing−field−initializers

Warn if a structure's initializer has some fields missing. For example, the following code would cause such a warning, because `x.h` is implicitly zero:

```
struct s { int f, g, h; };
struct s x = { 3, 4 };
```

This option does not warn about designated initializers, so the following modification would not trigger a warning:

```
struct s { int f, g, h; };
struct s x = { .f = 3, .g = 4 };
```

This warning is included in **−Wextra**. To get other **−Wextra** warnings without this one, use **−Wextra −Wno−missing−field−initializers**.

−Wmissing−noreturn

Warn about functions which might be candidates for attribute `noreturn`. Note these are only possible candidates, not absolute ones. Care should be taken to manually verify functions actually do not ever return before adding the `noreturn` attribute, otherwise subtle code generation bugs could be introduced. You will not get a warning for `main` in hosted C environments.

−Wmissing−format−attribute

Warn about function pointers which might be candidates for `format` attributes. Note these are only possible candidates, not absolute ones. GCC will guess that function pointers with `format` attributes that are used in assignment, initialization, parameter passing or return statements should have a corresponding `format` attribute in the resulting type. I.e. the left-hand side of the assignment or initialization, the type of the parameter variable, or the return type of the containing function respectively should also have a `format` attribute to avoid the warning.

GCC will also warn about function definitions which might be candidates for `format` attributes. Again, these are only possible candidates. GCC will guess that `format` attributes might be appropriate for any function that calls a function like `vprintf` or `vscanf`, but this might not always be the case, and some functions for which `format` attributes are appropriate may not be detected.

−Wno−multichar

Do not warn if a multicharacter constant (**'FOOF'**) is used. Usually they indicate a typo in the user's code, as they have implementation-defined values, and should not be used in portable code.

−Wnormalized=<none|id|nfc|nfkc>

In ISO C and ISO C++, two identifiers are different if they are different sequences of characters. However, sometimes when characters outside the basic ASCII character set are used, you can have two different character sequences that look the same. To avoid confusion, the ISO 10646 standard sets out some *normalization rules* which when applied ensure that two sequences that look the same are turned into the same sequence. GCC can warn you if you are using identifiers which have not been normalized; this option controls that warning.

There are four levels of warning that GCC supports. The default is **−Wnormalized=nfc**, which warns about any identifier which is not in the ISO 10646 "C" normalized form, *NFC*. NFC is the recommended form for most uses.

Unfortunately, there are some characters which ISO C and ISO C++ allow in identifiers that when turned into NFC aren't allowable as identifiers. That is, there's no way to use these symbols in portable ISO C or C++ and have all your identifiers in NFC. **−Wnormalized=id** suppresses the warning for these characters. It is hoped that future versions of the standards involved will correct this, which is why this option is not the default.

You can switch the warning off for all characters by writing **−Wnormalized=none**. You would only want to do this if you were using some other normalization scheme (like "D"), because otherwise you can easily create bugs that are literally impossible to see.

Some characters in ISO 10646 have distinct meanings but look identical in some fonts or display methodologies, especially once formatting has been applied. For instance `\u207F`, ''SUPERSCRIPT LATIN SMALL LETTER N'', will display just like a regular `n` which has been placed in a superscript. ISO 10646 defines the *NFKC* normalisation scheme to convert all these into a standard form as well, and GCC will warn if your code is not in NFKC if you use **−Wnormalized=nfkc**. This warning is comparable to warning about every identifier that contains the letter O because it might be confused with the digit 0, and so is not the default, but may be useful as a local coding convention if the programming environment is unable to be fixed to display these characters distinctly.

−Wno−deprecated−declarations
Do not warn about uses of functions, variables, and types marked as deprecated by using the `deprecated` attribute. (@pxref{Function Attributes}, `@pxref`{Variable Attributes}, `@pxref`{Type Attributes}.)

−Wpacked
Warn if a structure is given the packed attribute, but the packed attribute has no effect on the layout or size of the structure. Such structures may be mis-aligned for little benefit. For instance, in this code, the variable `f.x` in `struct bar` will be misaligned even though `struct bar` does not itself have the packed attribute:

```
struct foo {
  int x;
  char a, b, c, d;
} __attribute__((packed));
struct bar {
  char z;
  struct foo f;
};
```

−Wpadded
Warn if padding is included in a structure, either to align an element of the structure or to align the whole structure. Sometimes when this happens it is possible to rearrange the fields of the structure to reduce the padding and so make the structure smaller.

−Wredundant−decls
Warn if anything is declared more than once in the same scope, even in cases where multiple declaration is valid and changes nothing.

−Wnested−externs (C only)
Warn if an `extern` declaration is encountered within a function.

−Wunreachable−code
Warn if the compiler detects that code will never be executed.

This option is intended to warn when the compiler detects that at least a whole line of source code will never be executed, because some condition is never satisfied or because it is after a procedure that never returns.

It is possible for this option to produce a warning even though there are circumstances under which part of the affected line can be executed, so care should be taken when removing apparently-unreachable code.

For instance, when a function is inlined, a warning may mean that the line is unreachable in only one inlined copy of the function.

This option is not made part of **−Wall** because in a debugging version of a program there is often substantial code which checks correct functioning of the program and is, hopefully, unreachable because the program does work. Another common use of unreachable code is to provide behavior which is selectable at compile−time.

−Winline
Warn if a function can not be inlined and it was declared as inline. Even with this option, the compiler will not warn about failures to inline functions declared in system headers.

The compiler uses a variety of heuristics to determine whether or not to inline a function. For example, the compiler takes into account the size of the function being inlined and the amount of inlining that has already been done in the current function. Therefore, seemingly insignificant changes in the source program can cause the warnings produced by **−Winline** to appear or disappear.

−Wno−invalid−offsetof (C++ only)
Suppress warnings from applying the **offsetof** macro to a non-POD type. According to the 1998 ISO C++ standard, applying **offsetof** to a non-POD type is undefined. In existing C++ implementations, however, **offsetof** typically gives meaningful results even when applied to certain kinds of non-POD types. (Such as a simple **struct** that fails to be a POD type only by virtue of having a constructor.) This flag is for users who are aware that they are writing nonportable code and who have deliberately chosen to ignore the warning about it.

The restrictions on **offsetof** may be relaxed in a future version of the C++ standard.

−Wno−int−to−pointer−cast (C only)
Suppress warnings from casts to pointer type of an integer of a different size.

−Wno−pointer−to−int−cast (C only)
Suppress warnings from casts from a pointer to an integer type of a different size.

−Winvalid−pch
Warn if a precompiled header is found in the search path but can't be used.

−Wlong−long
Warn if **long long** type is used. This is default. To inhibit the warning messages, use **−Wno−long−long**. Flags **−Wlong−long** and **−Wno−long−long** are taken into account only when **−pedantic** flag is used.

−Wvariadic−macros
Warn if variadic macros are used in pedantic ISO C90 mode, or the GNU alternate syntax when in pedantic ISO C99 mode. This is default. To inhibit the warning messages, use **−Wno−variadic−macros**.

−Wvolatile−register−var
Warn if a register variable is declared volatile. The volatile modifier does not inhibit all optimizations that may eliminate reads and/or writes to register variables.

−Wdisabled−optimization
Warn if a requested optimization pass is disabled. This warning does not generally indicate that there is anything wrong with your code; it merely indicates that GCC's optimizers were unable to handle the code effectively. Often, the problem is that your code is too big or too complex; GCC will refuse to optimize programs when the optimization itself is likely to take inordinate amounts of time.

−Wpointer−sign
Warn for pointer argument passing or assignment with different signedness. This option is only supported for C and Objective−C. It is implied by **−Wall** and by **−pedantic**, which can be disabled with **−Wno−pointer−sign**.

−Werror
Make all warnings into errors.

−Wstack−protector
This option is only active when **−fstack−protector** is active. It warns about functions that will not be protected against stack smashing.

Options for Debugging Your Program or GCC

GCC has various special options that are used for debugging either your program or GCC:

−g Produce debugging information in the operating system's native format (stabs, COFF, XCOFF, or DWARF 2). GDB can work with this debugging information.

On most systems that use stabs format, **−g** enables use of extra debugging information that only GDB can use; this extra information makes debugging work better in GDB but will probably make other debuggers crash or refuse to read the program. If you want to control for certain whether to generate the extra information, use **−gstabs+**, **−gstabs**, **−gxcoff+**, **−gxcoff**, or **−gvms** (see below).

GCC allows you to use **−g** with **−O**. The shortcuts taken by optimized code may occasionally produce surprising results: some variables you declared may not exist at all; flow of control may briefly move where you did not expect it; some statements may not be executed because they compute constant results or their values were already at hand; some statements may execute in different places because they were moved out of loops.

Nevertheless it proves possible to debug optimized output. This makes it reasonable to use the optimizer for programs that might have bugs.

The following options are useful when GCC is generated with the capability for more than one debugging format.

−ggdb
Produce debugging information for use by GDB. This means to use the most expressive format available (DWARF 2, stabs, or the native format if neither of those are supported), including GDB extensions if at all possible.

−gstabs
Produce debugging information in stabs format (if that is supported), without GDB extensions. This is the format used by DBX on most BSD systems. On MIPS, Alpha and System V Release 4 systems this option produces stabs debugging output which is not understood by DBX or SDB. On System V Release 4 systems this option requires the GNU assembler.

−feliminate−unused−debug−symbols
Produce debugging information in stabs format (if that is supported), for only symbols that are actually used.

−gstabs+
Produce debugging information in stabs format (if that is supported), using GNU extensions understood only by the GNU debugger (GDB). The use of these extensions is likely to make other debuggers crash or refuse to read the program.

−gcoff
Produce debugging information in COFF format (if that is supported). This is the format used by SDB on most System V systems prior to System V Release 4.

−gxcoff
Produce debugging information in XCOFF format (if that is supported). This is the format used by the DBX debugger on IBM RS/6000 systems.

−gxcoff+
Produce debugging information in XCOFF format (if that is supported), using GNU extensions understood only by the GNU debugger (GDB). The use of these extensions is likely to make other debuggers crash or refuse to read the program, and may cause assemblers other than the GNU assembler (GAS) to fail with an error.

−gdwarf−2
Produce debugging information in DWARF version 2 format (if that is supported). This is the format used by DBX on IRIX 6. With this option, GCC uses features of DWARF version 3 when they are useful; version 3 is upward compatible with version 2, but may still cause problems for older debuggers.

−gvms
Produce debugging information in VMS debug format (if that is supported). This is the format used by DEBUG on VMS systems.

−g*level*
−ggdb*level*
−gstabs*level*
−gcoff*level*
−gxcoff*level*
−gvms*level*
Request debugging information and also use *level* to specify how much information. The default level is 2.

Level 1 produces minimal information, enough for making backtraces in parts of the program that you don't plan to debug. This includes descriptions of functions and external variables, but no information about local variables and no line numbers.

Level 3 includes extra information, such as all the macro definitions present in the program. Some debuggers support macro expansion when you use **−g3**.

−gdwarf−2 does not accept a concatenated debug level, because GCC used to support an option **−gdwarf** that meant to generate debug information in version 1 of the DWARF format (which is very different from version 2), and it would have been too confusing. That debug format is long obsolete, but the option cannot be changed now. Instead use an additional **−g***level* option to change the debug level for DWARF2.

−feliminate−dwarf2−dups
Compress DWARF2 debugging information by eliminating duplicated information about each symbol. This option only makes sense when generating DWARF2 debugging information with **−gdwarf−2**.

−p Generate extra code to write profile information suitable for the analysis program **prof**. You must use this option when compiling the source files you want data about, and you must also use it when linking.

−pg
Generate extra code to write profile information suitable for the analysis program **gprof**. You must use this option when compiling the source files you want data about, and you must also use it when linking.

−Q Makes the compiler print out each function name as it is compiled, and print some statistics about each pass when it finishes.

−ftime−report
Makes the compiler print some statistics about the time consumed by each pass when it finishes.

−fmem−report
Makes the compiler print some statistics about permanent memory allocation when it finishes.

−fprofile−arcs
Add code so that program flow *arcs* are instrumented. During execution the program records how many times each branch and call is executed and how many times it is taken or returns. When the compiled program exits it saves this data to a file called *auxname.gcda* for each source file. The data may be used for profile-directed optimizations (**−fbranch−probabilities**), or for test coverage analysis (**−ftest−coverage**). Each object file's *auxname* is generated from the name of the output file, if explicitly specified and it is not the final executable, otherwise it is the basename of the source file. In both cases any suffix is removed (e.g. *foo.gcda* for input file *dir/foo.c*, or *dir/foo.gcda* for output file specified as **−o dir/foo.o**).

−−coverage
This option is used to compile and link code instrumented for coverage analysis. The option is a synonym for **−fprofile−arcs −ftest−coverage** (when compiling) and **−lgcov** (when linking). See the

documentation for those options for more details.

@bullet
Compile the source files with **−fprofile−arcs** plus optimization and code generation options. For test coverage analysis, use the additional **−ftest−coverage** option. You do not need to profile every source file in a program.

@cvmmfu
Link your object files with **−lgcov** or **−fprofile−arcs** (the latter implies the former).

@dwnngv
Run the program on a representative workload to generate the arc profile information. This may be repeated any number of times. You can run concurrent instances of your program, and provided that the file system supports locking, the data files will be correctly updated. Also `fork` calls are detected and correctly handled (double counting will not happen).

@exoohw
For profile-directed optimizations, compile the source files again with the same optimization and code generation options plus **−fbranch−probabilities**.

@fyppix
For test coverage analysis, use **gcov** to produce human readable information from the *.gcno* and *.gcda* files. Refer to the **gcov** documentation for further information.

With **−fprofile−arcs**, for each function of your program GCC creates a program flow graph, then finds a spanning tree for the graph. Only arcs that are not on the spanning tree have to be instrumented: the compiler adds code to count the number of times that these arcs are executed. When an arc is the only exit or only entrance to a block, the instrumentation code can be added to the block; otherwise, a new basic block must be created to hold the instrumentation code.

−ftest−coverage
Produce a notes file that the **gcov** code-coverage utility can use to show program coverage. Each source file's note file is called *auxname.gcno*. Refer to the **−fprofile−arcs** option above for a description of *auxname* and instructions on how to generate test coverage data. Coverage data will match the source files more closely, if you do not optimize.

−d*letters*
−fdump−rtl−*pass*
Says to make debugging dumps during compilation at times specified by *letters*. This is used for debugging the RTL-based passes of the compiler. The file names for most of the dumps are made by appending a pass number and a word to the *dumpname*. *dumpname* is generated from the name of the output file, if explicitly specified and it is not an executable, otherwise it is the basename of the source file.

Most debug dumps can be enabled either passing a letter to the **−d** option, or with a long **−fdump−rtl** switch; here are the possible letters for use in *letters* and *pass*, and their meanings:

−dA
Annotate the assembler output with miscellaneous debugging information.

−db
−fdump−rtl−bp
Dump after computing branch probabilities, to *file.09.bp*.

−dB
−fdump−rtl−bbro
Dump after block reordering, to *file.30.bbro*.

−dc
−fdump−rtl−combine
Dump after instruction combination, to the file *file.17.combine*.

−dC
−fdump−rtl−ce1
−fdump−rtl−ce2
−dC and **−fdump−rtl−ce1** enable dumping after the first if conversion, to the file *file.11.ce1*. **−dC** and **−fdump−rtl−ce2** enable dumping after the second if conversion, to the file *file.18.ce2*.

−dd
−fdump−rtl−btl
−fdump−rtl−dbr
−dd and **−fdump−rtl−btl** enable dumping after branch target load optimization, to *file.31.btl*. **−dd** and **−fdump−rtl−dbr** enable dumping after delayed branch scheduling, to *file.36.dbr*.

−dD
Dump all macro definitions, at the end of preprocessing, in addition to normal output.

−dE
−fdump−rtl−ce3
Dump after the third if conversion, to *file.28.ce3*.

−df
−fdump−rtl−cfg
−fdump−rtl−life
−df and **−fdump−rtl−cfg** enable dumping after control and data flow analysis, to *file.08.cfg*. **−df** and **−fdump−rtl−cfg** enable dumping dump after life analysis, to *file.16.life*.

−dg
−fdump−rtl−greg
Dump after global register allocation, to *file.23.greg*.

−dG
−fdump−rtl−gcse
−fdump−rtl−bypass
−dG and **−fdump−rtl−gcse** enable dumping after GCSE, to *file.05.gcse*. **−dG** and **−fdump−rtl−bypass** enable dumping after jump bypassing and control flow optimizations, to *file.07.bypass*.

−dh
−fdump−rtl−eh
Dump after finalization of EH handling code, to *file.02.eh*.

−di
−fdump−rtl−sibling
Dump after sibling call optimizations, to *file.01.sibling*.

−dj
−fdump−rtl−jump
Dump after the first jump optimization, to *file.03.jump*.

−dk
−fdump−rtl−stack
Dump after conversion from registers to stack, to *file.33.stack*.

−dl
−fdump−rtl−lreg
Dump after local register allocation, to *file.22.lreg*.

−dL
−fdump−rtl−loop
−fdump−rtl−loop2
−dL and **−fdump−rtl−loop** enable dumping after the first loop optimization pass, to *file.06.loop*. **−dL** and **−fdump−rtl−loop2** enable dumping after the second pass, to *file.13.loop2*.

−dm
−fdump−rtl−sms
Dump after modulo scheduling, to *file.20.sms*.

−dM
−fdump−rtl−mach
Dump after performing the machine dependent reorganization pass, to *file.35.mach*.

−dn
−fdump−rtl−rnreg
Dump after register renumbering, to *file.29.rnreg*.

−dN
−fdump−rtl−regmove
Dump after the register move pass, to *file.19.regmove*.

−do
−fdump−rtl−postreload
Dump after post-reload optimizations, to *file.24.postreload*.

−dr
−fdump−rtl−expand
Dump after RTL generation, to *file.00.expand*.

−dR
−fdump−rtl−sched2
Dump after the second scheduling pass, to *file.32.sched2*.

−ds
−fdump−rtl−cse
Dump after CSE (including the jump optimization that sometimes follows CSE), to *file.04.cse*.

−dS
−fdump−rtl−sched
Dump after the first scheduling pass, to *file.21.sched*.

−dt
−fdump−rtl−cse2
Dump after the second CSE pass (including the jump optimization that sometimes follows CSE), to *file.15.cse2*.

−dT
−fdump−rtl−tracer
Dump after running tracer, to *file.12.tracer*.

−dV
−fdump−rtl−vpt
−fdump−rtl−vartrack
−dV and **−fdump−rtl−vpt** enable dumping after the value profile transformations, to *file.10.vpt*.
−dV and **−fdump−rtl−vartrack** enable dumping after variable tracking, to *file.34.vartrack*.

−dw
−fdump−rtl−flow2
Dump after the second flow pass, to *file.26.flow2*.

−dz
−fdump−rtl−peephole2
Dump after the peephole pass, to *file.27.peephole2*.

−dZ
−fdump−rtl−web
Dump after live range splitting, to *file.14.web*.

−da
−fdump−rtl−all
Produce all the dumps listed above.

−dH
Produce a core dump whenever an error occurs.

−dm
Print statistics on memory usage, at the end of the run, to standard error.

−dp
Annotate the assembler output with a comment indicating which pattern and alternative was used. The length of each instruction is also printed.

−dP
Dump the RTL in the assembler output as a comment before each instruction. Also turns on **−dp** annotation.

−dv
For each of the other indicated dump files (either with **−d** or **−fdump−rtl−***pass*), dump a representation of the control flow graph suitable for viewing with VCG to *file.pass.vcg*.

−dx
Just generate RTL for a function instead of compiling it. Usually used with **r** (**−fdump−rtl−expand**).

−dy
Dump debugging information during parsing, to standard error.

−fdump−unnumbered
When doing debugging dumps (see **−d** option above), suppress instruction numbers and line number note output. This makes it more feasible to use diff on debugging dumps for compiler invocations with different options, in particular with and without **−g**.

−fdump−translation−unit (C++ only)
−fdump−translation−unit−*options* (C++ only)
Dump a representation of the tree structure for the entire translation unit to a file. The file name is made by appending *.tu* to the source file name. If the *−options* form is used, *options* controls the details of the dump as described for the **−fdump−tree** options.

−fdump−class−hierarchy (C++ only)
−fdump−class−hierarchy−*options* (C++ only)
Dump a representation of each class's hierarchy and virtual function table layout to a file. The file name is made by appending *.class* to the source file name. If the *−options* form is used, *options* controls the details of the dump as described for the **−fdump−tree** options.

−fdump−ipa−*switch*
Control the dumping at various stages of inter-procedural analysis language tree to a file. The file name is generated by appending a switch specific suffix to the source file name. The following dumps are possible:

all Enables all inter-procedural analysis dumps; currently the only produced dump is the **cgraph** dump.

cgraph
Dumps information about call-graph optimization, unused function removal, and inlining decisions.

−fdump−tree−*switch*
−fdump−tree−*switch−options*
Control the dumping at various stages of processing the intermediate language tree to a file. The file name is generated by appending a switch specific suffix to the source file name. If the *−options* form is used, *options* is a list of − separated options that control the details of the dump. Not all options are

applicable to all dumps, those which are not meaningful will be ignored. The following options are available

address
Print the address of each node. Usually this is not meaningful as it changes according to the environment and source file. Its primary use is for tying up a dump file with a debug environment.

slim
Inhibit dumping of members of a scope or body of a function merely because that scope has been reached. Only dump such items when they are directly reachable by some other path. When dumping pretty-printed trees, this option inhibits dumping the bodies of control structures.

raw
Print a raw representation of the tree. By default, trees are pretty-printed into a C−like representation.

details
Enable more detailed dumps (not honored by every dump option).

stats
Enable dumping various statistics about the pass (not honored by every dump option).

blocks
Enable showing basic block boundaries (disabled in raw dumps).

vops
Enable showing virtual operands for every statement.

lineno
Enable showing line numbers for statements.

uid Enable showing the unique ID (`DECL_UID`) for each variable.

all Turn on all options, except **raw**, **slim** and **lineno**.

The following tree dumps are possible:

original
Dump before any tree based optimization, to *file.original*.

optimized
Dump after all tree based optimization, to *file.optimized*.

inlined
Dump after function inlining, to *file.inlined*.

gimple
Dump each function before and after the gimplification pass to a file. The file name is made by appending *.gimple* to the source file name.

cfg Dump the control flow graph of each function to a file. The file name is made by appending *.cfg* to the source file name.

vcg Dump the control flow graph of each function to a file in VCG format. The file name is made by appending *.vcg* to the source file name. Note that if the file contains more than one function, the generated file cannot be used directly by VCG. You will need to cut and paste each function's graph into its own separate file first.

ch Dump each function after copying loop headers. The file name is made by appending *.ch* to the source file name.

ssa Dump SSA related information to a file. The file name is made by appending *.ssa* to the source file name.

salias
Dump structure aliasing variable information to a file. This file name is made by appending *.salias* to the source file name.

alias
Dump aliasing information for each function. The file name is made by appending *.alias* to the source file name.

ccp Dump each function after CCP. The file name is made by appending *.ccp* to the source file name.

storeccp
Dump each function after STORE−CCP. The file name is made by appending *.storeccp* to the source file name.

pre Dump trees after partial redundancy elimination. The file name is made by appending *.pre* to the source file name.

fre Dump trees after full redundancy elimination. The file name is made by appending *.fre* to the source file name.

copyprop
Dump trees after copy propagation. The file name is made by appending *.copyprop* to the source file name.

store_copyprop
Dump trees after store copy−propagation. The file name is made by appending *.store_copyprop* to the source file name.

dce Dump each function after dead code elimination. The file name is made by appending *.dce* to the source file name.

mudflap
Dump each function after adding mudflap instrumentation. The file name is made by appending *.mudflap* to the source file name.

sra Dump each function after performing scalar replacement of aggregates. The file name is made by appending *.sra* to the source file name.

sink
Dump each function after performing code sinking. The file name is made by appending *.sink* to the source file name.

dom
Dump each function after applying dominator tree optimizations. The file name is made by appending *.dom* to the source file name.

dse Dump each function after applying dead store elimination. The file name is made by appending *.dse* to the source file name.

phiopt
Dump each function after optimizing PHI nodes into straightline code. The file name is made by appending *.phiopt* to the source file name.

forwprop
Dump each function after forward propagating single use variables. The file name is made by appending *.forwprop* to the source file name.

copyrename
Dump each function after applying the copy rename optimization. The file name is made by appending *.copyrename* to the source file name.

nrv Dump each function after applying the named return value optimization on generic trees. The file name is made by appending *.nrv* to the source file name.

vect
Dump each function after applying vectorization of loops. The file name is made by appending *.vect* to the source file name.

vrp Dump each function after Value Range Propagation (VRP). The file name is made by appending *.vrp* to the source file name.

all Enable all the available tree dumps with the flags provided in this option.

−ftree−vectorizer−verbose=*n*
This option controls the amount of debugging output the vectorizer prints. This information is written to standard error, unless **−fdump−tree−all** or **−fdump−tree−vect** is specified, in which case it is output to the usual dump listing file, *.vect*.

−frandom−seed=*string*
This option provides a seed that GCC uses when it would otherwise use random numbers. It is used to generate certain symbol names that have to be different in every compiled file. It is also used to place unique stamps in coverage data files and the object files that produce them. You can use the **−frandom−seed** option to produce reproducibly identical object files.

The *string* should be different for every file you compile.

−fsched−verbose=*n*
On targets that use instruction scheduling, this option controls the amount of debugging output the scheduler prints. This information is written to standard error, unless **−dS** or **−dR** is specified, in which case it is output to the usual dump listing file, *.sched* or *.sched2* respectively. However for *n* greater than nine, the output is always printed to standard error.

For *n* greater than zero, **−fsched−verbose** outputs the same information as **−dRS**. For *n* greater than one, it also output basic block probabilities, detailed ready list information and unit/insn info. For *n* greater than two, it includes RTL at abort point, control-flow and regions info. And for *n* over four, **−fsched−verbose** also includes dependence info.

−save−temps
Store the usual ‘‘temporary’’ intermediate files permanently; place them in the current directory and name them based on the source file. Thus, compiling *foo.c* with **−c −save−temps** would produce files *foo.i* and *foo.s*, as well as *foo.o*. This creates a preprocessed *foo.i* output file even though the compiler now normally uses an integrated preprocessor.

When used in combination with the **−x** command line option, **−save−temps** is sensible enough to avoid over writing an input source file with the same extension as an intermediate file. The corresponding intermediate file may be obtained by renaming the source file before using **−save−temps**.

−time
Report the CPU time taken by each subprocess in the compilation sequence. For C source files, this is the compiler proper and assembler (plus the linker if linking is done). The output looks like this:

```
# cc1 0.12 0.01
# as 0.00 0.01
```

The first number on each line is the ‘‘user time’’, that is time spent executing the program itself. The second number is ‘‘system time’’, time spent executing operating system routines on behalf of the program. Both numbers are in seconds.

−fvar−tracking
Run variable tracking pass. It computes where variables are stored at each position in code. Better debugging information is then generated (if the debugging information format supports this information).

It is enabled by default when compiling with optimization (**−Os**, **−O**, **−O2**, ...), debugging information (**−g**) and the debug info format supports it.

−print−file−name=*library*
Print the full absolute name of the library file *library* that would be used when linking−−−and don't do anything else. With this option, GCC does not compile or link anything; it just prints the file name.

−print−multi−directory
Print the directory name corresponding to the multilib selected by any other switches present in the command line. This directory is supposed to exist in **GCC_EXEC_PREFIX**.

−print−multi−lib
Print the mapping from multilib directory names to compiler switches that enable them. The directory name is separated from the switches by **;**, and each switch starts with an **@} instead of the @samp{−**, without spaces between multiple switches. This is supposed to ease shell−processing.

−print−prog−name=*program*
Like **−print−file−name**, but searches for a program such as **cpp**.

−print−libgcc−file−name
Same as **−print−file−name=libgcc.a**.

This is useful when you use **−nostdlib** or **−nodefaultlibs** but you do want to link with *libgcc.a*. You can do

```
gcc -nostdlib <files>... ‘gcc -print-libgcc-file-name‘
```

−print−search−dirs
Print the name of the configured installation directory and a list of program and library directories **gcc** will search−−−and don't do anything else.

This is useful when **gcc** prints the error message **installation problem, cannot exec cpp0: No such file or directory**. To resolve this you either need to put *cpp0* and the other compiler components where **gcc** expects to find them, or you can set the environment variable **GCC_EXEC_PREFIX** to the directory where you installed them. Don't forget the trailing /.

−dumpmachine
Print the compiler's target machine (for example, **i686−pc−linux−gnu**)−−−and don't do anything else.

−dumpversion
Print the compiler version (for example, **3.0**)−−−and don't do anything else.

−dumpspecs
Print the compiler's built-in specs−−−and don't do anything else. (This is used when GCC itself is being built.)

−feliminate−unused−debug−types
Normally, when producing DWARF2 output, GCC will emit debugging information for all types declared in a compilation unit, regardless of whether or not they are actually used in that compilation unit. Sometimes this is useful, such as if, in the debugger, you want to cast a value to a type that is not actually used in your program (but is declared). More often, however, this results in a significant amount of wasted space. With this option, GCC will avoid producing debug symbol output for types that are nowhere used in the source file being compiled.

Options That Control Optimization

These options control various sorts of optimizations.

Without any optimization option, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results. Statements are independent: if you stop the program with a breakpoint between statements, you can then assign a new value to any variable or change the program counter to any other statement in the function and get exactly the results you would expect from the source code.

Turning on optimization flags makes the compiler attempt to improve the performance and/or code size at the expense of compilation time and possibly the ability to debug the program.

The compiler performs optimization based on the knowledge it has of the program. Optimization levels

−O2 and above, in particular, enable *unit-at-a-time* mode, which allows the compiler to consider information gained from later functions in the file when compiling a function. Compiling multiple files at once to a single output file in *unit-at-a-time* mode allows the compiler to use information gained from all of the files when compiling each of them.

Not all optimizations are controlled directly by a flag. Only optimizations that have a flag are listed.

−O
−O1

Optimize. Optimizing compilation takes somewhat more time, and a lot more memory for a large function.

With **−O**, the compiler tries to reduce code size and execution time, without performing any optimizations that take a great deal of compilation time.

−O turns on the following optimization flags: **−fdefer−pop −fdelayed−branch −fguess−branch−probability −fcprop−registers −floop−optimize −fif−conversion −fif−conversion2 −ftree−ccp −ftree−dce −ftree−dominator−opts −ftree−dse −ftree−ter −ftree−lrs −ftree−sra −ftree−copyrename −ftree−fre −ftree−ch −fmerge−constants**

−O also turns on **−fomit−frame−pointer** on machines where doing so does not interfere with debugging.

−O doesn't turn on **−ftree−sra** for the Ada compiler. This option must be explicitly specified on the command line to be enabled for the Ada compiler.

−O2

Optimize even more. GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. The compiler does not perform loop unrolling or function inlining when you specify **−O2**. As compared to **−O**, this option increases both compilation time and the performance of the generated code.

−O2 turns on all optimization flags specified by **−O**. It also turns on the following optimization flags: **−fthread−jumps −fcrossjumping −foptimize−sibling−calls −fcse−follow−jumps −fcse−skip−blocks −fgcse −fgcse−lm −fexpensive−optimizations −fstrength−reduce −frerun−cse−after−loop −frerun−loop−opt −fcaller−saves −fpeephole2 −fschedule−insns −fschedule−insns2 −fsched−interblock −fsched−spec −fregmove −fstrict−aliasing −fdelete−null−pointer−checks −freorder−blocks −freorder−functions −funit−at−a−time −falign−functions −falign−jumps −falign−loops −falign−labels −ftree−vrp −ftree−pre**

Please note the warning under **−fgcse** about invoking **−O2** on programs that use computed gotos.

−O3

Optimize yet more. **−O3** turns on all optimizations specified by **−O2** and also turns on the **−finline−functions**, **−funswitch−loops** and **−fgcse−after−reload** options.

−O0

Do not optimize. This is the default.

−Os

Optimize for size. **−Os** enables all **−O2** optimizations that do not typically increase code size. It also performs further optimizations designed to reduce code size.

−Os disables the following optimization flags: **−falign−functions −falign−jumps −falign−loops −falign−labels −freorder−blocks −freorder−blocks−and−partition −fprefetch−loop−arrays −ftree−vect−loop−version**

If you use multiple **−O** options, with or without level numbers, the last such option is the one that is effective.

Options of the form **−f***flag* specify machine-independent flags. Most flags have both positive and negative forms; the negative form of **−ffoo** would be **−fno−foo**. In the table below, only one of the forms is listed−−−the one you typically will use. You can figure out the other form by either removing **no−** or

adding it.

The following options control specific optimizations. They are either activated by **−O** options or are related to ones that are. You can use the following flags in the rare cases when ‘‘fine−tuning’’ of optimizations to be performed is desired.

−fno−default−inline
Do not make member functions inline by default merely because they are defined inside the class scope (C++ only). Otherwise, when you specify **−O**, member functions defined inside class scope are compiled inline by default; i.e., you don’t need to add **inline** in front of the member function name.

−fno−defer−pop
Always pop the arguments to each function call as soon as that function returns. For machines which must pop arguments after a function call, the compiler normally lets arguments accumulate on the stack for several function calls and pops them all at once.

Disabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−fforce−mem
Force memory operands to be copied into registers before doing arithmetic on them. This produces better code by making all memory references potential common subexpressions. When they are not common subexpressions, instruction combination should eliminate the separate register−load. This option is now a nop and will be removed in 4.2.

−fforce−addr
Force memory address constants to be copied into registers before doing arithmetic on them.

−fomit−frame−pointer
Don’t keep the frame pointer in a register for functions that don’t need one. This avoids the instructions to save, set up and restore frame pointers; it also makes an extra register available in many functions. **It also makes debugging impossible on some machines.**

On some machines, such as the VAX, this flag has no effect, because the standard calling sequence automatically handles the frame pointer and nothing is saved by pretending it doesn’t exist. The machine-description macro `FRAME_POINTER_REQUIRED` controls whether a target machine supports this flag.

Enabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−foptimize−sibling−calls
Optimize sibling and tail recursive calls.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fno−inline
Don’t pay attention to the `inline` keyword. Normally this option is used to keep the compiler from expanding any functions inline. Note that if you are not optimizing, no functions can be expanded inline.

−finline−functions
Integrate all simple functions into their callers. The compiler heuristically decides which functions are simple enough to be worth integrating in this way.

If all calls to a given function are integrated, and the function is declared `static`, then the function is normally not output as assembler code in its own right.

Enabled at level **−O3**.

−finline−functions−called−once
Consider all `static` functions called once for inlining into their caller even if they are not marked `inline`. If a call to a given function is integrated, then the function is not output as assembler code in its own right.

Enabled if **−funit−at−a−time** is enabled.

−fearly−inlining
Inline functions marked by `always_inline` and functions whose body seems smaller than the function call overhead early before doing **−fprofile−generate** instrumentation and real inlining pass. Doing so makes profiling significantly cheaper and usually inlining faster on programs having large chains of nested wrapper functions.

Enabled by default.

−finline−limit=*n*
By default, GCC limits the size of functions that can be inlined. This flag allows the control of this limit for functions that are explicitly marked as inline (i.e., marked with the inline keyword or defined within the class definition in c++). *n* is the size of functions that can be inlined in number of pseudo instructions (not counting parameter handling). The default value of *n* is 600. Increasing this value can result in more inlined code at the cost of compilation time and memory consumption. Decreasing usually makes the compilation faster and less code will be inlined (which presumably means slower programs). This option is particularly useful for programs that use inlining heavily such as those based on recursive templates with C++.

Inlining is actually controlled by a number of parameters, which may be specified individually by using **−−param** *name*=*value*. The **−finline−limit=***n* option sets some of these parameters as follows:

```
@item max-inline-insns-single
 is set to I<n>/2.
@item max-inline-insns-auto
 is set to I<n>/2.
@item min-inline-insns
 is set to 130 or I<n>/4, whichever is smaller.
@item max-inline-insns-rtl
 is set to I<n>.
```

See below for a documentation of the individual parameters controlling inlining.

Note: pseudo instruction represents, in this particular context, an abstract measurement of function's size. In no way does it represent a count of assembly instructions and as such its exact meaning might change from one release to an another.

−fkeep−inline−functions
In C, emit `static` functions that are declared `inline` into the object file, even if the function has been inlined into all of its callers. This switch does not affect functions using the `extern inline` extension in GNU C. In C++, emit any and all inline functions into the object file.

−fkeep−static−consts
Emit variables declared `static const` when optimization isn't turned on, even if the variables aren't referenced.

GCC enables this option by default. If you want to force the compiler to check if the variable was referenced, regardless of whether or not optimization is turned on, use the **−fno−keep−static−consts** option.

−fmerge−constants
Attempt to merge identical constants (string constants and floating point constants) across compilation units.

This option is the default for optimized compilation if the assembler and linker support it. Use **−fno−merge−constants** to inhibit this behavior.

Enabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−fmerge−all−constants
Attempt to merge identical constants and identical variables.

This option implies **−fmerge−constants**. In addition to **−fmerge−constants** this considers e.g. even

constant initialized arrays or initialized constant variables with integral or floating point types. Languages like C or C++ require each non-automatic variable to have distinct location, so using this option will result in non-conforming behavior.

−fmodulo−sched
Perform swing modulo scheduling immediately before the first scheduling pass. This pass looks at innermost loops and reorders their instructions by overlapping different iterations.

−fno−branch−count−reg
Do not use ‘‘decrement and branch’’ instructions on a count register, but instead generate a sequence of instructions that decrement a register, compare it against zero, then branch based upon the result. This option is only meaningful on architectures that support such instructions, which include x86, PowerPC, IA−64 and S/390.

The default is **−fbranch−count−reg**, enabled when **−fstrength−reduce** is enabled.

−fno−function−cse
Do not put function addresses in registers; make each instruction that calls a constant function contain the function’s address explicitly.

This option results in less efficient code, but some strange hacks that alter the assembler output may be confused by the optimizations performed when this option is not used.

The default is **−ffunction−cse**

−fno−zero−initialized−in−bss
If the target supports a BSS section, GCC by default puts variables that are initialized to zero into BSS. This can save space in the resulting code.

This option turns off this behavior because some programs explicitly rely on variables going to the data section. E.g., so that the resulting executable can find the beginning of that section and/or make assumptions based on that.

The default is **−fzero−initialized−in−bss**.

−fbounds−check
For front-ends that support it, generate additional code to check that indices used to access arrays are within the declared range. This is currently only supported by the Java and Fortran front−ends, where this option defaults to true and false respectively.

−fmudflap −fmudflapth −fmudflapir
For front-ends that support it (C and C++), instrument all risky pointer/array dereferencing operations, some standard library string/heap functions, and some other associated constructs with range/validity tests. Modules so instrumented should be immune to buffer overflows, invalid heap use, and some other classes of C/C++ programming errors. The instrumentation relies on a separate runtime library (*libmudflap*), which will be linked into a program if **−fmudflap** is given at link time. Run-time behavior of the instrumented program is controlled by the **MUDFLAP_OPTIONS** environment variable. See `env MUDFLAP_OPTIONS=−help a.out` for its options.

Use **−fmudflapth** instead of **−fmudflap** to compile and to link if your program is multi−threaded. Use **−fmudflapir**, in addition to **−fmudflap** or **−fmudflapth**, if instrumentation should ignore pointer reads. This produces less instrumentation (and therefore faster execution) and still provides some protection against outright memory corrupting writes, but allows erroneously read data to propagate within a program.

−fstrength−reduce
Perform the optimizations of loop strength reduction and elimination of iteration variables.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fthread−jumps
Perform optimizations where we check to see if a jump branches to a location where another comparison subsumed by the first is found. If so, the first branch is redirected to either the destination of the

second branch or a point immediately following it, depending on whether the condition is known to be true or false.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fcse−follow−jumps
In common subexpression elimination, scan through jump instructions when the target of the jump is not reached by any other path. For example, when CSE encounters an `if` statement with an `else` clause, CSE will follow the jump when the condition tested is false.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fcse−skip−blocks
This is similar to **−fcse−follow−jumps**, but causes CSE to follow jumps which conditionally skip over blocks. When CSE encounters a simple `if` statement with no else clause, **−fcse−skip−blocks** causes CSE to follow the jump around the body of the `if`.

Enabled at levels **−O2**, **−O3**, **−Os**.

−frerun−cse−after−loop
Re-run common subexpression elimination after loop optimizations has been performed.

Enabled at levels **−O2**, **−O3**, **−Os**.

−frerun−loop−opt
Run the loop optimizer twice.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fgcse
Perform a global common subexpression elimination pass. This pass also performs global constant and copy propagation.

Note: When compiling a program using computed gotos, a GCC extension, you may get better runtime performance if you disable the global common subexpression elimination pass by adding **−fno−gcse** to the command line.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fgcse−lm
When **−fgcse−lm** is enabled, global common subexpression elimination will attempt to move loads which are only killed by stores into themselves. This allows a loop containing a load/store sequence to be changed to a load outside the loop, and a copy/store within the loop.

Enabled by default when gcse is enabled.

−fgcse−sm
When **−fgcse−sm** is enabled, a store motion pass is run after global common subexpression elimination. This pass will attempt to move stores out of loops. When used in conjunction with **−fgcse−lm**, loops containing a load/store sequence can be changed to a load before the loop and a store after the loop.

Not enabled at any optimization level.

−fgcse−las
When **−fgcse−las** is enabled, the global common subexpression elimination pass eliminates redundant loads that come after stores to the same memory location (both partial and full redundancies).

Not enabled at any optimization level.

−fgcse−after−reload
When **−fgcse−after−reload** is enabled, a redundant load elimination pass is performed after reload. The purpose of this pass is to cleanup redundant spilling.

−floop−optimize
Perform loop optimizations: move constant expressions out of loops, simplify exit test conditions and optionally do strength-reduction as well.

Enabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−floop−optimize2
Perform loop optimizations using the new loop optimizer. The optimizations (loop unrolling, peeling and unswitching, loop invariant motion) are enabled by separate flags.

−funsafe−loop−optimizations
If given, the loop optimizer will assume that loop indices do not overflow, and that the loops with non-trivial exit condition are not infinite. This enables a wider range of loop optimizations even if the loop optimizer itself cannot prove that these assumptions are valid. Using **−Wunsafe−loop−optimizations**, the compiler will warn you if it finds this kind of loop.

−fcrossjumping
Perform cross-jumping transformation. This transformation unifies equivalent code and save code size. The resulting code may or may not perform better than without cross−jumping.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fif−conversion
Attempt to transform conditional jumps into branch-less equivalents. This include use of conditional moves, min, max, set flags and abs instructions, and some tricks doable by standard arithmetics. The use of conditional execution on chips where it is available is controlled by `if−conversion2`.

Enabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−fif−conversion2
Use conditional execution (where available) to transform conditional jumps into branch-less equivalents.

Enabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−fdelete−null−pointer−checks
Use global dataflow analysis to identify and eliminate useless checks for null pointers. The compiler assumes that dereferencing a null pointer would have halted the program. If a pointer is checked after it has already been dereferenced, it cannot be null.

In some environments, this assumption is not true, and programs can safely dereference null pointers. Use **−fno−delete−null−pointer−checks** to disable this optimization for programs which depend on that behavior.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fexpensive−optimizations
Perform a number of minor optimizations that are relatively expensive.

Enabled at levels **−O2**, **−O3**, **−Os**.

−foptimize−register−move
−fregmove
Attempt to reassign register numbers in move instructions and as operands of other simple instructions in order to maximize the amount of register tying. This is especially helpful on machines with two-operand instructions.

Note **−fregmove** and **−foptimize−register−move** are the same optimization.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fdelayed−branch
If supported for the target machine, attempt to reorder instructions to exploit instruction slots available after delayed branch instructions.

Enabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−fschedule−insns
If supported for the target machine, attempt to reorder instructions to eliminate execution stalls due to required data being unavailable. This helps machines that have slow floating point or memory load instructions by allowing other instructions to be issued until the result of the load or floating point instruction is required.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fschedule−insns2
Similar to **−fschedule−insns**, but requests an additional pass of instruction scheduling after register allocation has been done. This is especially useful on machines with a relatively small number of registers and where memory load instructions take more than one cycle.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fno−sched−interblock
Don't schedule instructions across basic blocks. This is normally enabled by default when scheduling before register allocation, i.e. with **−fschedule−insns** or at **−O2** or higher.

−fno−sched−spec
Don't allow speculative motion of non-load instructions. This is normally enabled by default when scheduling before register allocation, i.e. with **−fschedule−insns** or at **−O2** or higher.

−fsched−spec−load
Allow speculative motion of some load instructions. This only makes sense when scheduling before register allocation, i.e. with **−fschedule−insns** or at **−O2** or higher.

−fsched−spec−load−dangerous
Allow speculative motion of more load instructions. This only makes sense when scheduling before register allocation, i.e. with **−fschedule−insns** or at **−O2** or higher.

−fsched−stalled−insns=*n*
Define how many insns (if any) can be moved prematurely from the queue of stalled insns into the ready list, during the second scheduling pass.

−fsched−stalled−insns−dep=*n*
Define how many insn groups (cycles) will be examined for a dependency on a stalled insn that is candidate for premature removal from the queue of stalled insns. Has an effect only during the second scheduling pass, and only if **−fsched−stalled−insns** is used and its value is not zero.

−fsched2−use−superblocks
When scheduling after register allocation, do use superblock scheduling algorithm. Superblock scheduling allows motion across basic block boundaries resulting on faster schedules. This option is experimental, as not all machine descriptions used by GCC model the CPU closely enough to avoid unreliable results from the algorithm.

This only makes sense when scheduling after register allocation, i.e. with **−fschedule−insns2** or at **−O2** or higher.

−fsched2−use−traces
Use **−fsched2−use−superblocks** algorithm when scheduling after register allocation and additionally perform code duplication in order to increase the size of superblocks using tracer pass. See **−ftracer** for details on trace formation.

This mode should produce faster but significantly longer programs. Also without **−fbranch−probabilities** the traces constructed may not match the reality and hurt the performance. This only makes sense when scheduling after register allocation, i.e. with **−fschedule−insns2** or at **−O2** or higher.

−freschedule−modulo−scheduled−loops
The modulo scheduling comes before the traditional scheduling, if a loop was modulo scheduled we may want to prevent the later scheduling passes from changing its schedule, we use this option to

control that.

−fcaller−saves
Enable values to be allocated in registers that will be clobbered by function calls, by emitting extra instructions to save and restore the registers around such calls. Such allocation is done only when it seems to result in better code than would otherwise be produced.

This option is always enabled by default on certain machines, usually those which have no call-preserved registers to use instead.

Enabled at levels **−O2**, **−O3**, **−Os**.

−ftree−pre
Perform Partial Redundancy Elimination (PRE) on trees. This flag is enabled by default at **−O2** and **−O3**.

−ftree−fre
Perform Full Redundancy Elimination (FRE) on trees. The difference between FRE and PRE is that FRE only considers expressions that are computed on all paths leading to the redundant computation. This analysis faster than PRE, though it exposes fewer redundancies. This flag is enabled by default at **−O** and higher.

−ftree−copy−prop
Perform copy propagation on trees. This pass eliminates unnecessary copy operations. This flag is enabled by default at **−O** and higher.

−ftree−store−copy−prop
Perform copy propagation of memory loads and stores. This pass eliminates unnecessary copy operations in memory references (structures, global variables, arrays, etc). This flag is enabled by default at **−O2** and higher.

−ftree−salias
Perform structural alias analysis on trees. This flag is enabled by default at **−O** and higher.

−ftree−sink
Perform forward store motion on trees. This flag is enabled by default at **−O** and higher.

−ftree−ccp
Perform sparse conditional constant propagation (CCP) on trees. This pass only operates on local scalar variables and is enabled by default at **−O** and higher.

−ftree−store−ccp
Perform sparse conditional constant propagation (CCP) on trees. This pass operates on both local scalar variables and memory stores and loads (global variables, structures, arrays, etc). This flag is enabled by default at **−O2** and higher.

−ftree−dce
Perform dead code elimination (DCE) on trees. This flag is enabled by default at **−O** and higher.

−ftree−dominator−opts
Perform a variety of simple scalar cleanups (constant/copy propagation, redundancy elimination, range propagation and expression simplification) based on a dominator tree traversal. This also performs jump threading (to reduce jumps to jumps). This flag is enabled by default at **−O** and higher.

−ftree−ch
Perform loop header copying on trees. This is beneficial since it increases effectiveness of code motion optimizations. It also saves one jump. This flag is enabled by default at **−O** and higher. It is not enabled for **−Os**, since it usually increases code size.

−ftree−loop−optimize
Perform loop optimizations on trees. This flag is enabled by default at **−O** and higher.

−ftree−loop−linear
Perform linear loop transformations on tree. This flag can improve cache performance and allow further loop optimizations to take place.

−ftree−loop−im
Perform loop invariant motion on trees. This pass moves only invariants that would be hard to handle at RTL level (function calls, operations that expand to nontrivial sequences of insns). With **−funswitch−loops** it also moves operands of conditions that are invariant out of the loop, so that we can use just trivial invariantness analysis in loop unswitching. The pass also includes store motion.

−ftree−loop−ivcanon
Create a canonical counter for number of iterations in the loop for that determining number of iterations requires complicated analysis. Later optimizations then may determine the number easily. Useful especially in connection with unrolling.

−fivopts
Perform induction variable optimizations (strength reduction, induction variable merging and induction variable elimination) on trees.

−ftree−sra
Perform scalar replacement of aggregates. This pass replaces structure references with scalars to prevent committing structures to memory too early. This flag is enabled by default at **−O** and higher.

−ftree−copyrename
Perform copy renaming on trees. This pass attempts to rename compiler temporaries to other variables at copy locations, usually resulting in variable names which more closely resemble the original variables. This flag is enabled by default at **−O** and higher.

−ftree−ter
Perform temporary expression replacement during the SSA−>normal phase. Single use/single def temporaries are replaced at their use location with their defining expression. This results in non-GIMPLE code, but gives the expanders much more complex trees to work on resulting in better RTL generation. This is enabled by default at **−O** and higher.

−ftree−lrs
Perform live range splitting during the SSA−>normal phase. Distinct live ranges of a variable are split into unique variables, allowing for better optimization later. This is enabled by default at **−O** and higher.

−ftree−vectorize
Perform loop vectorization on trees.

−ftree−vect−loop−version
Perform loop versioning when doing loop vectorization on trees. When a loop appears to be vectorizable except that data alignment or data dependence cannot be determined at compile time then vectorized and non-vectorized versions of the loop are generated along with runtime checks for alignment or dependence to control which version is executed. This option is enabled by default except at level **−Os** where it is disabled.

−ftree−vrp
Perform Value Range Propagation on trees. This is similar to the constant propagation pass, but instead of values, ranges of values are propagated. This allows the optimizers to remove unnecessary range checks like array bound checks and null pointer checks. This is enabled by default at **−O2** and higher. Null pointer check elimination is only done if **−fdelete−null−pointer−checks** is enabled.

−ftracer
Perform tail duplication to enlarge superblock size. This transformation simplifies the control flow of the function allowing other optimizations to do better job.

−funroll−loops
Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop. **−funroll−loops** implies both **−fstrength−reduce** and **−frerun−cse−after−loop**. This option makes

code larger, and may or may not make it run faster.

−funroll−all−loops
Unroll all loops, even if their number of iterations is uncertain when the loop is entered. This usually makes programs run more slowly. **−funroll−all−loops** implies the same options as **−funroll−loops**,

−fsplit−ivs−in−unroller
Enables expressing of values of induction variables in later iterations of the unrolled loop using the value in the first iteration. This breaks long dependency chains, thus improving efficiency of the scheduling passes.

Combination of **−fweb** and CSE is often sufficient to obtain the same effect. However in cases the loop body is more complicated than a single basic block, this is not reliable. It also does not work at all on some of the architectures due to restrictions in the CSE pass.

This optimization is enabled by default.

−fvariable−expansion−in−unroller
With this option, the compiler will create multiple copies of some local variables when unrolling a loop which can result in superior code.

−fprefetch−loop−arrays
If supported by the target machine, generate instructions to prefetch memory to improve the performance of loops that access large arrays.

These options may generate better or worse code; results are highly dependent on the structure of loops within the source code.

−fno−peephole
−fno−peephole2
Disable any machine-specific peephole optimizations. The difference between **−fno−peephole** and **−fno−peephole2** is in how they are implemented in the compiler; some targets use one, some use the other, a few use both.

−fpeephole is enabled by default. **−fpeephole2** enabled at levels **−O2**, **−O3**, **−Os**.

−fno−guess−branch−probability
Do not guess branch probabilities using heuristics.

GCC will use heuristics to guess branch probabilities if they are not provided by profiling feedback (**−fprofile−arcs**). These heuristics are based on the control flow graph. If some branch probabilities are specified by **__builtin_expect**, then the heuristics will be used to guess branch probabilities for the rest of the control flow graph, taking the **__builtin_expect** info into account. The interactions between the heuristics and **__builtin_expect** can be complex, and in some cases, it may be useful to disable the heuristics so that the effects of **__builtin_expect** are easier to understand.

The default is **−fguess−branch−probability** at levels **−O**, **−O2**, **−O3**, **−Os**.

−freorder−blocks
Reorder basic blocks in the compiled function in order to reduce number of taken branches and improve code locality.

Enabled at levels **−O2**, **−O3**.

−freorder−blocks−and−partition
In addition to reordering basic blocks in the compiled function, in order to reduce number of taken branches, partitions hot and cold basic blocks into separate sections of the assembly and .o files, to improve paging and cache locality performance.

This optimization is automatically turned off in the presence of exception handling, for linkonce sections, for functions with a user-defined section attribute and on any architecture that does not support named sections.

−freorder−functions

Reorder functions in the object file in order to improve code locality. This is implemented by using special subsections `.text.hot` for most frequently executed functions and `.text.unlikely` for unlikely executed functions. Reordering is done by the linker so object file format must support named sections and linker must place them in a reasonable way.

Also profile feedback must be available in to make this option effective. See **−fprofile−arcs** for details.

Enabled at levels **−O2**, **−O3**, **−Os**.

−fstrict−aliasing

Allows the compiler to assume the strictest aliasing rules applicable to the language being compiled. For C (and C++), this activates optimizations based on the type of expressions. In particular, an object of one type is assumed never to reside at the same address as an object of a different type, unless the types are almost the same. For example, an `unsigned int` can alias an `int`, but not a `void*` or a `double`. A character type may alias any other type.

Pay special attention to code like this:

```
union a_union {
  int i;
  double d;
};

int f() {
  a_union t;
  t.d = 3.0;
  return t.i;
}
```

The practice of reading from a different union member than the one most recently written to (called ‘‘type−punning’’) is common. Even with **−fstrict−aliasing**, type-punning is allowed, provided the memory is accessed through the union type. So, the code above will work as expected. However, this code might not:

```
int f() {
  a_union t;
  int* ip;
  t.d = 3.0;
  ip = &t.i;
  return *ip;
}
```

Every language that wishes to perform language-specific alias analysis should define a function that computes, given an `tree` node, an alias set for the node. Nodes in different alias sets are not allowed to alias. For an example, see the C front-end function `c_get_alias_set`.

Enabled at levels **−O2**, **−O3**, **−Os**.

−falign−functions

−falign−functions=*n*

Align the start of functions to the next power-of-two greater than *n*, skipping up to *n* bytes. For instance, **−falign−functions=32** aligns functions to the next 32−byte boundary, but **−falign−functions=24** would align to the next 32−byte boundary only if this can be done by skipping 23 bytes or less.

−fno−align−functions and **−falign−functions=1** are equivalent and mean that functions will not be aligned.

Some assemblers only support this flag when *n* is a power of two; in that case, it is rounded up.

If *n* is not specified or is zero, use a machine-dependent default.

Enabled at levels **−O2**, **−O3**.

−falign−labels
−falign−labels=*n*
Align all branch targets to a power-of-two boundary, skipping up to *n* bytes like **−falign−functions**. This option can easily make code slower, because it must insert dummy operations for when the branch target is reached in the usual flow of the code.

−fno−align−labels and **−falign−labels=1** are equivalent and mean that labels will not be aligned.

If **−falign−loops** or **−falign−jumps** are applicable and are greater than this value, then their values are used instead.

If *n* is not specified or is zero, use a machine-dependent default which is very likely to be **1**, meaning no alignment.

Enabled at levels **−O2**, **−O3**.

−falign−loops
−falign−loops=*n*
Align loops to a power-of-two boundary, skipping up to *n* bytes like **−falign−functions**. The hope is that the loop will be executed many times, which will make up for any execution of the dummy operations.

−fno−align−loops and **−falign−loops=1** are equivalent and mean that loops will not be aligned.

If *n* is not specified or is zero, use a machine-dependent default.

Enabled at levels **−O2**, **−O3**.

−falign−jumps
−falign−jumps=*n*
Align branch targets to a power-of-two boundary, for branch targets where the targets can only be reached by jumping, skipping up to *n* bytes like **−falign−functions**. In this case, no dummy operations need be executed.

−fno−align−jumps and **−falign−jumps=1** are equivalent and mean that loops will not be aligned.

If *n* is not specified or is zero, use a machine-dependent default.

Enabled at levels **−O2**, **−O3**.

−funit−at−a−time
Parse the whole compilation unit before starting to produce code. This allows some extra optimizations to take place but consumes more memory (in general). There are some compatibility issues with *unit-at-at-time* mode:

* enabling *unit-at-a-time* mode may change the order in which functions, variables, and top-level `asm` statements are emitted, and will likely break code relying on some particular ordering. The majority of such top-level `asm` statements, though, can be replaced by `section` attributes.
* *unit-at-a-time* mode removes unreferenced static variables and functions. This may result in undefined references when an `asm` statement refers directly to variables or functions that are otherwise unused. In that case either the variable/function shall be listed as an operand of the `asm` statement operand or, in the case of top-level `asm` statements the attribute `used` shall be used on the declaration.
* Static functions now can use non-standard passing conventions that may break `asm` statements calling functions directly. Again, attribute `used` will prevent this behavior.

As a temporary workaround, **−fno−unit−at−a−time** can be used, but this scheme may not be supported by future releases of GCC.

Enabled at levels **−O2**, **−O3**.

−fweb
Constructs webs as commonly used for register allocation purposes and assign each web individual pseudo register. This allows the register allocation pass to operate on pseudos directly, but also strengthens several other optimization passes, such as CSE, loop optimizer and trivial dead code remover. It can, however, make debugging impossible, since variables will no longer stay in a ‘‘home register’’.

Enabled by default with **−funroll−loops**.

−fwhole−program
Assume that the current compilation unit represents whole program being compiled. All public functions and variables with the exception of `main` and those merged by attribute `externally_visible` become static functions and in a affect gets more aggressively optimized by interprocedural optimizers. While this option is equivalent to proper use of `static` keyword for programs consisting of single file, in combination with option **−−combine** this flag can be used to compile most of smaller scale C programs since the functions and variables become local for the whole combined compilation unit, not for the single source file itself.

−fno−cprop−registers
After register allocation and post-register allocation instruction splitting, we perform a copy-propagation pass to try to reduce scheduling dependencies and occasionally eliminate the copy.

Disabled at levels **−O**, **−O2**, **−O3**, **−Os**.

−fprofile−generate
Enable options usually used for instrumenting application to produce profile useful for later recompilation with profile feedback based optimization. You must use **−fprofile−generate** both when compiling and when linking your program.

The following options are enabled: `−fprofile−arcs`, `−fprofile−values`, `−fvpt`.

−fprofile−use
Enable profile feedback directed optimizations, and optimizations generally profitable only with profile feedback available.

The following options are enabled: `−fbranch−probabilities`, `−fvpt`, `−funroll−loops`, `−fpeel−loops`, `−ftracer`, `−fno−loop−optimize`.

The following options control compiler behavior regarding floating point arithmetic. These options trade off between speed and correctness. All must be specifically enabled.

−ffloat−store
Do not store floating point variables in registers, and inhibit other options that might change whether a floating point value is taken from a register or memory.

This option prevents undesirable excess precision on machines such as the 68000 where the floating registers (of the 68881) keep more precision than a `double` is supposed to have. Similarly for the x86 architecture. For most programs, the excess precision does only good, but a few programs rely on the precise definition of IEEE floating point. Use **−ffloat−store** for such programs, after modifying them to store all pertinent intermediate computations into variables.

−ffast−math
Sets **−fno−math−errno**, **−funsafe−math−optimizations**, **−fno−trapping−math**, **−ffinite−math−only**, **−fno−rounding−math**, **−fno−signaling−nans** and **fcx-limited-range**.

This option causes the preprocessor macro `__FAST_MATH__` to be defined.

This option should never be turned on by any **−O** option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications for math functions.

−fno−math−errno
Do not set ERRNO after calling math functions that are executed with a single instruction, e.g., sqrt. A program that relies on IEEE exceptions for math error handling may want to use this flag for speed while maintaining IEEE arithmetic compatibility.

This option should never be turned on by any **−O** option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications for math functions.

The default is **−fmath−errno**.

On Darwin systems, the math library never sets `errno`. There is therefore no reason for the compiler to consider the possibility that it might, and **−fno−math−errno** is the default.

−funsafe−math−optimizations
Allow optimizations for floating-point arithmetic that (a) assume that arguments and results are valid and (b) may violate IEEE or ANSI standards. When used at link−time, it may include libraries or startup files that change the default FPU control word or other similar optimizations.

This option should never be turned on by any **−O** option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications for math functions.

The default is **−fno−unsafe−math−optimizations**.

−ffinite−math−only
Allow optimizations for floating-point arithmetic that assume that arguments and results are not NaNs or +−Infs.

This option should never be turned on by any **−O** option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications.

The default is **−fno−finite−math−only**.

−fno−trapping−math
Compile code assuming that floating-point operations cannot generate user-visible traps. These traps include division by zero, overflow, underflow, inexact result and invalid operation. This option implies **−fno−signaling−nans**. Setting this option may allow faster code if one relies on ‘‘non−stop’’ IEEE arithmetic, for example.

This option should never be turned on by any **−O** option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications for math functions.

The default is **−ftrapping−math**.

−frounding−math
Disable transformations and optimizations that assume default floating point rounding behavior. This is round-to-zero for all floating point to integer conversions, and round-to-nearest for all other arithmetic truncations. This option should be specified for programs that change the FP rounding mode dynamically, or that may be executed with a non-default rounding mode. This option disables constant folding of floating point expressions at compile-time (which may be affected by rounding mode) and arithmetic transformations that are unsafe in the presence of sign-dependent rounding modes.

The default is **−fno−rounding−math**.

This option is experimental and does not currently guarantee to disable all GCC optimizations that are affected by rounding mode. Future versions of GCC may provide finer control of this setting using C99’s `FENV_ACCESS` pragma. This command line option will be used to specify the default state for `FENV_ACCESS`.

−fsignaling−nans
Compile code assuming that IEEE signaling NaNs may generate user-visible traps during floating-point operations. Setting this option disables optimizations that may change the number of exceptions visible with signaling NaNs. This option implies **−ftrapping−math**.

This option causes the preprocessor macro `__SUPPORT_SNAN__` to be defined.

The default is **−fno−signaling−nans**.

This option is experimental and does not currently guarantee to disable all GCC optimizations that affect signaling NaN behavior.

−fsingle−precision−constant
Treat floating point constant as single precision constant instead of implicitly converting it to double precision constant.

−fcx−limited−range
−fno−cx−limited−range
When enabled, this option states that a range reduction step is not needed when performing complex division. The default is **−fno−cx−limited−range**, but is enabled by **−ffast−math**.

This option controls the default setting of the ISO C99 `CX_LIMITED_RANGE` pragma. Nevertheless, the option applies to all languages.

The following options control optimizations that may improve performance, but are not enabled by any **−O** options. This section includes experimental options that may produce broken code.

−fbranch−probabilities
After running a program compiled with **−fprofile−arcs**, you can compile it a second time using **−fbranch−probabilities**, to improve optimizations based on the number of times each branch was taken. When the program compiled with **−fprofile−arcs** exits it saves arc execution counts to a file called *sourcename.gcda* for each source file The information in this data file is very dependent on the structure of the generated code, so you must use the same source code and the same optimization options for both compilations.

With **−fbranch−probabilities**, GCC puts a **REG_BR_PROB** note on each **JUMP_INSN** and **CALL_INSN**. These can be used to improve optimization. Currently, they are only used in one place: in *reorg.c*, instead of guessing which path a branch is mostly to take, the **REG_BR_PROB** values are used to exactly determine which path is taken more often.

−fprofile−values
If combined with **−fprofile−arcs**, it adds code so that some data about values of expressions in the program is gathered.

With **−fbranch−probabilities**, it reads back the data gathered from profiling values of expressions and adds **REG_VALUE_PROFILE** notes to instructions for their later usage in optimizations.

Enabled with **−fprofile−generate** and **−fprofile−use**.

−fvpt
If combined with **−fprofile−arcs**, it instructs the compiler to add a code to gather information about values of expressions.

With **−fbranch−probabilities**, it reads back the data gathered and actually performs the optimizations based on them. Currently the optimizations include specialization of division operation using the knowledge about the value of the denominator.

−frename−registers
Attempt to avoid false dependencies in scheduled code by making use of registers left over after register allocation. This optimization will most benefit processors with lots of registers. Depending on the debug information format adopted by the target, however, it can make debugging impossible, since variables will no longer stay in a ‘‘home register’’.

Enabled by default with **−funroll−loops**.

−ftracer
Perform tail duplication to enlarge superblock size. This transformation simplifies the control flow of the function allowing other optimizations to do better job.

Enabled with **−fprofile−use**.

−funroll−loops
Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop. **−funroll−loops** implies **−frerun−cse−after−loop**, **−fweb** and **−frename−registers**. It also turns on complete loop peeling (i.e. complete removal of loops with small constant number of iterations). This option makes code larger, and may or may not make it run faster.

Enabled with **−fprofile−use**.

−funroll−all−loops
Unroll all loops, even if their number of iterations is uncertain when the loop is entered. This usually makes programs run more slowly. **−funroll−all−loops** implies the same options as **−funroll−loops**.

−fpeel−loops
Peels the loops for that there is enough information that they do not roll much (from profile feedback). It also turns on complete loop peeling (i.e. complete removal of loops with small constant number of iterations).

Enabled with **−fprofile−use**.

−fmove−loop−invariants
Enables the loop invariant motion pass in the new loop optimizer. Enabled at level **−O1**

−funswitch−loops
Move branches with loop invariant conditions out of the loop, with duplicates of the loop on both branches (modified according to result of the condition).

−fprefetch−loop−arrays
If supported by the target machine, generate instructions to prefetch memory to improve the performance of loops that access large arrays.

Disabled at level **−Os**.

−ffunction−sections
−fdata−sections
Place each function or data item into its own section in the output file if the target supports arbitrary sections. The name of the function or the name of the data item determines the section's name in the output file.

Use these options on systems where the linker can perform optimizations to improve locality of reference in the instruction space. Most systems using the ELF object format and SPARC processors running Solaris 2 have linkers with such optimizations. AIX may have these optimizations in the future.

Only use these options when there are significant benefits from doing so. When you specify these options, the assembler and linker will create larger object and executable files and will also be slower. You will not be able to use `gprof` on all systems if you specify this option and you may have problems with debugging if you specify both this option and **−g**.

−fbranch−target−load−optimize
Perform branch target register load optimization before prologue / epilogue threading. The use of target registers can typically be exposed only during reload, thus hoisting loads out of loops and doing inter-block scheduling needs a separate optimization pass.

−fbranch−target−load−optimize2
Perform branch target register load optimization after prologue / epilogue threading.

−fbtr−bb−exclusive
When performing branch target register load optimization, don't reuse branch target registers in within any basic block.

−fstack−protector
Emit extra code to check for buffer overflows, such as stack smashing attacks. This is done by adding a guard variable to functions with vulnerable objects. This includes functions that call alloca, and functions with buffers larger than 8 bytes. The guards are initialized when a function is entered and then checked when the function exits. If a guard check fails, an error message is printed and the program exits.

−fstack−protector−all
Like **−fstack−protector** except that all functions are protected.

−−param *name=value*
In some places, GCC uses various constants to control the amount of optimization that is done. For example, GCC will not inline functions that contain more that a certain number of instructions. You can control some of these constants on the command-line using the **−−param** option.

The names of specific parameters, and the meaning of the values, are tied to the internals of the compiler, and are subject to change without notice in future releases.

In each case, the *value* is an integer. The allowable choices for *name* are given in the following table:

salias-max-implicit-fields
The maximum number of fields in a variable without direct structure accesses for which structure aliasing will consider trying to track each field. The default is 5

sra-max-structure-size
The maximum structure size, in bytes, at which the scalar replacement of aggregates (SRA) optimization will perform block copies. The default value, 0, implies that GCC will select the most appropriate size itself.

sra-field-structure-ratio
The threshold ratio (as a percentage) between instantiated fields and the complete structure size. We say that if the ratio of the number of bytes in instantiated fields to the number of bytes in the complete structure exceeds this parameter, then block copies are not used. The default is 75.

max-crossjump-edges
The maximum number of incoming edges to consider for crossjumping. The algorithm used by **−fcrossjumping** is O(N^2) in the number of edges incoming to each block. Increasing values mean more aggressive optimization, making the compile time increase with probably small improvement in executable size.

min-crossjump-insns
The minimum number of instructions which must be matched at the end of two blocks before crossjumping will be performed on them. This value is ignored in the case where all instructions in the block being crossjumped from are matched. The default value is 5.

max-grow-copy-bb-insns
The maximum code size expansion factor when copying basic blocks instead of jumping. The expansion is relative to a jump instruction. The default value is 8.

max-goto-duplication-insns
The maximum number of instructions to duplicate to a block that jumps to a computed goto. To avoid O(N^2) behavior in a number of passes, GCC factors computed gotos early in the compilation process, and unfactors them as late as possible. Only computed jumps at the end of a basic blocks with no more than max-goto-duplication-insns are unfactored. The default value is 8.

max-delay-slot-insn-search
The maximum number of instructions to consider when looking for an instruction to fill a delay slot. If more than this arbitrary number of instructions is searched, the time savings from filling

the delay slot will be minimal so stop searching. Increasing values mean more aggressive optimization, making the compile time increase with probably small improvement in executable run time.

max-delay-slot-live-search
When trying to fill delay slots, the maximum number of instructions to consider when searching for a block with valid live register information. Increasing this arbitrarily chosen value means more aggressive optimization, increasing the compile time. This parameter should be removed when the delay slot code is rewritten to maintain the control-flow graph.

max-gcse-memory
The approximate maximum amount of memory that will be allocated in order to perform the global common subexpression elimination optimization. If more memory than specified is required, the optimization will not be done.

max-gcse-passes
The maximum number of passes of GCSE to run. The default is 1.

max-pending-list-length
The maximum number of pending dependencies scheduling will allow before flushing the current state and starting over. Large functions with few branches or calls can create excessively large lists which needlessly consume memory and resources.

max-inline-insns-single
Several parameters control the tree inliner used in gcc. This number sets the maximum number of instructions (counted in GCC's internal representation) in a single function that the tree inliner will consider for inlining. This only affects functions declared inline and methods implemented in a class declaration (C++). The default value is 450.

max-inline-insns-auto
When you use **−finline−functions** (included in **−O3**), a lot of functions that would otherwise not be considered for inlining by the compiler will be investigated. To those functions, a different (more restrictive) limit compared to functions declared inline can be applied. The default value is 90.

large-function-insns
The limit specifying really large functions. For functions larger than this limit after inlining inlining is constrained by **−−param large-function-growth**. This parameter is useful primarily to avoid extreme compilation time caused by non-linear algorithms used by the backend. This parameter is ignored when **−funit−at−a−time** is not used. The default value is 2700.

large-function-growth
Specifies maximal growth of large function caused by inlining in percents. This parameter is ignored when **−funit−at−a−time** is not used. The default value is 100 which limits large function growth to 2.0 times the original size.

large-unit-insns
The limit specifying large translation unit. Growth caused by inlining of units larger than this limit is limited by **−−param inline-unit-growth**. For small units this might be too tight (consider unit consisting of function A that is inline and B that just calls A three time. If B is small relative to A, the growth of unit is 300\% and yet such inlining is very sane. For very large units consisting of small inlininable functions however the overall unit growth limit is needed to avoid exponential explosion of code size. Thus for smaller units, the size is increased to **−−param large-unit-insns** before aplying **−−param inline-unit-growth**. The default is 10000

inline-unit-growth
Specifies maximal overall growth of the compilation unit caused by inlining. This parameter is ignored when **−funit−at−a−time** is not used. The default value is 50 which limits unit growth to 1.5 times the original size.

max-inline-insns-recursive
max-inline-insns-recursive-auto
Specifies maximum number of instructions out-of-line copy of self recursive inline function can grow into by performing recursive inlining.

For functions declared inline **−−param max-inline-insns-recursive** is taken into acount. For function not declared inline, recursive inlining happens only when **−finline−functions** (included in **−O3**) is enabled and **−−param max-inline-insns-recursive-auto** is used. The default value is 450.

max-inline-recursive-depth
max-inline-recursive-depth-auto
Specifies maximum recursion depth used by the recursive inlining.

For functions declared inline **−−param max-inline-recursive-depth** is taken into acount. For function not declared inline, recursive inlining happens only when **−finline−functions** (included in **−O3**) is enabled and **−−param max-inline-recursive-depth-auto** is used. The default value is 450.

min-inline-recursive-probability
Recursive inlining is profitable only for function having deep recursion in average and can hurt for function having little recursion depth by increasing the prologue size or complexity of function body to other optimizers.

When profile feedback is available (see **−fprofile−generate**) the actual recursion depth can be guessed from probability that function will recurse via given call expression. This parameter limits inlining only to call expression whose probability exceeds given threshold (in percents). The default value is 10.

inline-call-cost
Specify cost of call instruction relative to simple arithmetics operations (having cost of 1). Increasing this cost disqualifies inlining of non-leaf functions and at the same time increases size of leaf function that is believed to reduce function size by being inlined. In effect it increases amount of inlining for code having large abstraction penalty (many functions that just pass the arguments to other functions) and decrease inlining for code with low abstraction penalty. The default value is 16.

max-unrolled-insns
The maximum number of instructions that a loop should have if that loop is unrolled, and if the loop is unrolled, it determines how many times the loop code is unrolled.

max-average-unrolled-insns
The maximum number of instructions biased by probabilities of their execution that a loop should have if that loop is unrolled, and if the loop is unrolled, it determines how many times the loop code is unrolled.

max-unroll-times
The maximum number of unrollings of a single loop.

max-peeled-insns
The maximum number of instructions that a loop should have if that loop is peeled, and if the loop is peeled, it determines how many times the loop code is peeled.

max-peel-times
The maximum number of peelings of a single loop.

max-completely-peeled-insns
The maximum number of insns of a completely peeled loop.

max-completely-peel-times
The maximum number of iterations of a loop to be suitable for complete peeling.

max-unswitch-insns
The maximum number of insns of an unswitched loop.

max-unswitch-level
The maximum number of branches unswitched in a single loop.

lim-expensive
The minimum cost of an expensive expression in the loop invariant motion.

iv-consider-all-candidates-bound
Bound on number of candidates for induction variables below that all candidates are considered for each use in induction variable optimizations. Only the most relevant candidates are considered if there are more candidates, to avoid quadratic time complexity.

iv-max-considered-uses
The induction variable optimizations give up on loops that contain more induction variable uses.

iv-always-prune-cand-set-bound
If number of candidates in the set is smaller than this value, we always try to remove unnecessary ivs from the set during its optimization when a new iv is added to the set.

scev-max-expr-size
Bound on size of expressions used in the scalar evolutions analyzer. Large expressions slow the analyzer.

vect-max-version-checks
The maximum number of runtime checks that can be performed when doing loop versioning in the vectorizer. See option ftree-vect-loop-version for more information.

max-iterations-to-track
The maximum number of iterations of a loop the brute force algorithm for analysis of # of iterations of the loop tries to evaluate.

hot-bb-count-fraction
Select fraction of the maximal count of repetitions of basic block in program given basic block needs to have to be considered hot.

hot-bb-frequency-fraction
Select fraction of the maximal frequency of executions of basic block in function given basic block needs to have to be considered hot

max-predicted-iterations
The maximum number of loop iterations we predict statically. This is useful in cases where function contain single loop with known bound and other loop with unknown. We predict the known number of iterations correctly, while the unknown number of iterations average to roughly 10. This means that the loop without bounds would appear artificially cold relative to the other one.

tracer-dynamic-coverage
tracer-dynamic-coverage-feedback
This value is used to limit superblock formation once the given percentage of executed instructions is covered. This limits unnecessary code size expansion.

The **tracer-dynamic-coverage-feedback** is used only when profile feedback is available. The real profiles (as opposed to statically estimated ones) are much less balanced allowing the threshold to be larger value.

tracer-max-code-growth
Stop tail duplication once code growth has reached given percentage. This is rather hokey argument, as most of the duplicates will be eliminated later in cross jumping, so it may be set to much higher values than is the desired code growth.

tracer-min-branch-ratio
Stop reverse growth when the reverse probability of best edge is less than this threshold (in percent).

tracer-min-branch-ratio
tracer-min-branch-ratio-feedback
Stop forward growth if the best edge do have probability lower than this threshold.

Similarly to **tracer-dynamic-coverage** two values are present, one for compilation for profile feedback and one for compilation without. The value for compilation with profile feedback needs to be more conservative (higher) in order to make tracer effective.

max-cse-path-length
Maximum number of basic blocks on path that cse considers. The default is 10.

max-cse-insns
The maximum instructions CSE process before flushing. The default is 1000.

global-var-threshold
Counts the number of function calls (*n*) and the number of call-clobbered variables (*v*). If *n*x*v* is larger than this limit, a single artificial variable will be created to represent all the call-clobbered variables at function call sites. This artificial variable will then be made to alias every call-clobbered variable. (done as `int * size_t` on the host machine; beware overflow).

max-aliased-vops
Maximum number of virtual operands allowed to represent aliases before triggering the alias grouping heuristic. Alias grouping reduces compile times and memory consumption needed for aliasing at the expense of precision loss in alias information.

ggc-min-expand
GCC uses a garbage collector to manage its own memory allocation. This parameter specifies the minimum percentage by which the garbage collector's heap should be allowed to expand between collections. Tuning this may improve compilation speed; it has no effect on code generation.

The default is 30% + 70% * (RAM/1GB) with an upper bound of 100% when RAM >= 1GB. If `getrlimit` is available, the notion of ''RAM'' is the smallest of actual RAM and `RLIMIT_DATA` or `RLIMIT_AS`. If GCC is not able to calculate RAM on a particular platform, the lower bound of 30% is used. Setting this parameter and **ggc-min-heapsize** to zero causes a full collection to occur at every opportunity. This is extremely slow, but can be useful for debugging.

ggc-min-heapsize
Minimum size of the garbage collector's heap before it begins bothering to collect garbage. The first collection occurs after the heap expands by **ggc-min-expand**% beyond **ggc-min-heapsize**. Again, tuning this may improve compilation speed, and has no effect on code generation.

The default is the smaller of RAM/8, RLIMIT_RSS, or a limit which tries to ensure that RLIMIT_DATA or RLIMIT_AS are not exceeded, but with a lower bound of 4096 (four megabytes) and an upper bound of 131072 (128 megabytes). If GCC is not able to calculate RAM on a particular platform, the lower bound is used. Setting this parameter very large effectively disables garbage collection. Setting this parameter and **ggc-min-expand** to zero causes a full collection to occur at every opportunity.

max-reload-search-insns
The maximum number of instruction reload should look backward for equivalent register. Increasing values mean more aggressive optimization, making the compile time increase with probably slightly better performance. The default value is 100.

max-cselib-memory-location
The maximum number of memory locations cselib should take into acount. Increasing values mean more aggressive optimization, making the compile time increase with probably slightly

better performance. The default value is 500.

max-flow-memory-location
Similar as **max-cselib-memory-location** but for dataflow liveness. The default value is 100.

reorder-blocks-duplicate
reorder-blocks-duplicate-feedback
Used by basic block reordering pass to decide whether to use unconditional branch or duplicate the code on its destination. Code is duplicated when its estimated size is smaller than this value multiplied by the estimated size of unconditional jump in the hot spots of the program.

The **reorder-block-duplicate-feedback** is used only when profile feedback is available and may be set to higher values than **reorder-block-duplicate** since information about the hot spots is more accurate.

max-sched-region-blocks
The maximum number of blocks in a region to be considered for interblock scheduling. The default value is 10.

max-sched-region-insns
The maximum number of insns in a region to be considered for interblock scheduling. The default value is 100.

min-sched-prob
The minimum probability of reaching a source block for interblock speculative scheduling. The default value is 40.

max-last-value-rtl
The maximum size measured as number of RTLs that can be recorded in an expression in combiner for a pseudo register as last known value of that register. The default is 10000.

integer-share-limit
Small integer constants can use a shared data structure, reducing the compiler's memory usage and increasing its speed. This sets the maximum value of a shared integer constant's. The default value is 256.

min-virtual-mappings
Specifies the minimum number of virtual mappings in the incremental SSA updater that should be registered to trigger the virtual mappings heuristic defined by virtual−mappings−ratio. The default value is 100.

virtual-mappings-ratio
If the number of virtual mappings is virtual-mappings-ratio bigger than the number of virtual symbols to be updated, then the incremental SSA updater switches to a full update for those symbols. The default ratio is 3.

ssp-buffer-size
The minimum size of buffers (i.e. arrays) that will receive stack smashing protection when **−fstack−protection** is used.

max-jump-thread-duplication-stmts
Maximum number of statements allowed in a block that needs to be duplicated when threading jumps.

max-fields-for-field-sensitive
Maximum number of fields in a structure we will treat in a field sensitive manner during pointer analysis.

Options Controlling the Preprocessor

These options control the C preprocessor, which is run on each C source file before actual compilation.

If you use the **−E** option, nothing is done except preprocessing. Some of these options make sense only together with **−E** because they cause the preprocessor output to be unsuitable for actual compilation.

You can use **−Wp,***option* to bypass the compiler driver and pass *option* directly through to the preprocessor. If *option* contains commas, it is split into multiple options at the commas. However, many options are modified, translated or interpreted by the compiler driver before being passed to the preprocessor, and **−Wp** forcibly bypasses this phase. The preprocessor's direct interface is undocumented and subject to change, so whenever possible you should avoid using **−Wp** and let the driver handle the options instead.

−Xpreprocessor *option*

Pass *option* as an option to the preprocessor. You can use this to supply system-specific preprocessor options which GCC does not know how to recognize.

If you want to pass an option that takes an argument, you must use **−Xpreprocessor** twice, once for the option and once for the argument.

−D *name*

Predefine *name* as a macro, with definition `1`.

−D *name*=*definition*

The contents of *definition* are tokenized and processed as if they appeared during translation phase three in a **#define** directive. In particular, the definition will be truncated by embedded newline characters.

If you are invoking the preprocessor from a shell or shell-like program you may need to use the shell's quoting syntax to protect characters such as spaces that have a meaning in the shell syntax.

If you wish to define a function-like macro on the command line, write its argument list with surrounding parentheses before the equals sign (if any). Parentheses are meaningful to most shells, so you will need to quote the option. With **sh** and **csh**, **−D'***name***(***args...***)=***definition***'** works.

−D and **−U** options are processed in the order they are given on the command line. All **−imacros** *file* and **−include** *file* options are processed after all **−D** and **−U** options.

−U *name*

Cancel any previous definition of *name*, either built in or provided with a **−D** option.

−undef

Do not predefine any system-specific or GCC-specific macros. The standard predefined macros remain defined.

−I *dir*

Add the directory *dir* to the list of directories to be searched for header files. Directories named by **−I** are searched before the standard system include directories. If the directory *dir* is a standard system include directory, the option is ignored to ensure that the default search order for system directories and the special treatment of system headers are not defeated .

−o *file*

Write output to *file*. This is the same as specifying *file* as the second non-option argument to **cpp**. **gcc** has a different interpretation of a second non-option argument, so you must use **−o** to specify the output file.

−Wall

Turns on all optional warnings which are desirable for normal code. At present this is **−Wcomment**, **−Wtrigraphs**, **−Wmultichar** and a warning about integer promotion causing a change of sign in `#if` expressions. Note that many of the preprocessor's warnings are on by default and have no options to control them.

−Wcomment
−Wcomments
Warn whenever a comment-start sequence **/*** appears in a **/*** comment, or whenever a backslash-newline appears in a // comment. (Both forms have the same effect.)

−Wtrigraphs
@anchor{Wtrigraphs} Most trigraphs in comments cannot affect the meaning of the program. However, a trigraph that would form an escaped newline (**??/** at the end of a line) can, by changing where the comment begins or ends. Therefore, only trigraphs that would form escaped newlines produce warnings inside a comment.

This option is implied by **−Wall**. If **−Wall** is not given, this option is still enabled unless trigraphs are enabled. To get trigraph conversion without warnings, but get the other **−Wall** warnings, use **−trigraphs −Wall −Wno−trigraphs**.

−Wtraditional
Warn about certain constructs that behave differently in traditional and ISO C. Also warn about ISO C constructs that have no traditional C equivalent, and problematic constructs which should be avoided.

−Wimport
Warn the first time **#import** is used.

−Wundef
Warn whenever an identifier which is not a macro is encountered in an **#if** directive, outside of **defined**. Such identifiers are replaced with zero.

−Wunused−macros
Warn about macros defined in the main file that are unused. A macro is *used* if it is expanded or tested for existence at least once. The preprocessor will also warn if the macro has not been used at the time it is redefined or undefined.

Built-in macros, macros defined on the command line, and macros defined in include files are not warned about.

Note: If a macro is actually used, but only used in skipped conditional blocks, then CPP will report it as unused. To avoid the warning in such a case, you might improve the scope of the macro's definition by, for example, moving it into the first skipped block. Alternatively, you could provide a dummy use with something like:

```
#if defined the_macro_causing_the_warning
#endif
```

−Wendif−labels
Warn whenever an **#else** or an **#endif** are followed by text. This usually happens in code of the form

```
#if FOO
...
#else FOO
...
#endif FOO
```

The second and third `FOO` should be in comments, but often are not in older programs. This warning is on by default.

−Werror
Make all warnings into hard errors. Source code which triggers warnings will be rejected.

−Wsystem−headers
Issue warnings for code in system headers. These are normally unhelpful in finding bugs in your own code, therefore suppressed. If you are responsible for the system library, you may want to see them.

−w Suppress all warnings, including those which GNU CPP issues by default.

−pedantic
Issue all the mandatory diagnostics listed in the C standard. Some of them are left out by default, since they trigger frequently on harmless code.

−pedantic−errors
Issue all the mandatory diagnostics, and make all mandatory diagnostics into errors. This includes mandatory diagnostics that GCC issues without **−pedantic** but treats as warnings.

−M
Instead of outputting the result of preprocessing, output a rule suitable for **make** describing the dependencies of the main source file. The preprocessor outputs one **make** rule containing the object file name for that source file, a colon, and the names of all the included files, including those coming from **−include** or **−imacros** command line options.

Unless specified explicitly (with **−MT** or **−MQ**), the object file name consists of the basename of the source file with any suffix replaced with object file suffix. If there are many included files then the rule is split into several lines using \−newline. The rule has no commands.

This option does not suppress the preprocessor's debug output, such as **−dM**. To avoid mixing such debug output with the dependency rules you should explicitly specify the dependency output file with **−MF**, or use an environment variable like **DEPENDENCIES_OUTPUT**. Debug output will still be sent to the regular output stream as normal.

Passing **−M** to the driver implies **−E**, and suppresses warnings with an implicit **−w**.

−MM
Like **−M** but do not mention header files that are found in system header directories, nor header files that are included, directly or indirectly, from such a header.

This implies that the choice of angle brackets or double quotes in an **#include** directive does not in itself determine whether that header will appear in **−MM** dependency output. This is a slight change in semantics from GCC versions 3.0 and earlier.

@anchor{dashMF}

−MF *file*
When used with **−M** or **−MM**, specifies a file to write the dependencies to. If no **−MF** switch is given the preprocessor sends the rules to the same place it would have sent preprocessed output.

When used with the driver options **−MD** or **−MMD**, **−MF** overrides the default dependency output file.

−MG
In conjunction with an option such as **−M** requesting dependency generation, **−MG** assumes missing header files are generated files and adds them to the dependency list without raising an error. The dependency filename is taken directly from the `#include` directive without prepending any path. **−MG** also suppresses preprocessed output, as a missing header file renders this useless.

This feature is used in automatic updating of makefiles.

−MP
This option instructs CPP to add a phony target for each dependency other than the main file, causing each to depend on nothing. These dummy rules work around errors **make** gives if you remove header files without updating the *Makefile* to match.

This is typical output:

```
test.o: test.c test.h

test.h:
```

−MT *target*
Change the target of the rule emitted by dependency generation. By default CPP takes the name of the main input file, including any path, deletes any file suffix such as **.c**, and appends the platform's usual

object suffix. The result is the target.

An **−MT** option will set the target to be exactly the string you specify. If you want multiple targets, you can specify them as a single argument to **−MT**, or use multiple **−MT** options.

For example, **−MT '$(objpfx)foo.o'** might give

```
$(objpfx)foo.o: foo.c
```

−MQ *target*
Same as **−MT**, but it quotes any characters which are special to Make. **−MQ '$(objpfx)foo.o'** gives

```
$$(objpfx)foo.o: foo.c
```

The default target is automatically quoted, as if it were given with **−MQ**.

−MD
−MD is equivalent to **−M −MF** *file*, except that **−E** is not implied. The driver determines *file* based on whether an **−o** option is given. If it is, the driver uses its argument but with a suffix of *.d*, otherwise it take the basename of the input file and applies a *.d* suffix.

If **−MD** is used in conjunction with **−E**, any **−o** switch is understood to specify the dependency output file (but @pxref{dashMF,,−MF}), but if used without **−E**, each **−o** is understood to specify a target object file.

Since **−E** is not implied, **−MD** can be used to generate a dependency output file as a side-effect of the compilation process.

−MMD
Like **−MD** except mention only user header files, not system header files.

−fpch−deps
When using precompiled headers, this flag will cause the dependency-output flags to also list the files from the precompiled header's dependencies. If not specified only the precompiled header would be listed and not the files that were used to create it because those files are not consulted when a precompiled header is used.

−fpch−preprocess
This option allows use of a precompiled header together with **−E**. It inserts a special #pragma, #pragma GCC pch_preprocess "<filename>" in the output to mark the place where the precompiled header was found, and its filename. When **−fpreprocessed** is in use, GCC recognizes this #pragma and loads the PCH.

This option is off by default, because the resulting preprocessed output is only really suitable as input to GCC. It is switched on by **−save−temps**.

You should not write this #pragma in your own code, but it is safe to edit the filename if the PCH file is available in a different location. The filename may be absolute or it may be relative to GCC's current directory.

−x c
−x c++
−x objective-c
−x assembler-with-cpp
Specify the source language: C, C++, Objective−C, or assembly. This has nothing to do with standards conformance or extensions; it merely selects which base syntax to expect. If you give none of these options, cpp will deduce the language from the extension of the source file: **.c**, **.cc**, **.m**, or **.S**. Some other common extensions for C++ and assembly are also recognized. If cpp does not recognize the extension, it will treat the file as C; this is the most generic mode.

Note: Previous versions of cpp accepted a **−lang** option which selected both the language and the standards conformance level. This option has been removed, because it conflicts with the **−l** option.

−std=*standard*
−ansi
Specify the standard to which the code should conform. Currently CPP knows about C and C++ standards; others may be added in the future.

standard may be one of:

`iso9899:1990`
`c89`
The ISO C standard from 1990. **c89** is the customary shorthand for this version of the standard.

The **−ansi** option is equivalent to **−std=c89**.

`iso9899:199409`
The 1990 C standard, as amended in 1994.

`iso9899:1999`
`c99`
`iso9899:199x`
`c9x`
The revised ISO C standard, published in December 1999. Before publication, this was known as C9X.

`gnu89`
The 1990 C standard plus GNU extensions. This is the default.

`gnu99`
`gnu9x`
The 1999 C standard plus GNU extensions.

`c++98`
The 1998 ISO C++ standard plus amendments.

`gnu++98`
The same as **−std=c++98** plus GNU extensions. This is the default for C++ code.

−I−
Split the include path. Any directories specified with **−I** options before **−I−** are searched only for headers requested with `#include "`*file*`"`; they are not searched for `#include <`*file*`>`. If additional directories are specified with **−I** options after the **−I−**, those directories are searched for all **#include** directives.

In addition, **−I−** inhibits the use of the directory of the current file directory as the first search directory for `#include "`*file*`"`. This option has been deprecated.

−nostdinc
Do not search the standard system directories for header files. Only the directories you have specified with **−I** options (and the directory of the current file, if appropriate) are searched.

−nostdinc++
Do not search for header files in the C++−specific standard directories, but do still search the other standard directories. (This option is used when building the C++ library.)

−include *file*
Process *file* as if `#include "file"` appeared as the first line of the primary source file. However, the first directory searched for *file* is the preprocessor's working directory *instead of* the directory containing the main source file. If not found there, it is searched for in the remainder of the `#include "..."` search chain as normal.

If multiple **−include** options are given, the files are included in the order they appear on the command line.

−imacros *file*
Exactly like **−include**, except that any output produced by scanning *file* is thrown away. Macros it defines remain defined. This allows you to acquire all the macros from a header without also processing its declarations.

All files specified by **−imacros** are processed before all files specified by **−include**.

−idirafter *dir*
Search *dir* for header files, but do it *after* all directories specified with **−I** and the standard system directories have been exhausted. *dir* is treated as a system include directory.

−iprefix *prefix*
Specify *prefix* as the prefix for subsequent **−iwithprefix** options. If the prefix represents a directory, you should include the final /.

−iwithprefix *dir*
−iwithprefixbefore *dir*
Append *dir* to the prefix specified previously with **−iprefix**, and add the resulting directory to the include search path. **−iwithprefixbefore** puts it in the same place **−I** would; **−iwithprefix** puts it where **−idirafter** would.

−isysroot *dir*
This option is like the **−−sysroot** option, but applies only to header files. See the **−−sysroot** option for more information.

−isystem *dir*
Search *dir* for header files, after all directories specified by **−I** but before the standard system directories. Mark it as a system directory, so that it gets the same special treatment as is applied to the standard system directories.

−iquote *dir*
Search *dir* only for header files requested with `#include "file"`; they are not searched for `#include <file>`, before all directories specified by **−I** and before the standard system directories.

−fdollars−in−identifiers
@anchor{fdollars−in−identifiers} Accept **$** in identifiers.

−fextended−identifiers
Accept universal character names in identifiers. This option is experimental; in a future version of GCC, it will be enabled by default for C99 and C++.

−fpreprocessed
Indicate to the preprocessor that the input file has already been preprocessed. This suppresses things like macro expansion, trigraph conversion, escaped newline splicing, and processing of most directives. The preprocessor still recognizes and removes comments, so that you can pass a file preprocessed with **−C** to the compiler without problems. In this mode the integrated preprocessor is little more than a tokenizer for the front ends.

−fpreprocessed is implicit if the input file has one of the extensions **.i**, **.ii** or **.mi**. These are the extensions that GCC uses for preprocessed files created by **−save−temps**.

−ftabstop=*width*
Set the distance between tab stops. This helps the preprocessor report correct column numbers in warnings or errors, even if tabs appear on the line. If the value is less than 1 or greater than 100, the option is ignored. The default is 8.

−fexec−charset=*charset*
Set the execution character set, used for string and character constants. The default is UTF−8. *charset* can be any encoding supported by the system's `iconv` library routine.

−fwide−exec−charset=*charset*
Set the wide execution character set, used for wide string and character constants. The default is UTF−32 or UTF−16, whichever corresponds to the width of `wchar_t`. As with **−fexec−charset**, *charset* can be any encoding supported by the system's `iconv` library routine; however, you will have problems with encodings that do not fit exactly in `wchar_t`.

−finput−charset=*charset*
Set the input character set, used for translation from the character set of the input file to the source character set used by GCC. If the locale does not specify, or GCC cannot get this information from the locale, the default is UTF−8. This can be overridden by either the locale or this command line option. Currently the command line option takes precedence if there's a conflict. *charset* can be any encoding supported by the system's `iconv` library routine.

−fworking−directory
Enable generation of linemarkers in the preprocessor output that will let the compiler know the current working directory at the time of preprocessing. When this option is enabled, the preprocessor will emit, after the initial linemarker, a second linemarker with the current working directory followed by two slashes. GCC will use this directory, when it's present in the preprocessed input, as the directory emitted as the current working directory in some debugging information formats. This option is implicitly enabled if debugging information is enabled, but this can be inhibited with the negated form **−fno−working−directory**. If the **−P** flag is present in the command line, this option has no effect, since no `#line` directives are emitted whatsoever.

−fno−show−column
Do not print column numbers in diagnostics. This may be necessary if diagnostics are being scanned by a program that does not understand the column numbers, such as **dejagnu**.

−A *predicate=answer*
Make an assertion with the predicate *predicate* and answer *answer*. This form is preferred to the older form **−A** *predicate***(***answer***)**, which is still supported, because it does not use shell special characters.

−A *−predicate=answer*
Cancel an assertion with the predicate *predicate* and answer *answer*.

−dCHARS
CHARS is a sequence of one or more of the following characters, and must not be preceded by a space. Other characters are interpreted by the compiler proper, or reserved for future versions of GCC, and so are silently ignored. If you specify characters whose behavior conflicts, the result is undefined.

M Instead of the normal output, generate a list of **#define** directives for all the macros defined during the execution of the preprocessor, including predefined macros. This gives you a way of finding out what is predefined in your version of the preprocessor. The following command:

```
cpp -dM < /dev/null | sort
```

will show all the predefined macros and their values. Alternately to be sure the exact same built-in preprocessor code is running you can use this command:

```
gcc -dM -E - < /dev/null | sort
```

D Like **M** except in two respects: it does *not* include the predefined macros, and it outputs *both* the **#define** directives and the result of preprocessing. Both kinds of output go to the standard output file.

N Like **D**, but emit only the macro names, not their expansions.

I Output **#include** directives in addition to the result of preprocessing.

−P Inhibit generation of linemarkers in the output from the preprocessor. This might be useful when running the preprocessor on something that is not C code, and will be sent to a program which might be confused by the linemarkers.

−C Do not discard comments. All comments are passed through to the output file, except for comments in processed directives, which are deleted along with the directive.

You should be prepared for side effects when using **−C**; it causes the preprocessor to treat comments as tokens in their own right. For example, comments appearing at the start of what would be a directive line have the effect of turning that line into an ordinary source line, since the first token on the line is no longer a **#**.

−CC
Do not discard comments, including during macro expansion. This is like **−C**, except that comments contained within macros are also passed through to the output file where the macro is expanded.

In addition to the side-effects of the **−C** option, the **−CC** option causes all C++−style comments inside a macro to be converted to C−style comments. This is to prevent later use of that macro from inadvertently commenting out the remainder of the source line.

The **−CC** option is generally used to support lint comments.

−traditional−cpp
Try to imitate the behavior of old-fashioned C preprocessors, as opposed to ISO C preprocessors.

−trigraphs
Process trigraph sequences. These are three-character sequences, all starting with **??**, that are defined by ISO C to stand for single characters. For example, **??/** stands for \, so **'??/n'** is a character constant for a newline. By default, GCC ignores trigraphs, but in standard-conforming modes it converts them. See the **−std** and **−ansi** options.

The nine trigraphs and their replacements are

```
Trigraph:       ??(  ??)  ??<  ??>  ??=  ??/  ??'  ??!  ??-
Replacement:      [    ]    {    }    #    \    ^    |    ~
```

−remap
Enable special code to work around file systems which only permit very short file names, such as MS−DOS.

−−help
−−target−help
Print text describing all the command line options instead of preprocessing anything.

−v Verbose mode. Print out GNU CPP's version number at the beginning of execution, and report the final form of the include path.

−H Print the name of each header file used, in addition to other normal activities. Each name is indented to show how deep in the **#include** stack it is. Precompiled header files are also printed, even if they are found to be invalid; an invalid precompiled header file is printed with **...x** and a valid one with **...!** .

−version
−−version
Print out GNU CPP's version number. With one dash, proceed to preprocess as normal. With two dashes, exit immediately.

Passing Options to the Assembler

You can pass options to the assembler.

−Wa,*option*
Pass *option* as an option to the assembler. If *option* contains commas, it is split into multiple options at the commas.

−Xassembler *option*
Pass *option* as an option to the assembler. You can use this to supply system-specific assembler options which GCC does not know how to recognize.

If you want to pass an option that takes an argument, you must use **−Xassembler** twice, once for the option and once for the argument.

Options for Linking

These options come into play when the compiler links object files into an executable output file. They are meaningless if the compiler is not doing a link step.

object-file-name
A file name that does not end in a special recognized suffix is considered to name an object file or library. (Object files are distinguished from libraries by the linker according to the file contents.) If linking is done, these object files are used as input to the linker.

−c
−S
−E If any of these options is used, then the linker is not run, and object file names should not be used as arguments.

−l*library*
−l *library*
Search the library named *library* when linking. (The second alternative with the library as a separate argument is only for POSIX compliance and is not recommended.)

It makes a difference where in the command you write this option; the linker searches and processes libraries and object files in the order they are specified. Thus, **foo.o −lz bar.o** searches library **z** after file *foo.o* but before *bar.o*. If *bar.o* refers to functions in **z**, those functions may not be loaded.

The linker searches a standard list of directories for the library, which is actually a file named *liblibrary.a*. The linker then uses this file as if it had been specified precisely by name.

The directories searched include several standard system directories plus any that you specify with **−L**.

Normally the files found this way are library files−−−archive files whose members are object files. The linker handles an archive file by scanning through it for members which define symbols that have so far been referenced but not defined. But if the file that is found is an ordinary object file, it is linked in the usual fashion. The only difference between using an **−l** option and specifying a file name is that **−l** surrounds *library* with **lib** and **.a** and searches several directories.

−lobjc
You need this special case of the **−l** option in order to link an Objective-C or Objective−C++ program.

−nostartfiles
Do not use the standard system startup files when linking. The standard system libraries are used normally, unless **−nostdlib** or **−nodefaultlibs** is used.

−nodefaultlibs
Do not use the standard system libraries when linking. Only the libraries you specify will be passed to the linker. The standard startup files are used normally, unless **−nostartfiles** is used. The compiler may generate calls to `memcmp`, `memset`, `memcpy` and `memmove`. These entries are usually resolved by entries in libc. These entry points should be supplied through some other mechanism when this option is specified.

−nostdlib
Do not use the standard system startup files or libraries when linking. No startup files and only the libraries you specify will be passed to the linker. The compiler may generate calls to `memcmp`, `memset`, `memcpy` and `memmove`. These entries are usually resolved by entries in libc. These entry points should be supplied through some other mechanism when this option is specified.

One of the standard libraries bypassed by **−nostdlib** and **−nodefaultlibs** is *libgcc.a*, a library of internal subroutines that GCC uses to overcome shortcomings of particular machines, or special needs for some languages.

In most cases, you need *libgcc.a* even when you want to avoid other standard libraries. In other words, when you specify **−nostdlib** or **−nodefaultlibs** you should usually specify **−lgcc** as well. This ensures that you have no unresolved references to internal GCC library subroutines. (For example, **__main**, used to ensure C++ constructors will be called.)

−pie
Produce a position independent executable on targets which support it. For predictable results, you must also specify the same set of options that were used to generate code (**−fpie**, **−fPIE**, or model suboptions) when you specify this option.

−rdynamic
Pass the flag **−export−dynamic** to the ELF linker, on targets that support it. This instructs the linker to add all symbols, not only used ones, to the dynamic symbol table. This option is needed for some uses of `dlopen` or to allow obtaining backtraces from within a program.

−s Remove all symbol table and relocation information from the executable.

−static
On systems that support dynamic linking, this prevents linking with the shared libraries. On other systems, this option has no effect.

−shared
Produce a shared object which can then be linked with other objects to form an executable. Not all systems support this option. For predictable results, you must also specify the same set of options that were used to generate code (**−fpic**, **−fPIC**, or model suboptions) when you specify this option.[1]

−shared−libgcc
−static−libgcc
On systems that provide *libgcc* as a shared library, these options force the use of either the shared or static version respectively. If no shared version of *libgcc* was built when the compiler was configured, these options have no effect.

There are several situations in which an application should use the shared *libgcc* instead of the static version. The most common of these is when the application wishes to throw and catch exceptions across different shared libraries. In that case, each of the libraries as well as the application itself should use the shared *libgcc*.

Therefore, the G++ and GCJ drivers automatically add **−shared−libgcc** whenever you build a shared library or a main executable, because C++ and Java programs typically use exceptions, so this is the right thing to do.

If, instead, you use the GCC driver to create shared libraries, you may find that they will not always be linked with the shared *libgcc*. If GCC finds, at its configuration time, that you have a non-GNU linker or a GNU linker that does not support option **−−eh−frame−hdr**, it will link the shared version of *libgcc* into shared libraries by default. Otherwise, it will take advantage of the linker and optimize away the linking with the shared version of *libgcc*, linking with the static version of libgcc by default. This allows exceptions to propagate through such shared libraries, without incurring relocation costs at library load time.

However, if a library or main executable is supposed to throw or catch exceptions, you must link it using the G++ or GCJ driver, as appropriate for the languages used in the program, or using the option **−shared−libgcc**, such that it is linked with the shared *libgcc*.

−symbolic
Bind references to global symbols when building a shared object. Warn about any unresolved references (unless overridden by the link editor option **−Xlinker −z −Xlinker defs**). Only a few systems support this option.

−Xlinker *option*
Pass *option* as an option to the linker. You can use this to supply system-specific linker options which GCC does not know how to recognize.

If you want to pass an option that takes an argument, you must use **−Xlinker** twice, once for the option and once for the argument. For example, to pass **−assert definitions**, you must write **−Xlinker −assert −Xlinker definitions**. It does not work to write **−Xlinker "−assert definitions"**, because this passes the entire string as a single argument, which is not what the linker expects.

−Wl,*option*
Pass *option* as an option to the linker. If *option* contains commas, it is split into multiple options at the commas.

−u *symbol*
Pretend the symbol *symbol* is undefined, to force linking of library modules to define it. You can use **−u** multiple times with different symbols to force loading of additional library modules.

Options for Directory Search

These options specify directories to search for header files, for libraries and for parts of the compiler:

−I*dir*
Add the directory *dir* to the head of the list of directories to be searched for header files. This can be used to override a system header file, substituting your own version, since these directories are searched before the system header file directories. However, you should not use this option to add directories that contain vendor-supplied system header files (use **−isystem** for that). If you use more than one **−I** option, the directories are scanned in left-to-right order; the standard system directories come after.

If a standard system include directory, or a directory specified with **−isystem**, is also specified with **−I**, the **−I** option will be ignored. The directory will still be searched but as a system directory at its normal position in the system include chain. This is to ensure that GCC's procedure to fix buggy system headers and the ordering for the include_next directive are not inadvertently changed. If you really need to change the search order for system directories, use the **−nostdinc** and/or **−isystem** options.

−iquote*dir*
Add the directory *dir* to the head of the list of directories to be searched for header files only for the case of **#include "***file***"**; they are not searched for **#include <***file***>**, otherwise just like **−I**.

−L*dir*
Add directory *dir* to the list of directories to be searched for **−l**.

−B*prefix*
This option specifies where to find the executables, libraries, include files, and data files of the compiler itself.

The compiler driver program runs one or more of the subprograms *cpp*, *cc1*, *as* and *ld*. It tries *prefix* as a prefix for each program it tries to run, both with and without *machine*/*version*/.

For each subprogram to be run, the compiler driver first tries the **−B** prefix, if any. If that name is not found, or if **−B** was not specified, the driver tries two standard prefixes, which are */usr/lib/gcc/* and */usr/local/lib/gcc/*. If neither of those results in a file name that is found, the unmodified program name is searched for using the directories specified in your **PATH** environment variable.

The compiler will check to see if the path provided by the **−B** refers to a directory, and if necessary it will add a directory separator character at the end of the path.

−B prefixes that effectively specify directory names also apply to libraries in the linker, because the compiler translates these options into **−L** options for the linker. They also apply to includes files in the preprocessor, because the compiler translates these options into **−isystem** options for the preprocessor. In this case, the compiler appends **include** to the prefix.

The run-time support file *libgcc.a* can also be searched for using the **−B** prefix, if needed. If it is not found there, the two standard prefixes above are tried, and that is all. The file is left out of the link if it is not found by those means.

Another way to specify a prefix much like the **−B** prefix is to use the environment variable **GCC_EXEC_PREFIX**.

As a special kludge, if the path provided by **−B** is *[dir/]stageN/*, where *N* is a number in the range 0 to 9, then it will be replaced by *[dir/]include*. This is to help with boot-strapping the compiler.

−specs=*file*
Process *file* after the compiler reads in the standard *specs* file, in order to override the defaults that the *gcc* driver program uses when determining what switches to pass to *cc1*, *cc1plus*, *as*, *ld*, etc. More than one **−specs=***file* can be specified on the command line, and they are processed in order, from left to right.

−−sysroot=*dir*
Use *dir* as the logical root directory for headers and libraries. For example, if the compiler would normally search for headers in */usr/include* and libraries in */usr/lib*, it will instead search *dir/usr/include* and *dir/usr/lib*.

If you use both this option and the **−isysroot** option, then the **−−sysroot** option will apply to libraries, but the **−isysroot** option will apply to header files.

The GNU linker (beginning with version 2.16) has the necessary support for this option. If your linker does not support this option, the header file aspect of **−−sysroot** will still work, but the library aspect will not.

−I−
This option has been deprecated. Please use **−iquote** instead for **−I** directories before the **−I−** and remove the **−I−**. Any directories you specify with **−I** options before the **−I−** option are searched only for the case of **#include "***file***"**; they are not searched for **#include <***file***>**.

If additional directories are specified with **−I** options after the **−I−**, these directories are searched for all **#include** directives. (Ordinarily *all* **−I** directories are used this way.)

In addition, the **−I−** option inhibits the use of the current directory (where the current input file came from) as the first search directory for **#include "***file***"**. There is no way to override this effect of **−I−**. With **−I.** you can specify searching the directory which was current when the compiler was invoked. That is not exactly the same as what the preprocessor does by default, but it is often satisfactory.

−I− does not inhibit the use of the standard system directories for header files. Thus, **−I−** and **−nostdinc** are independent.

Specifying Target Machine and Compiler Version

The usual way to run GCC is to run the executable called *gcc*, or *<machine>−gcc* when cross−compiling, or *<machine>−gcc−<version>* to run a version other than the one that was installed last. Sometimes this is inconvenient, so GCC provides options that will switch to another cross-compiler or version.

−b *machine*
The argument *machine* specifies the target machine for compilation.

The value to use for *machine* is the same as was specified as the machine type when configuring GCC as a cross−compiler. For example, if a cross-compiler was configured with **configure arm-elf**, meaning to compile for an arm processor with elf binaries, then you would specify **−b arm-elf** to run that cross compiler. Because there are other options beginning with **−b**, the configuration must contain a hyphen.

−V *version*
The argument *version* specifies which version of GCC to run. This is useful when multiple versions are installed. For example, *version* might be **4.0**, meaning to run GCC version 4.0.

The **−V** and **−b** options work by running the *<machine>−gcc−<version>* executable, so there's no real reason to use them if you can just run that directly.

Hardware Models and Configurations

Earlier we discussed the standard option **−b** which chooses among different installed compilers for completely different target machines, such as VAX vs. 68000 vs. 80386.

In addition, each of these target machine types can have its own special options, starting with **−m**, to choose among various hardware models or configurations−−−for example, 68010 vs 68020, floating coprocessor or none. A single installed version of the compiler can compile for any model or configuration, according to the options specified.

Some configurations of the compiler also support additional special options, usually for compatibility with other compilers on the same platform.

ARC Options

These options are defined for ARC implementations:

−EL
Compile code for little endian mode. This is the default.

−EB
Compile code for big endian mode.

−mmangle−cpu
Prepend the name of the cpu to all public symbol names. In multiple-processor systems, there are many ARC variants with different instruction and register set characteristics. This flag prevents code compiled for one cpu to be linked with code compiled for another. No facility exists for handling variants that are ‘‘almost identical’’. This is an all or nothing option.

−mcpu=*cpu*
Compile code for ARC variant *cpu*. Which variants are supported depend on the configuration. All variants support **−mcpu=base**, this is the default.

−mtext=*text-section*
−mdata=*data-section*
−mrodata=*readonly-data-section*
Put functions, data, and readonly data in *text-section*, *data-section*, and *readonly-data-section* respectively by default. This can be overridden with the `section` attribute.

ARM Options

These **−m** options are defined for Advanced RISC Machines (ARM) architectures:

−mabi=*name*
Generate code for the specified ABI. Permissible values are: **apcs-gnu**, **atpcs**, **aapcs**, **aapcs-linux** and **iwmmxt**.

−mapcs−frame
Generate a stack frame that is compliant with the ARM Procedure Call Standard for all functions, even if this is not strictly necessary for correct execution of the code. Specifying **−fomit−frame−pointer** with this option will cause the stack frames not to be generated for leaf functions. The default is **−mno−apcs−frame**.

−mapcs
This is a synonym for **−mapcs−frame**.

−mthumb−interwork
Generate code which supports calling between the ARM and Thumb instruction sets. Without this option the two instruction sets cannot be reliably used inside one program. The default is **−mno−thumb−interwork**, since slightly larger code is generated when **−mthumb−interwork** is specified.

−mno−sched−prolog
Prevent the reordering of instructions in the function prolog, or the merging of those instruction with the instructions in the function’s body. This means that all functions will start with a recognizable set

of instructions (or in fact one of a choice from a small set of different function prologues), and this information can be used to locate the start if functions inside an executable piece of code. The default is **−msched−prolog**.

−mhard−float
Generate output containing floating point instructions. This is the default.

−msoft−float
Generate output containing library calls for floating point. **Warning:** the requisite libraries are not available for all ARM targets. Normally the facilities of the machine's usual C compiler are used, but this cannot be done directly in cross−compilation. You must make your own arrangements to provide suitable library functions for cross−compilation.

−msoft−float changes the calling convention in the output file; therefore, it is only useful if you compile *all* of a program with this option. In particular, you need to compile *libgcc.a*, the library that comes with GCC, with **−msoft−float** in order for this to work.

−mfloat−abi=*name*
Specifies which ABI to use for floating point values. Permissible values are: **soft**, **softfp** and **hard**.

soft and **hard** are equivalent to **−msoft−float** and **−mhard−float** respectively. **softfp** allows the generation of floating point instructions, but still uses the soft-float calling conventions.

−mlittle−endian
Generate code for a processor running in little-endian mode. This is the default for all standard configurations.

−mbig−endian
Generate code for a processor running in big-endian mode; the default is to compile code for a little-endian processor.

−mwords−little−endian
This option only applies when generating code for big-endian processors. Generate code for a little-endian word order but a big-endian byte order. That is, a byte order of the form **32107654**. Note: this option should only be used if you require compatibility with code for big-endian ARM processors generated by versions of the compiler prior to 2.8.

−mcpu=*name*
This specifies the name of the target ARM processor. GCC uses this name to determine what kind of instructions it can emit when generating assembly code. Permissible names are: **arm2**, **arm250**, **arm3**, **arm6**, **arm60**, **arm600**, **arm610**, **arm620**, **arm7**, **arm7m**, **arm7d**, **arm7dm**, **arm7di**, **arm7dmi**, **arm70**, **arm700**, **arm700i**, **arm710**, **arm710c**, **arm7100**, **arm7500**, **arm7500fe**, **arm7tdmi**, **arm7tdmi−s**, **arm8**, **strongarm**, **strongarm110**, **strongarm1100**, **arm8**, **arm810**, **arm9**, **arm9e**, **arm920**, **arm920t**, **arm922t**, **arm946e−s**, **arm966e−s**, **arm968e−s**, **arm926ej−s**, **arm940t**, **arm9tdmi**, **arm10tdmi**, **arm1020t**, **arm1026ej−s**, **arm10e**, **arm1020e**, **arm1022e**, **arm1136j−s**, **arm1136jf−s**, **mpcore**, **mpcorenovfp**, **arm1176jz−s**, **arm1176jzf−s**, **xscale**, **iwmmxt**, **ep9312**.

−mtune=*name*
This option is very similar to the **−mcpu=** option, except that instead of specifying the actual target processor type, and hence restricting which instructions can be used, it specifies that GCC should tune the performance of the code as if the target were of the type specified in this option, but still choosing the instructions that it will generate based on the cpu specified by a **−mcpu=** option. For some ARM implementations better performance can be obtained by using this option.

−march=*name*
This specifies the name of the target ARM architecture. GCC uses this name to determine what kind of instructions it can emit when generating assembly code. This option can be used in conjunction with or instead of the **−mcpu=** option. Permissible names are: **armv2**, **armv2a**, **armv3**, **armv3m**, **armv4**, **armv4t**, **armv5**, **armv5t**, **armv5te**, **armv6**, **armv6j**, **iwmmxt**, **ep9312**.

−mfpu=*name*
−mfpe=*number*
−mfp=*number*
This specifies what floating point hardware (or hardware emulation) is available on the target. Permissible names are: **fpa**, **fpe2**, **fpe3**, **maverick**, **vfp**. **−mfp** and **−mfpe** are synonyms for **−mfpu**=**fpe***number*, for compatibility with older versions of GCC.

If **−msoft−float** is specified this specifies the format of floating point values.

−mstructure−size−boundary=*n*
The size of all structures and unions will be rounded up to a multiple of the number of bits set by this option. Permissible values are 8, 32 and 64. The default value varies for different toolchains. For the COFF targeted toolchain the default value is 8. A value of 64 is only allowed if the underlying ABI supports it.

Specifying the larger number can produce faster, more efficient code, but can also increase the size of the program. Different values are potentially incompatible. Code compiled with one value cannot necessarily expect to work with code or libraries compiled with another value, if they exchange information using structures or unions.

−mabort−on−noreturn
Generate a call to the function `abort` at the end of a `noreturn` function. It will be executed if the function tries to return.

−mlong−calls
−mno−long−calls
Tells the compiler to perform function calls by first loading the address of the function into a register and then performing a subroutine call on this register. This switch is needed if the target function will lie outside of the 64 megabyte addressing range of the offset based version of subroutine call instruction.

Even if this switch is enabled, not all function calls will be turned into long calls. The heuristic is that static functions, functions which have the **short-call** attribute, functions that are inside the scope of a **#pragma no_long_calls** directive and functions whose definitions have already been compiled within the current compilation unit, will not be turned into long calls. The exception to this rule is that weak function definitions, functions with the **long-call** attribute or the **section** attribute, and functions that are within the scope of a **#pragma long_calls** directive, will always be turned into long calls.

This feature is not enabled by default. Specifying **−mno−long−calls** will restore the default behavior, as will placing the function calls within the scope of a **#pragma long_calls_off** directive. Note these switches have no effect on how the compiler generates code to handle function calls via function pointers.

−mnop−fun−dllimport
Disable support for the `dllimport` attribute.

−msingle−pic−base
Treat the register used for PIC addressing as read−only, rather than loading it in the prologue for each function. The run-time system is responsible for initializing this register with an appropriate value before execution begins.

−mpic−register=*reg*
Specify the register to be used for PIC addressing. The default is R10 unless stack-checking is enabled, when R9 is used.

−mcirrus−fix−invalid−insns
Insert NOPs into the instruction stream to in order to work around problems with invalid Maverick instruction combinations. This option is only valid if the **−mcpu=ep9312** option has been used to enable generation of instructions for the Cirrus Maverick floating point co−processor. This option is not enabled by default, since the problem is only present in older Maverick implementations. The default can be re-enabled by use of the **−mno−cirrus−fix−invalid−insns** switch.

−mpoke−function−name
Write the name of each function into the text section, directly preceding the function prologue. The generated code is similar to this:

```
t0
    .ascii "arm_poke_function_name", 0
    .align
t1
    .word 0xff000000 + (t1 - t0)
arm_poke_function_name
    mov     ip, sp
    stmfd   sp!, {fp, ip, lr, pc}
    sub     fp, ip, #4
```

When performing a stack backtrace, code can inspect the value of `pc` stored at `fp + 0`. If the trace function then looks at location `pc − 12` and the top 8 bits are set, then we know that there is a function name embedded immediately preceding this location and has length `((pc[−3]) & 0xff000000)`.

−mthumb
Generate code for the 16−bit Thumb instruction set. The default is to use the 32−bit ARM instruction set.

−mtpcs−frame
Generate a stack frame that is compliant with the Thumb Procedure Call Standard for all non-leaf functions. (A leaf function is one that does not call any other functions.) The default is **−mno−tpcs−frame**.

−mtpcs−leaf−frame
Generate a stack frame that is compliant with the Thumb Procedure Call Standard for all leaf functions. (A leaf function is one that does not call any other functions.) The default is **−mno−apcs−leaf−frame**.

−mcallee−super−interworking
Gives all externally visible functions in the file being compiled an ARM instruction set header which switches to Thumb mode before executing the rest of the function. This allows these functions to be called from non-interworking code.

−mcaller−super−interworking
Allows calls via function pointers (including virtual functions) to execute correctly regardless of whether the target code has been compiled for interworking or not. There is a small overhead in the cost of executing a function pointer if this option is enabled.

−mtp=*name*
Specify the access model for the thread local storage pointer. The valid models are **soft**, which generates calls to `__aeabi_read_tp`, **cp15**, which fetches the thread pointer from `cp15` directly (supported in the arm6k architecture), and **auto**, which uses the best available method for the selected processor. The default setting is **auto**.

AVR Options

These options are defined for AVR implementations:

−mmcu=*mcu*
Specify ATMEL AVR instruction set or MCU type.

Instruction set avr1 is for the minimal AVR core, not supported by the C compiler, only for assembler programs (MCU types: at90s1200, attiny10, attiny11, attiny12, attiny15, attiny28).

Instruction set avr2 (default) is for the classic AVR core with up to 8K program memory space (MCU types: at90s2313, at90s2323, attiny22, at90s2333, at90s2343, at90s4414, at90s4433, at90s4434, at90s8515, at90c8534, at90s8535).

Instruction set avr3 is for the classic AVR core with up to 128K program memory space (MCU types: atmega103, atmega603, at43usb320, at76c711).

Instruction set avr4 is for the enhanced AVR core with up to 8K program memory space (MCU types: atmega8, atmega83, atmega85).

Instruction set avr5 is for the enhanced AVR core with up to 128K program memory space (MCU types: atmega16, atmega161, atmega163, atmega32, atmega323, atmega64, atmega128, at43usb355, at94k).

−msize
Output instruction sizes to the asm file.

−minit−stack=*N*
Specify the initial stack address, which may be a symbol or numeric value, **__stack** is the default.

−mno−interrupts
Generated code is not compatible with hardware interrupts. Code size will be smaller.

−mcall−prologues
Functions prologues/epilogues expanded as call to appropriate subroutines. Code size will be smaller.

−mno−tablejump
Do not generate tablejump insns which sometimes increase code size.

−mtiny−stack
Change only the low 8 bits of the stack pointer.

−mint8
Assume int to be 8 bit integer. This affects the sizes of all types: A char will be 1 byte, an int will be 1 byte, an long will be 2 bytes and long long will be 4 bytes. Please note that this option does not comply to the C standards, but it will provide you with smaller code size.

Blackfin Options

−momit−leaf−frame−pointer
Don't keep the frame pointer in a register for leaf functions. This avoids the instructions to save, set up and restore frame pointers and makes an extra register available in leaf functions. The option **−fomit−frame−pointer** removes the frame pointer for all functions which might make debugging harder.

−mspecld−anomaly
When enabled, the compiler will ensure that the generated code does not contain speculative loads after jump instructions. This option is enabled by default.

−mno−specld−anomaly
Don't generate extra code to prevent speculative loads from occurring.

−mcsync−anomaly
When enabled, the compiler will ensure that the generated code does not contain CSYNC or SSYNC instructions too soon after conditional branches. This option is enabled by default.

−mno−csync−anomaly
Don't generate extra code to prevent CSYNC or SSYNC instructions from occurring too soon after a conditional branch.

−mlow−64k
When enabled, the compiler is free to take advantage of the knowledge that the entire program fits into the low 64k of memory.

−mno−low−64k
Assume that the program is arbitrarily large. This is the default.

−mid−shared−library
Generate code that supports shared libraries via the library ID method. This allows for execute in place and shared libraries in an environment without virtual memory management. This option implies **−fPIC**.

−mno−id−shared−library
Generate code that doesn't assume ID based shared libraries are being used. This is the default.

−mshared−library−id=n
Specified the identification number of the ID based shared library being compiled. Specifying a value of 0 will generate more compact code, specifying other values will force the allocation of that number to the current library but is no more space or time efficient than omitting this option.

−mlong−calls
−mno−long−calls
Tells the compiler to perform function calls by first loading the address of the function into a register and then performing a subroutine call on this register. This switch is needed if the target function will lie outside of the 24 bit addressing range of the offset based version of subroutine call instruction.

This feature is not enabled by default. Specifying **−mno−long−calls** will restore the default behavior. Note these switches have no effect on how the compiler generates code to handle function calls via function pointers.

CRIS Options

These options are defined specifically for the CRIS ports.

−march=*architecture-type*
−mcpu=*architecture-type*
Generate code for the specified architecture. The choices for *architecture-type* are **v3**, **v8** and **v10** for respectively ETRAX 4, ETRAX 100, and ETRAX 100 LX. Default is **v0** except for cris−axis−linux−gnu, where the default is **v10**.

−mtune=*architecture-type*
Tune to *architecture-type* everything applicable about the generated code, except for the ABI and the set of available instructions. The choices for *architecture-type* are the same as for **−march=***architecture-type*.

−mmax−stack−frame=*n*
Warn when the stack frame of a function exceeds *n* bytes.

−melinux−stacksize=*n*
Only available with the **cris-axis-aout** target. Arranges for indications in the program to the kernel loader that the stack of the program should be set to *n* bytes.

−metrax4
−metrax100
The options **−metrax4** and **−metrax100** are synonyms for **−march=v3** and **−march=v8** respectively.

−mmul−bug−workaround
−mno−mul−bug−workaround
Work around a bug in the `muls` and `mulu` instructions for CPU models where it applies. This option is active by default.

−mpdebug
Enable CRIS-specific verbose debug-related information in the assembly code. This option also has the effect to turn off the **#NO_APP** formatted-code indicator to the assembler at the beginning of the assembly file.

−mcc−init
Do not use condition-code results from previous instruction; always emit compare and test instructions before use of condition codes.

−mno−side−effects
Do not emit instructions with side-effects in addressing modes other than post−increment.

−mstack−align
−mno−stack−align
−mdata−align
−mno−data−align
−mconst−align
−mno−const−align
These options (no−options) arranges (eliminate arrangements) for the stack−frame, individual data and constants to be aligned for the maximum single data access size for the chosen CPU model. The default is to arrange for 32−bit alignment. ABI details such as structure layout are not affected by these options.

−m32−bit
−m16−bit
−m8−bit
Similar to the stack− data− and const-align options above, these options arrange for stack−frame, writable data and constants to all be 32−bit, 16−bit or 8−bit aligned. The default is 32−bit alignment.

−mno−prologue−epilogue
−mprologue−epilogue
With **−mno−prologue−epilogue**, the normal function prologue and epilogue that sets up the stack-frame are omitted and no return instructions or return sequences are generated in the code. Use this option only together with visual inspection of the compiled code: no warnings or errors are generated when call-saved registers must be saved, or storage for local variable needs to be allocated.

−mno−gotplt
−mgotplt
With **−fpic** and **−fPIC**, don't generate (do generate) instruction sequences that load addresses for functions from the PLT part of the GOT rather than (traditional on other architectures) calls to the PLT. The default is **−mgotplt**.

−maout
Legacy no-op option only recognized with the cris-axis-aout target.

−melf
Legacy no-op option only recognized with the cris-axis-elf and cris-axis-linux-gnu targets.

−melinux
Only recognized with the cris-axis-aout target, where it selects a GNU/linux−like multilib, include files and instruction set for **−march=v8**.

−mlinux
Legacy no-op option only recognized with the cris-axis-linux-gnu target.

−sim
This option, recognized for the cris-axis-aout and cris-axis-elf arranges to link with input-output functions from a simulator library. Code, initialized data and zero-initialized data are allocated consecutively.

−sim2
Like **−sim**, but pass linker options to locate initialized data at 0x40000000 and zero-initialized data at 0x80000000.

CRX Options

These options are defined specifically for the CRX ports.

−mmac
Enable the use of multiply-accumulate instructions. Disabled by default.

−mpush−args
Push instructions will be used to pass outgoing arguments when functions are called. Enabled by default.

Darwin Options

These options are defined for all architectures running the Darwin operating system.

FSF GCC on Darwin does not create ‘‘fat’’ object files; it will create an object file for the single architecture that it was built to target. Apple’s GCC on Darwin does create ‘‘fat’’ files if multiple **−arch** options are used; it does so by running the compiler or linker multiple times and joining the results together with *lipo*.

The subtype of the file created (like **ppc7400** or **ppc970** or **i686**) is determined by the flags that specify the ISA that GCC is targetting, like **−mcpu** or **−march**. The **−force_cpusubtype_ALL** option can be used to override this.

The Darwin tools vary in their behavior when presented with an ISA mismatch. The assembler, *as*, will only permit instructions to be used that are valid for the subtype of the file it is generating, so you cannot put 64−bit instructions in an **ppc750** object file. The linker for shared libraries, */usr/bin/libtool*, will fail and print an error if asked to create a shared library with a less restrictive subtype than its input files (for instance, trying to put a **ppc970** object file in a **ppc7400** library). The linker for executables, *ld*, will quietly give the executable the most restrictive subtype of any of its input files.

−F*dir*
Add the framework directory *dir* to the head of the list of directories to be searched for header files. These directories are interleaved with those specified by **−I** options and are scanned in a left-to-right order.

A framework directory is a directory with frameworks in it. A framework is a directory with a **“Headers”** and/or **“PrivateHeaders”** directory contained directly in it that ends in **“.framework”**. The name of a framework is the name of this directory excluding the **“.framework”**. Headers associated with the framework are found in one of those two directories, with **“Headers”** being searched first. A subframework is a framework directory that is in a framework’s **“Frameworks”** directory. Includes of subframework headers can only appear in a header of a framework that contains the subframework, or in a sibling subframework header. Two subframeworks are siblings if they occur in the same framework. A subframework should not have the same name as a framework, a warning will be issued if this is violated. Currently a subframework cannot have subframeworks, in the future, the mechanism may be extended to support this. The standard frameworks can be found in **“/System/Library/Frameworks”** and **“/Library/Frameworks”**. An example include looks like `#include <Framework/header.h>`, where **Framework** denotes the name of the framework and header.h is found in the **“PrivateHeaders”** or **“Headers”** directory.

−gused
Emit debugging information for symbols that are used. For STABS debugging format, this enables **−feliminate−unused−debug−symbols**. This is by default ON.

−gfull
Emit debugging information for all symbols and types.

−mmacosx−version−min=*version*
The earliest version of MacOS X that this executable will run on is *version*. Typical values of *version* include `10.1`, `10.2`, and `10.3.9`.

The default for this option is to make choices that seem to be most useful.

−mone−byte−bool
Override the defaults for **bool** so that **sizeof(bool)==1**. By default **sizeof(bool)** is **4** when compiling for Darwin/PowerPC and **1** when compiling for Darwin/x86, so this option has no effect on x86.

Warning: The **−mone−byte−bool** switch causes GCC to generate code that is not binary compatible with code generated without that switch. Using this switch may require recompiling all other modules in a program, including system libraries. Use this switch to conform to a non-default data model.

−mfix−and−continue
−ffix−and−continue
−findirect−data
Generate code suitable for fast turn around development. Needed to enable gdb to dynamically load `.o` files into already running programs. **−findirect−data** and **−ffix−and−continue** are provided for backwards compatibility.

−all_load
Loads all members of static archive libraries. See man *ld* (1) for more information.

−arch_errors_fatal
Cause the errors having to do with files that have the wrong architecture to be fatal.

−bind_at_load
Causes the output file to be marked such that the dynamic linker will bind all undefined references when the file is loaded or launched.

−bundle
Produce a Mach-o bundle format file. See man *ld* (1) for more information.

−bundle_loader *executable*
This option specifies the *executable* that will be loading the build output file being linked. See man *ld* (1) for more information.

−dynamiclib
When passed this option, GCC will produce a dynamic library instead of an executable when linking, using the Darwin *libtool* command.

−force_cpusubtype_ALL
This causes GCC's output file to have the *ALL* subtype, instead of one controlled by the **−mcpu** or **−march** option.

−allowable_client *client_name*
−client_name
−compatibility_version
−current_version
−dead_strip
−dependency−file
−dylib_file
−dylinker_install_name
−dynamic
−exported_symbols_list
−filelist
−flat_namespace
−force_flat_namespace
−headerpad_max_install_names
−image_base
−init
−install_name
−keep_private_externs
−multi_module
−multiply_defined
−multiply_defined_unused
−noall_load
−no_dead_strip_inits_and_terms
−nofixprebinding
−nomultidefs

−noprebind
−noseglinkedit
−pagezero_size
−prebind
−prebind_all_twolevel_modules
−private_bundle
−read_only_relocs
−sectalign
−sectobjectsymbols
−whyload
−seg1addr
−sectcreate
−sectobjectsymbols
−sectorder
−segaddr
−segs_read_only_addr
−segs_read_write_addr
−seg_addr_table
−seg_addr_table_filename
−seglinkedit
−segprot
−segs_read_only_addr
−segs_read_write_addr
−single_module
−static
−sub_library
−sub_umbrella
−twolevel_namespace
−umbrella
−undefined
−unexported_symbols_list
−weak_reference_mismatches
−whatsloaded

These options are passed to the Darwin linker. The Darwin linker man page describes them in detail.

DEC Alpha Options

These **−m** options are defined for the DEC Alpha implementations:

−mno−soft−float
−msoft−float

Use (do not use) the hardware floating-point instructions for floating-point operations. When **−msoft−float** is specified, functions in *libgcc.a* will be used to perform floating-point operations. Unless they are replaced by routines that emulate the floating-point operations, or compiled in such a way as to call such emulations routines, these routines will issue floating-point operations. If you are compiling for an Alpha without floating-point operations, you must ensure that the library is built so as not to call them.

Note that Alpha implementations without floating-point operations are required to have floating-point registers.

−mfp−reg
−mno−fp−regs

Generate code that uses (does not use) the floating-point register set. **−mno−fp−regs** implies **−msoft−float**. If the floating-point register set is not used, floating point operands are passed in integer registers as if they were integers and floating-point results are passed in `$0` instead of `$f0`. This is a non-standard calling sequence, so any function with a floating-point argument or return value called

by code compiled with **−mno−fp−regs** must also be compiled with that option.

A typical use of this option is building a kernel that does not use, and hence need not save and restore, any floating-point registers.

−mieee

The Alpha architecture implements floating-point hardware optimized for maximum performance. It is mostly compliant with the IEEE floating point standard. However, for full compliance, software assistance is required. This option generates code fully IEEE compliant code *except* that the *inexact-flag* is not maintained (see below). If this option is turned on, the preprocessor macro `_IEEE_FP` is defined during compilation. The resulting code is less efficient but is able to correctly support denormalized numbers and exceptional IEEE values such as not-a-number and plus/minus infinity. Other Alpha compilers call this option **−ieee_with_no_inexact**.

−mieee−with−inexact

This is like **−mieee** except the generated code also maintains the IEEE *inexact-flag*. Turning on this option causes the generated code to implement fully-compliant IEEE math. In addition to `_IEEE_FP`, `_IEEE_FP_EXACT` is defined as a preprocessor macro. On some Alpha implementations the resulting code may execute significantly slower than the code generated by default. Since there is very little code that depends on the *inexact-flag*, you should normally not specify this option. Other Alpha compilers call this option **−ieee_with_inexact**.

−mfp−trap−mode=*trap-mode*

This option controls what floating-point related traps are enabled. Other Alpha compilers call this option **−fptm** *trap-mode*. The trap mode can be set to one of four values:

n This is the default (normal) setting. The only traps that are enabled are the ones that cannot be disabled in software (e.g., division by zero trap).

u In addition to the traps enabled by **n**, underflow traps are enabled as well.

su Like **su**, but the instructions are marked to be safe for software completion (see Alpha architecture manual for details).

sui Like **su**, but inexact traps are enabled as well.

−mfp−rounding−mode=*rounding-mode*

Selects the IEEE rounding mode. Other Alpha compilers call this option **−fprm** *rounding-mode*. The *rounding-mode* can be one of:

n Normal IEEE rounding mode. Floating point numbers are rounded towards the nearest machine number or towards the even machine number in case of a tie.

m Round towards minus infinity.

c Chopped rounding mode. Floating point numbers are rounded towards zero.

d Dynamic rounding mode. A field in the floating point control register (*fpcr*, see Alpha architecture reference manual) controls the rounding mode in effect. The C library initializes this register for rounding towards plus infinity. Thus, unless your program modifies the *fpcr*, **d** corresponds to round towards plus infinity.

−mtrap−precision=*trap-precision*

In the Alpha architecture, floating point traps are imprecise. This means without software assistance it is impossible to recover from a floating trap and program execution normally needs to be terminated. GCC can generate code that can assist operating system trap handlers in determining the exact location that caused a floating point trap. Depending on the requirements of an application, different levels of precisions can be selected:

p Program precision. This option is the default and means a trap handler can only identify which program caused a floating point exception.

f Function precision. The trap handler can determine the function that caused a floating point exception.

i Instruction precision. The trap handler can determine the exact instruction that caused a floating point exception.

Other Alpha compilers provide the equivalent options called **−scope_safe** and **−resumption_safe**.

−mieee−conformant
This option marks the generated code as IEEE conformant. You must not use this option unless you also specify **−mtrap−precision=i** and either **−mfp−trap−mode=su** or **−mfp−trap−mode=sui**. Its only effect is to emit the line **.eflag 48** in the function prologue of the generated assembly file. Under DEC Unix, this has the effect that IEEE-conformant math library routines will be linked in.

−mbuild−constants
Normally GCC examines a 32− or 64−bit integer constant to see if it can construct it from smaller constants in two or three instructions. If it cannot, it will output the constant as a literal and generate code to load it from the data segment at runtime.

Use this option to require GCC to construct *all* integer constants using code, even if it takes more instructions (the maximum is six).

You would typically use this option to build a shared library dynamic loader. Itself a shared library, it must relocate itself in memory before it can find the variables and constants in its own data segment.

−malpha−as
−mgas
Select whether to generate code to be assembled by the vendor-supplied assembler (**−malpha−as**) or by the GNU assembler **−mgas**.

−mbwx
−mno−bwx
−mcix
−mno−cix
−mfix
−mno−fix
−mmax
−mno−max
Indicate whether GCC should generate code to use the optional BWX, CIX, FIX and MAX instruction sets. The default is to use the instruction sets supported by the CPU type specified via **−mcpu=** option or that of the CPU on which GCC was built if none was specified.

−mfloat−vax
−mfloat−ieee
Generate code that uses (does not use) VAX F and G floating point arithmetic instead of IEEE single and double precision.

−mexplicit−relocs
−mno−explicit−relocs
Older Alpha assemblers provided no way to generate symbol relocations except via assembler macros. Use of these macros does not allow optimal instruction scheduling. GNU binutils as of version 2.12 supports a new syntax that allows the compiler to explicitly mark which relocations should apply to which instructions. This option is mostly useful for debugging, as GCC detects the capabilities of the assembler when it is built and sets the default accordingly.

−msmall−data
−mlarge−data
When **−mexplicit−relocs** is in effect, static data is accessed via *gp-relative* relocations. When **−msmall−data** is used, objects 8 bytes long or smaller are placed in a *small data area* (the `.sdata` and `.sbss` sections) and are accessed via 16−bit relocations off of the `$gp` register. This limits the size of the small data area to 64KB, but allows the variables to be directly accessed via a single instruction.

The default is **−mlarge−data**. With this option the data area is limited to just below 2GB. Programs

that require more than 2GB of data must use `malloc` or `mmap` to allocate the data in the heap instead of in the program's data segment.

When generating code for shared libraries, **−fpic** implies **−msmall−data** and **−fPIC** implies **−mlarge−data**.

−msmall−text
−mlarge−text
When **−msmall−text** is used, the compiler assumes that the code of the entire program (or shared library) fits in 4MB, and is thus reachable with a branch instruction. When **−msmall−data** is used, the compiler can assume that all local symbols share the same `$gp` value, and thus reduce the number of instructions required for a function call from 4 to 1.

The default is **−mlarge−text**.

−mcpu=*cpu_type*
Set the instruction set and instruction scheduling parameters for machine type *cpu_type*. You can specify either the **EV** style name or the corresponding chip number. GCC supports scheduling parameters for the EV4, EV5 and EV6 family of processors and will choose the default values for the instruction set from the processor you specify. If you do not specify a processor type, GCC will default to the processor on which the compiler was built.

Supported values for *cpu_type* are

ev4
ev45
21064
Schedules as an EV4 and has no instruction set extensions.

ev5
21164
Schedules as an EV5 and has no instruction set extensions.

ev56
21164a
Schedules as an EV5 and supports the BWX extension.

pca56
21164pc
21164PC
Schedules as an EV5 and supports the BWX and MAX extensions.

ev6
21264
Schedules as an EV6 and supports the BWX, FIX, and MAX extensions.

ev67
21264a
Schedules as an EV6 and supports the BWX, CIX, FIX, and MAX extensions.

−mtune=*cpu_type*
Set only the instruction scheduling parameters for machine type *cpu_type*. The instruction set is not changed.

−mmemory−latency=*time*
Sets the latency the scheduler should assume for typical memory references as seen by the application. This number is highly dependent on the memory access patterns used by the application and the size of the external cache on the machine.

Valid options for *time* are

number
A decimal number representing clock cycles.

L1
L2
L3
main
The compiler contains estimates of the number of clock cycles for ''typical'' EV4 & EV5 hardware for the Level 1, 2 & 3 caches (also called Dcache, Scache, and Bcache), as well as to main memory. Note that L3 is only valid for EV5.

DEC Alpha/VMS Options

These **−m** options are defined for the DEC Alpha/VMS implementations:

−mvms−return−codes
Return VMS condition codes from main. The default is to return POSIX style condition (e.g. error) codes.

FRV Options

−mgpr−32
Only use the first 32 general purpose registers.

−mgpr−64
Use all 64 general purpose registers.

−mfpr−32
Use only the first 32 floating point registers.

−mfpr−64
Use all 64 floating point registers

−mhard−float
Use hardware instructions for floating point operations.

−msoft−float
Use library routines for floating point operations.

−malloc−cc
Dynamically allocate condition code registers.

−mfixed−cc
Do not try to dynamically allocate condition code registers, only use `icc0` and `fcc0`.

−mdword
Change ABI to use double word insns.

−mno−dword
Do not use double word instructions.

−mdouble
Use floating point double instructions.

−mno−double
Do not use floating point double instructions.

−mmedia
Use media instructions.

−mno−media
Do not use media instructions.

−mmuladd
Use multiply and add/subtract instructions.

−mno−muladd
Do not use multiply and add/subtract instructions.

−mfdpic
Select the FDPIC ABI, that uses function descriptors to represent pointers to functions. Without any PIC/PIE−related options, it implies **−fPIE**. With **−fpic** or **−fpie**, it assumes GOT entries and small data are within a 12−bit range from the GOT base address; with **−fPIC** or **−fPIE**, GOT offsets are computed with 32 bits.

−minline−plt
Enable inlining of PLT entries in function calls to functions that are not known to bind locally. It has no effect without **−mfdpic**. It's enabled by default if optimizing for speed and compiling for shared libraries (i.e., **−fPIC** or **−fpic**), or when an optimization option such as **−O3** or above is present in the command line.

−mTLS
Assume a large TLS segment when generating thread-local code.

−mtls
Do not assume a large TLS segment when generating thread-local code.

−mgprel−ro
Enable the use of GPREL relocations in the FDPIC ABI for data that is known to be in read-only sections. It's enabled by default, except for **−fpic** or **−fpie**: even though it may help make the global offset table smaller, it trades 1 instruction for 4. With **−fPIC** or **−fPIE**, it trades 3 instructions for 4, one of which may be shared by multiple symbols, and it avoids the need for a GOT entry for the referenced symbol, so it's more likely to be a win. If it is not, **−mno−gprel−ro** can be used to disable it.

−multilib−library−pic
Link with the (library, not FD) pic libraries. It's implied by **−mlibrary−pic**, as well as by **−fPIC** and **−fpic** without **−mfdpic**. You should never have to use it explicitly.

−mlinked−fp
Follow the EABI requirement of always creating a frame pointer whenever a stack frame is allocated. This option is enabled by default and can be disabled with **−mno−linked−fp**.

−mlong−calls
Use indirect addressing to call functions outside the current compilation unit. This allows the functions to be placed anywhere within the 32−bit address space.

−malign−labels
Try to align labels to an 8−byte boundary by inserting nops into the previous packet. This option only has an effect when VLIW packing is enabled. It doesn't create new packets; it merely adds nops to existing ones.

−mlibrary−pic
Generate position-independent EABI code.

−macc−4
Use only the first four media accumulator registers.

−macc−8
Use all eight media accumulator registers.

−mpack
Pack VLIW instructions.

−mno−pack
Do not pack VLIW instructions.

−mno−eflags
Do not mark ABI switches in e_flags.

−mcond−move
Enable the use of conditional-move instructions (default).

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mno−cond−move
Disable the use of conditional-move instructions.

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mscc
Enable the use of conditional set instructions (default).

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mno−scc
Disable the use of conditional set instructions.

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mcond−exec
Enable the use of conditional execution (default).

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mno−cond−exec
Disable the use of conditional execution.

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mvliw−branch
Run a pass to pack branches into VLIW instructions (default).

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mno−vliw−branch
Do not run a pass to pack branches into VLIW instructions.

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mmulti−cond−exec
Enable optimization of && and || in conditional execution (default).

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mno−multi−cond−exec
Disable optimization of && and || in conditional execution.

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mnested−cond−exec
Enable nested conditional execution optimizations (default).

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−mno−nested−cond−exec
Disable nested conditional execution optimizations.

This switch is mainly for debugging the compiler and will likely be removed in a future version.

−moptimize−membar
This switch removes redundant `membar` instructions from the compiler generated code. It is enabled by default.

−mno−optimize−membar
This switch disables the automatic removal of redundant `membar` instructions from the generated code.

−mtomcat−stats
Cause gas to print out tomcat statistics.

−mcpu=*cpu*
Select the processor type for which to generate code. Possible values are **frv**, **fr550**, **tomcat**, **fr500**, **fr450**, **fr405**, **fr400**, **fr300** and **simple**.

H8/300 Options

These **−m** options are defined for the H8/300 implementations:

−mrelax
Shorten some address references at link time, when possible; uses the linker option **−relax**.

−mh
Generate code for the H8/300H.

−ms
Generate code for the H8S.

−mn
Generate code for the H8S and H8/300H in the normal mode. This switch must be used either with **−mh** or **−ms**.

−ms2600
Generate code for the H8S/2600. This switch must be used with **−ms**.

−mint32
Make `int` data 32 bits by default.

−malign−300
On the H8/300H and H8S, use the same alignment rules as for the H8/300. The default for the H8/300H and H8S is to align longs and floats on 4 byte boundaries. **−malign−300** causes them to be aligned on 2 byte boundaries. This option has no effect on the H8/300.

HPPA Options

These **−m** options are defined for the HPPA family of computers:

−march=*architecture-type*
Generate code for the specified architecture. The choices for *architecture-type* are **1.0** for PA 1.0, **1.1** for PA 1.1, and **2.0** for PA 2.0 processors. Refer to */usr/lib/sched.models* on an HP-UX system to determine the proper architecture option for your machine. Code compiled for lower numbered architectures will run on higher numbered architectures, but not the other way around.

−mpa−risc−1−0
−mpa−risc−1−1
−mpa−risc−2−0
Synonyms for **−march=1.0**, **−march=1.1**, and **−march=2.0** respectively.

−mbig−switch
Generate code suitable for big switch tables. Use this option only if the assembler/linker complain about out of range branches within a switch table.

−mjump−in−delay
Fill delay slots of function calls with unconditional jump instructions by modifying the return pointer for the function call to be the target of the conditional jump.

−mdisable−fpregs
Prevent floating point registers from being used in any manner. This is necessary for compiling kernels which perform lazy context switching of floating point registers. If you use this option and attempt to perform floating point operations, the compiler will abort.

−mdisable−indexing
Prevent the compiler from using indexing address modes. This avoids some rather obscure problems when compiling MIG generated code under MACH.

−mno−space−regs
Generate code that assumes the target has no space registers. This allows GCC to generate faster indirect calls and use unscaled index address modes.

Such code is suitable for level 0 PA systems and kernels.

−mfast−indirect−calls
Generate code that assumes calls never cross space boundaries. This allows GCC to emit code which performs faster indirect calls.

This option will not work in the presence of shared libraries or nested functions.

−mfixed−range=*register-range*
Generate code treating the given register range as fixed registers. A fixed register is one that the register allocator can not use. This is useful when compiling kernel code. A register range is specified as two registers separated by a dash. Multiple register ranges can be specified separated by a comma.

−mlong−load−store
Generate 3−instruction load and store sequences as sometimes required by the HP-UX 10 linker. This is equivalent to the **+k** option to the HP compilers.

−mportable−runtime
Use the portable calling conventions proposed by HP for ELF systems.

−mgas
Enable the use of assembler directives only GAS understands.

−mschedule=*cpu-type*
Schedule code according to the constraints for the machine type *cpu-type*. The choices for *cpu-type* are **700 7100**, **7100LC**, **7200**, **7300** and **8000**. Refer to */usr/lib/sched.models* on an HP-UX system to determine the proper scheduling option for your machine. The default scheduling is **8000**.

−mlinker−opt
Enable the optimization pass in the HP-UX linker. Note this makes symbolic debugging impossible. It also triggers a bug in the HP-UX 8 and HP-UX 9 linkers in which they give bogus error messages when linking some programs.

−msoft−float
Generate output containing library calls for floating point. **Warning:** the requisite libraries are not available for all HPPA targets. Normally the facilities of the machine's usual C compiler are used, but this cannot be done directly in cross−compilation. You must make your own arrangements to provide suitable library functions for cross−compilation. The embedded target **hppa1.1−*−pro** does provide software floating point support.

−msoft−float changes the calling convention in the output file; therefore, it is only useful if you compile *all* of a program with this option. In particular, you need to compile *libgcc.a*, the library that comes with GCC, with **−msoft−float** in order for this to work.

−msio
Generate the predefine, `_SIO`, for server IO. The default is **−mwsio**. This generates the predefines, `__hp9000s700`, `__hp9000s700__` and `_WSIO`, for workstation IO. These options are available under HP-UX and HI−UX.

−mgnu−ld
Use GNU ld specific options. This passes **−shared** to ld when building a shared library. It is the default when GCC is configured, explicitly or implicitly, with the GNU linker. This option does not have any affect on which ld is called, it only changes what parameters are passed to that ld. The ld that is called is determined by the **−−with−ld** configure option, GCC's program search path, and finally by the user's **PATH**. The linker used by GCC can be printed using **which 'gcc**

−print−prog−name=ld‘. This option is only available on the 64 bit HP-UX GCC, i.e. configured with **hppa*64*−*−hpux***.

−mhp−ld

Use HP ld specific options. This passes **−b** to ld when building a shared library and passes **+Accept TypeMismatch** to ld on all links. It is the default when GCC is configured, explicitly or implicitly, with the HP linker. This option does not have any affect on which ld is called, it only changes what parameters are passed to that ld. The ld that is called is determined by the **−−with−ld** configure option, GCC's program search path, and finally by the user's **PATH**. The linker used by GCC can be printed using **which ‘gcc −print−prog−name=ld‘**. This option is only available on the 64 bit HP-UX GCC, i.e. configured with **hppa*64*−*−hpux***.

−mlong−calls

Generate code that uses long call sequences. This ensures that a call is always able to reach linker generated stubs. The default is to generate long calls only when the distance from the call site to the beginning of the function or translation unit, as the case may be, exceeds a predefined limit set by the branch type being used. The limits for normal calls are 7,600,000 and 240,000 bytes, respectively for the PA 2.0 and PA 1.X architectures. Sibcalls are always limited at 240,000 bytes.

Distances are measured from the beginning of functions when using the **−ffunction−sections** option, or when using the **−mgas** and **−mno−portable−runtime** options together under HP-UX with the SOM linker.

It is normally not desirable to use this option as it will degrade performance. However, it may be useful in large applications, particularly when partial linking is used to build the application.

The types of long calls used depends on the capabilities of the assembler and linker, and the type of code being generated. The impact on systems that support long absolute calls, and long pic symbol-difference or pc-relative calls should be relatively small. However, an indirect call is used on 32−bit ELF systems in pic code and it is quite long.

−munix=*unix-std*

Generate compiler predefines and select a startfile for the specified UNIX standard. The choices for *unix-std* are **93**, **95** and **98**. **93** is supported on all HP-UX versions. **95** is available on HP-UX 10.10 and later. **98** is available on HP-UX 11.11 and later. The default values are **93** for HP-UX 10.00, **95** for HP-UX 10.10 though to 11.00, and **98** for HP-UX 11.11 and later.

−munix=93 provides the same predefines as GCC 3.3 and 3.4. **−munix=95** provides additional predefines for `XOPEN_UNIX` and `_XOPEN_SOURCE_EXTENDED`, and the startfile *unix95.o*. **−munix=98** provides additional predefines for `_XOPEN_UNIX`, `_XOPEN_SOURCE_EXTENDED`, `_INCLUDE__STDC_A1_SOURCE` and `_INCLUDE_XOPEN_SOURCE_500`, and the startfile *unix98.o*.

It is *important* to note that this option changes the interfaces for various library routines. It also affects the operational behavior of the C library. Thus, *extreme* care is needed in using this option.

Library code that is intended to operate with more than one UNIX standard must test, set and restore the variable *__xpg4_extended_mask* as appropriate. Most GNU software doesn't provide this capability.

−nolibdld

Suppress the generation of link options to search libdld.sl when the **−static** option is specified on HP-UX 10 and later.

−static

The HP-UX implementation of setlocale in libc has a dependency on libdld.sl. There isn't an archive version of libdld.sl. Thus, when the **−static** option is specified, special link options are needed to resolve this dependency.

On HP-UX 10 and later, the GCC driver adds the necessary options to link with libdld.sl when the **−static** option is specified. This causes the resulting binary to be dynamic. On the 64−bit port, the

linkers generate dynamic binaries by default in any case. The **−nolibdld** option can be used to prevent the GCC driver from adding these link options.

−threads
Add support for multithreading with the *dce thread* library under HP−UX. This option sets flags for both the preprocessor and linker.

Intel 386 and AMD x86−64 Options

These **−m** options are defined for the i386 and x86−64 family of computers:

−mtune=*cpu-type*
Tune to *cpu-type* everything applicable about the generated code, except for the ABI and the set of available instructions. The choices for *cpu-type* are:

i386
Original Intel's i386 CPU.

i486
Intel's i486 CPU. (No scheduling is implemented for this chip.)

i586, pentium
Intel Pentium CPU with no MMX support.

pentium-mmx
Intel PentiumMMX CPU based on Pentium core with MMX instruction set support.

i686, pentiumpro
Intel PentiumPro CPU.

pentium2
Intel Pentium2 CPU based on PentiumPro core with MMX instruction set support.

pentium3, pentium3m
Intel Pentium3 CPU based on PentiumPro core with MMX and SSE instruction set support.

pentium-m
Low power version of Intel Pentium3 CPU with MMX, SSE and SSE2 instruction set support. Used by Centrino notebooks.

pentium4, pentium4m
Intel Pentium4 CPU with MMX, SSE and SSE2 instruction set support.

prescott
Improved version of Intel Pentium4 CPU with MMX, SSE, SSE2 and SSE3 instruction set support.

nocona
Improved version of Intel Pentium4 CPU with 64−bit extensions, MMX, SSE, SSE2 and SSE3 instruction set support.

k6 AMD K6 CPU with MMX instruction set support.

k6−2, k6−3
Improved versions of AMD K6 CPU with MMX and 3dNOW! instruction set support.

athlon, athlon-tbird
AMD Athlon CPU with MMX, 3dNOW!, enhanced 3dNOW! and SSE prefetch instructions support.

athlon−4, athlon−xp, athlon-mp
Improved AMD Athlon CPU with MMX, 3dNOW!, enhanced 3dNOW! and full SSE instruction set support.

k8, opteron, athlon64, athlon-fx
AMD K8 core based CPUs with x86−64 instruction set support. (This supersets MMX, SSE, SSE2, 3dNOW!, enhanced 3dNOW! and 64−bit instruction set extensions.)

winchip−c6
IDT Winchip C6 CPU, dealt in same way as i486 with additional MMX instruction set support.

winchip2
IDT Winchip2 CPU, dealt in same way as i486 with additional MMX and 3dNOW! instruction set support.

c3 Via C3 CPU with MMX and 3dNOW! instruction set support. (No scheduling is implemented for this chip.)

c3−2
Via C3−2 CPU with MMX and SSE instruction set support. (No scheduling is implemented for this chip.)

While picking a specific *cpu-type* will schedule things appropriately for that particular chip, the compiler will not generate any code that does not run on the i386 without the **−march=***cpu-type* option being used.

−march=*cpu-type*
Generate instructions for the machine type *cpu-type*. The choices for *cpu-type* are the same as for **−mtune**. Moreover, specifying **−march=***cpu-type* implies **−mtune=***cpu-type*.

−mcpu=*cpu-type*
A deprecated synonym for **−mtune**.

−m386
−m486
−mpentium
−mpentiumpro
These options are synonyms for **−mtune=i386**, **−mtune=i486**, **−mtune=pentium**, and **−mtune=pentiumpro** respectively. These synonyms are deprecated.

−mfpmath=*unit*
Generate floating point arithmetics for selected unit *unit*. The choices for *unit* are:

387 Use the standard 387 floating point coprocessor present majority of chips and emulated otherwise. Code compiled with this option will run almost everywhere. The temporary results are computed in 80bit precision instead of precision specified by the type resulting in slightly different results compared to most of other chips. See **−ffloat−store** for more detailed description.

This is the default choice for i386 compiler.

sse Use scalar floating point instructions present in the SSE instruction set. This instruction set is supported by Pentium3 and newer chips, in the AMD line by Athlon−4, Athlon-xp and Athlon-mp chips. The earlier version of SSE instruction set supports only single precision arithmetics, thus the double and extended precision arithmetics is still done using 387. Later version, present only in Pentium4 and the future AMD x86−64 chips supports double precision arithmetics too.

For the i386 compiler, you need to use **−march=***cpu-type*, **−msse** or **−msse2** switches to enable SSE extensions and make this option effective. For the x86−64 compiler, these extensions are enabled by default.

The resulting code should be considerably faster in the majority of cases and avoid the numerical instability problems of 387 code, but may break some existing code that expects temporaries to be 80bit.

This is the default choice for the x86−64 compiler.

sse,387
Attempt to utilize both instruction sets at once. This effectively double the amount of available registers and on chips with separate execution units for 387 and SSE the execution resources too. Use this option with care, as it is still experimental, because the GCC register allocator does not model separate functional units well resulting in instable performance.

−masm=*dialect*
Output asm instructions using selected *dialect*. Supported choices are **intel** or **att** (the default one). Darwin does not support **intel**.

−mieee−fp
−mno−ieee−fp
Control whether or not the compiler uses IEEE floating point comparisons. These handle correctly the case where the result of a comparison is unordered.

−msoft−float
Generate output containing library calls for floating point. **Warning:** the requisite libraries are not part of GCC. Normally the facilities of the machine's usual C compiler are used, but this can't be done directly in cross−compilation. You must make your own arrangements to provide suitable library functions for cross−compilation.

On machines where a function returns floating point results in the 80387 register stack, some floating point opcodes may be emitted even if **−msoft−float** is used.

−mno−fp−ret−in−387
Do not use the FPU registers for return values of functions.

The usual calling convention has functions return values of types `float` and `double` in an FPU register, even if there is no FPU. The idea is that the operating system should emulate an FPU.

The option **−mno−fp−ret−in−387** causes such values to be returned in ordinary CPU registers instead.

−mno−fancy−math−387
Some 387 emulators do not support the `sin`, `cos` and `sqrt` instructions for the 387. Specify this option to avoid generating those instructions. This option is the default on FreeBSD, OpenBSD and NetBSD. This option is overridden when **−march** indicates that the target cpu will always have an FPU and so the instruction will not need emulation. As of revision 2.6.1, these instructions are not generated unless you also use the **−funsafe−math−optimizations** switch.

−malign−double
−mno−align−double
Control whether GCC aligns `double`, `long double`, and `long long` variables on a two word boundary or a one word boundary. Aligning `double` variables on a two word boundary will produce code that runs somewhat faster on a **Pentium** at the expense of more memory.

Warning: if you use the **−malign−double** switch, structures containing the above types will be aligned differently than the published application binary interface specifications for the 386 and will not be binary compatible with structures in code compiled without that switch.

−m96bit−long−double
−m128bit−long−double
These switches control the size of `long double` type. The i386 application binary interface specifies the size to be 96 bits, so **−m96bit−long−double** is the default in 32 bit mode.

Modern architectures (Pentium and newer) would prefer `long double` to be aligned to an 8 or 16 byte boundary. In arrays or structures conforming to the ABI, this would not be possible. So specifying a **−m128bit−long−double** will align `long double` to a 16 byte boundary by padding the `long double` with an additional 32 bit zero.

In the x86−64 compiler, **−m128bit−long−double** is the default choice as its ABI specifies that `long double` is to be aligned on 16 byte boundary.

Notice that neither of these options enable any extra precision over the x87 standard of 80 bits for a `long double`.

Warning: if you override the default value for your target ABI, the structures and arrays containing `long double` variables will change their size as well as function calling convention for function taking `long double` will be modified. Hence they will not be binary compatible with arrays or

structures in code compiled without that switch.

−mmlarge−data−threshold=*number*
When **−mcmodel=medium** is specified, the data greater than *threshold* are placed in large data section. This value must be the same across all object linked into the binary and defaults to 65535.

−msvr3−shlib
−mno−svr3−shlib
Control whether GCC places uninitialized local variables into the `bss` or `data` segments. **−msvr3−shlib** places them into `bss`. These options are meaningful only on System V Release 3.

−mrtd
Use a different function-calling convention, in which functions that take a fixed number of arguments return with the `ret` *num* instruction, which pops their arguments while returning. This saves one instruction in the caller since there is no need to pop the arguments there.

You can specify that an individual function is called with this calling sequence with the function attribute **stdcall**. You can also override the **−mrtd** option by using the function attribute **cdecl**.

Warning: this calling convention is incompatible with the one normally used on Unix, so you cannot use it if you need to call libraries compiled with the Unix compiler.

Also, you must provide function prototypes for all functions that take variable numbers of arguments (including `printf`); otherwise incorrect code will be generated for calls to those functions.

In addition, seriously incorrect code will result if you call a function with too many arguments. (Normally, extra arguments are harmlessly ignored.)

−mregparm=*num*
Control how many registers are used to pass integer arguments. By default, no registers are used to pass arguments, and at most 3 registers can be used. You can control this behavior for a specific function by using the function attribute **regparm**.

Warning: if you use this switch, and *num* is nonzero, then you must build all modules with the same value, including any libraries. This includes the system libraries and startup modules.

−msseregparm
Use SSE register passing conventions for float and double arguments and return values. You can control this behavior for a specific function by using the function attribute **sseregparm**.

Warning: if you use this switch then you must build all modules with the same value, including any libraries. This includes the system libraries and startup modules.

−mpreferred−stack−boundary=*num*
Attempt to keep the stack boundary aligned to a 2 raised to *num* byte boundary. If **−mpreferred−stack−boundary** is not specified, the default is 4 (16 bytes or 128 bits), except when optimizing for code size (**−Os**), in which case the default is the minimum correct alignment (4 bytes for x86, and 8 bytes for x86−64).

On Pentium and PentiumPro, `double` and `long double` values should be aligned to an 8 byte boundary (see **−malign−double**) or suffer significant run time performance penalties. On Pentium III, the Streaming SIMD Extension (SSE) data type `__m128` suffers similar penalties if it is not 16 byte aligned.

To ensure proper alignment of this values on the stack, the stack boundary must be as aligned as that required by any value stored on the stack. Further, every function must be generated such that it keeps the stack aligned. Thus calling a function compiled with a higher preferred stack boundary from a function compiled with a lower preferred stack boundary will most likely misalign the stack. It is recommended that libraries that use callbacks always use the default setting.

This extra alignment does consume extra stack space, and generally increases code size. Code that is sensitive to stack space usage, such as embedded systems and operating system kernels, may want to reduce the preferred alignment to **−mpreferred−stack−boundary=2**.

−mmmx
−mno−mmx
−msse
−mno−sse
−msse2
−mno−sse2
−msse3
−mno−sse3
−m3dnow
−mno−3dnow
These switches enable or disable the use of instructions in the MMX, SSE, SSE2 or 3DNow! extended instruction sets. These extensions are also available as built-in functions: see **X86 Built-in Functions**, for details of the functions enabled and disabled by these switches.

To have SSE/SSE2 instructions generated automatically from floating-point code (as opposed to 387 instructions), see **−mfpmath=sse**.

These options will enable GCC to use these extended instructions in generated code, even without **−mfpmath=sse**. Applications which perform runtime CPU detection must compile separate files for each supported architecture, using the appropriate flags. In particular, the file containing the CPU detection code should be compiled without these options.

−mpush−args
−mno−push−args
Use PUSH operations to store outgoing parameters. This method is shorter and usually equally fast as method using SUB/MOV operations and is enabled by default. In some cases disabling it may improve performance because of improved scheduling and reduced dependencies.

−maccumulate−outgoing−args
If enabled, the maximum amount of space required for outgoing arguments will be computed in the function prologue. This is faster on most modern CPUs because of reduced dependencies, improved scheduling and reduced stack usage when preferred stack boundary is not equal to 2. The drawback is a notable increase in code size. This switch implies **−mno−push−args**.

−mthreads
Support thread-safe exception handling on **Mingw32**. Code that relies on thread-safe exception handling must compile and link all code with the **−mthreads** option. When compiling, **−mthreads** defines **−D_MT**; when linking, it links in a special thread helper library **−lmingwthrd** which cleans up per thread exception handling data.

−mno−align−stringops
Do not align destination of inlined string operations. This switch reduces code size and improves performance in case the destination is already aligned, but GCC doesn't know about it.

−minline−all−stringops
By default GCC inlines string operations only when destination is known to be aligned at least to 4 byte boundary. This enables more inlining, increase code size, but may improve performance of code that depends on fast memcpy, strlen and memset for short lengths.

−momit−leaf−frame−pointer
Don't keep the frame pointer in a register for leaf functions. This avoids the instructions to save, set up and restore frame pointers and makes an extra register available in leaf functions. The option **−fomit−frame−pointer** removes the frame pointer for all functions which might make debugging harder.

−mtls−direct−seg−refs
−mno−tls−direct−seg−refs
Controls whether TLS variables may be accessed with offsets from the TLS segment register (`%gs` for 32−bit, `%fs` for 64−bit), or whether the thread base pointer must be added. Whether or not this is legal depends on the operating system, and whether it maps the segment to cover the entire TLS area.

For systems that use GNU libc, the default is on.

These **−m** switches are supported in addition to the above on AMD x86−64 processors in 64−bit environments.

−m32
−m64
Generate code for a 32−bit or 64−bit environment. The 32−bit environment sets int, long and pointer to 32 bits and generates code that runs on any i386 system. The 64−bit environment sets int to 32 bits and long and pointer to 64 bits and generates code for AMD's x86−64 architecture.

−mno−red−zone
Do not use a so called red zone for x86−64 code. The red zone is mandated by the x86−64 ABI, it is a 128−byte area beyond the location of the stack pointer that will not be modified by signal or interrupt handlers and therefore can be used for temporary data without adjusting the stack pointer. The flag **−mno−red−zone** disables this red zone.

−mcmodel=small
Generate code for the small code model: the program and its symbols must be linked in the lower 2 GB of the address space. Pointers are 64 bits. Programs can be statically or dynamically linked. This is the default code model.

−mcmodel=kernel
Generate code for the kernel code model. The kernel runs in the negative 2 GB of the address space. This model has to be used for Linux kernel code.

−mcmodel=medium
Generate code for the medium model: The program is linked in the lower 2 GB of the address space but symbols can be located anywhere in the address space. Programs can be statically or dynamically linked, but building of shared libraries are not supported with the medium model.

−mcmodel=large
Generate code for the large model: This model makes no assumptions about addresses and sizes of sections. Currently GCC does not implement this model.

IA−64 Options

These are the **−m** options defined for the Intel IA−64 architecture.

−mbig−endian
Generate code for a big endian target. This is the default for HP−UX.

−mlittle−endian
Generate code for a little endian target. This is the default for AIX5 and GNU/Linux.

−mgnu−as
−mno−gnu−as
Generate (or don't) code for the GNU assembler. This is the default.

−mgnu−ld
−mno−gnu−ld
Generate (or don't) code for the GNU linker. This is the default.

−mno−pic
Generate code that does not use a global pointer register. The result is not position independent code, and violates the IA−64 ABI.

−mvolatile−asm−stop
−mno−volatile−asm−stop
Generate (or don't) a stop bit immediately before and after volatile asm statements.

−mregister−names

−mno−register−names
Generate (or don't) **in**, **loc**, and **out** register names for the stacked registers. This may make assembler output more readable.

−mno−sdata
−msdata
Disable (or enable) optimizations that use the small data section. This may be useful for working around optimizer bugs.

−mconstant−gp
Generate code that uses a single constant global pointer value. This is useful when compiling kernel code.

−mauto−pic
Generate code that is self−relocatable. This implies **−mconstant−gp**. This is useful when compiling firmware code.

−minline−float−divide−min−latency
Generate code for inline divides of floating point values using the minimum latency algorithm.

−minline−float−divide−max−throughput
Generate code for inline divides of floating point values using the maximum throughput algorithm.

−minline−int−divide−min−latency
Generate code for inline divides of integer values using the minimum latency algorithm.

−minline−int−divide−max−throughput
Generate code for inline divides of integer values using the maximum throughput algorithm.

−minline−sqrt−min−latency
Generate code for inline square roots using the minimum latency algorithm.

−minline−sqrt−max−throughput
Generate code for inline square roots using the maximum throughput algorithm.

−mno−dwarf2−asm
−mdwarf2−asm
Don't (or do) generate assembler code for the DWARF2 line number debugging info. This may be useful when not using the GNU assembler.

−mearly−stop−bits
−mno−early−stop−bits
Allow stop bits to be placed earlier than immediately preceding the instruction that triggered the stop bit. This can improve instruction scheduling, but does not always do so.

−mfixed−range=*register-range*
Generate code treating the given register range as fixed registers. A fixed register is one that the register allocator can not use. This is useful when compiling kernel code. A register range is specified as two registers separated by a dash. Multiple register ranges can be specified separated by a comma.

−mtls−size=*tls-size*
Specify bit size of immediate TLS offsets. Valid values are 14, 22, and 64.

−mtune=*cpu-type*
Tune the instruction scheduling for a particular CPU, Valid values are itanium, itanium1, merced, itanium2, and mckinley.

−mt
−pthread
Add support for multithreading using the POSIX threads library. This option sets flags for both the preprocessor and linker. It does not affect the thread safety of object code produced by the compiler or that of libraries supplied with it. These are HP-UX specific flags.

−milp32
−mlp64
Generate code for a 32−bit or 64−bit environment. The 32−bit environment sets int, long and pointer to 32 bits. The 64−bit environment sets int to 32 bits and long and pointer to 64 bits. These are HP-UX specific flags.

M32C Options

−mcpu=*name*
Select the CPU for which code is generated. *name* may be one of **r8c** for the R8C/Tiny series, **m16c** for the M16C (up to /60) series, **m32cm** for the M16C/80 series, or **m32c** for the M32C/80 series.

−msim
Specifies that the program will be run on the simulator. This causes an alternate runtime library to be linked in which supports, for example, file I/O. You must not use this option when generating programs that will run on real hardware; you must provide your own runtime library for whatever I/O functions are needed.

−memregs=*number*
Specifies the number of memory-based pseudo-registers GCC will use during code generation. These pseudo-registers will be used like real registers, so there is a tradeoff between GCC's ability to fit the code into available registers, and the performance penalty of using memory instead of registers. Note that all modules in a program must be compiled with the same value for this option. Because of that, you must not use this option with the default runtime libraries gcc builds.

M32R/D Options

These **−m** options are defined for Renesas M32R/D architectures:

−m32r2
Generate code for the M32R/2.

−m32rx
Generate code for the M32R/X.

−m32r
Generate code for the M32R. This is the default.

−mmodel=small
Assume all objects live in the lower 16MB of memory (so that their addresses can be loaded with the `ld24` instruction), and assume all subroutines are reachable with the `bl` instruction. This is the default.

The addressability of a particular object can be set with the `model` attribute.

−mmodel=medium
Assume objects may be anywhere in the 32−bit address space (the compiler will generate `seth/add3` instructions to load their addresses), and assume all subroutines are reachable with the `bl` instruction.

−mmodel=large
Assume objects may be anywhere in the 32−bit address space (the compiler will generate `seth/add3` instructions to load their addresses), and assume subroutines may not be reachable with the `bl` instruction (the compiler will generate the much slower `seth/add3/jl` instruction sequence).

−msdata=none
Disable use of the small data area. Variables will be put into one of **.data**, **bss**, or **.rodata** (unless the `section` attribute has been specified). This is the default.

The small data area consists of sections **.sdata** and **.sbss**. Objects may be explicitly put in the small data area with the `section` attribute using one of these sections.

−msdata=sdata
Put small global and static data in the small data area, but do not generate special code to reference them.

−msdata=use
Put small global and static data in the small data area, and generate special instructions to reference them.

−G *num*
Put global and static objects less than or equal to *num* bytes into the small data or bss sections instead of the normal data or bss sections. The default value of *num* is 8. The **−msdata** option must be set to one of **sdata** or **use** for this option to have any effect.

All modules should be compiled with the same **−G** *num* value. Compiling with different values of *num* may or may not work; if it doesn't the linker will give an error message−−−incorrect code will not be generated.

−mdebug
Makes the M32R specific code in the compiler display some statistics that might help in debugging programs.

−malign−loops
Align all loops to a 32−byte boundary.

−mno−align−loops
Do not enforce a 32−byte alignment for loops. This is the default.

−missue−rate=*number*
Issue *number* instructions per cycle. *number* can only be 1 or 2.

−mbranch−cost=*number*
number can only be 1 or 2. If it is 1 then branches will be preferred over conditional code, if it is 2, then the opposite will apply.

−mflush−trap=*number*
Specifies the trap number to use to flush the cache. The default is 12. Valid numbers are between 0 and 15 inclusive.

−mno−flush−trap
Specifies that the cache cannot be flushed by using a trap.

−mflush−func=*name*
Specifies the name of the operating system function to call to flush the cache. The default is *_flush_cache*, but a function call will only be used if a trap is not available.

−mno−flush−func
Indicates that there is no OS function for flushing the cache.

M680x0 Options

These are the **−m** options defined for the 68000 series. The default values for these options depends on which style of 68000 was selected when the compiler was configured; the defaults for the most common choices are given below.

−m68000
−mc68000
Generate output for a 68000. This is the default when the compiler is configured for 68000−based systems.

Use this option for microcontrollers with a 68000 or EC000 core, including the 68008, 68302, 68306, 68307, 68322, 68328 and 68356.

−m68020
−mc68020
Generate output for a 68020. This is the default when the compiler is configured for 68020−based systems.

−m68881
Generate output containing 68881 instructions for floating point. This is the default for most 68020 systems unless **−−nfp** was specified when the compiler was configured.

−m68030
Generate output for a 68030. This is the default when the compiler is configured for 68030−based systems.

−m68040
Generate output for a 68040. This is the default when the compiler is configured for 68040−based systems.

This option inhibits the use of 68881/68882 instructions that have to be emulated by software on the 68040. Use this option if your 68040 does not have code to emulate those instructions.

−m68060
Generate output for a 68060. This is the default when the compiler is configured for 68060−based systems.

This option inhibits the use of 68020 and 68881/68882 instructions that have to be emulated by software on the 68060. Use this option if your 68060 does not have code to emulate those instructions.

−mcpu32
Generate output for a CPU32. This is the default when the compiler is configured for CPU32−based systems.

Use this option for microcontrollers with a CPU32 or CPU32+ core, including the 68330, 68331, 68332, 68333, 68334, 68336, 68340, 68341, 68349 and 68360.

−m5200
Generate output for a 520X ‘‘coldfire’’ family cpu. This is the default when the compiler is configured for 520X−based systems.

Use this option for microcontroller with a 5200 core, including the MCF5202, MCF5203, MCF5204 and MCF5202.

−m68020−40
Generate output for a 68040, without using any of the new instructions. This results in code which can run relatively efficiently on either a 68020/68881 or a 68030 or a 68040. The generated code does use the 68881 instructions that are emulated on the 68040.

−m68020−60
Generate output for a 68060, without using any of the new instructions. This results in code which can run relatively efficiently on either a 68020/68881 or a 68030 or a 68040. The generated code does use the 68881 instructions that are emulated on the 68060.

−msoft−float
Generate output containing library calls for floating point. **Warning:** the requisite libraries are not available for all m68k targets. Normally the facilities of the machine’s usual C compiler are used, but this can’t be done directly in cross−compilation. You must make your own arrangements to provide suitable library functions for cross−compilation. The embedded targets **m68k−*−aout** and **m68k−*−coff** do provide software floating point support.

−mshort
Consider type `int` to be 16 bits wide, like `short int`. Additionally, parameters passed on the stack are also aligned to a 16−bit boundary even on targets whose API mandates promotion to 32−bit.

−mnobitfield
Do not use the bit-field instructions. The **−m68000**, **−mcpu32** and **−m5200** options imply **−mnobitfield**.

−mbitfield
Do use the bit-field instructions. The **−m68020** option implies **−mbitfield**. This is the default if you use a configuration designed for a 68020.

−mrtd
Use a different function-calling convention, in which functions that take a fixed number of arguments return with the `rtd` instruction, which pops their arguments while returning. This saves one instruction in the caller since there is no need to pop the arguments there.

This calling convention is incompatible with the one normally used on Unix, so you cannot use it if you need to call libraries compiled with the Unix compiler.

Also, you must provide function prototypes for all functions that take variable numbers of arguments (including `printf`); otherwise incorrect code will be generated for calls to those functions.

In addition, seriously incorrect code will result if you call a function with too many arguments. (Normally, extra arguments are harmlessly ignored.)

The `rtd` instruction is supported by the 68010, 68020, 68030, 68040, 68060 and CPU32 processors, but not by the 68000 or 5200.

−malign−int
−mno−align−int
Control whether GCC aligns `int`, `long`, `long long`, `float`, `double`, and `long double` variables on a 32−bit boundary (**−malign−int**) or a 16−bit boundary (**−mno−align−int**). Aligning variables on 32−bit boundaries produces code that runs somewhat faster on processors with 32−bit busses at the expense of more memory.

Warning: if you use the **−malign−int** switch, GCC will align structures containing the above types differently than most published application binary interface specifications for the m68k.

−mpcrel
Use the pc-relative addressing mode of the 68000 directly, instead of using a global offset table. At present, this option implies **−fpic**, allowing at most a 16−bit offset for pc-relative addressing. **−fPIC** is not presently supported with **−mpcrel**, though this could be supported for 68020 and higher processors.

−mno−strict−align
−mstrict−align
Do not (do) assume that unaligned memory references will be handled by the system.

−msep−data
Generate code that allows the data segment to be located in a different area of memory from the text segment. This allows for execute in place in an environment without virtual memory management. This option implies **−fPIC**.

−mno−sep−data
Generate code that assumes that the data segment follows the text segment. This is the default.

−mid−shared−library
Generate code that supports shared libraries via the library ID method. This allows for execute in place and shared libraries in an environment without virtual memory management. This option implies **−fPIC**.

−mno−id−shared−library
Generate code that doesn't assume ID based shared libraries are being used. This is the default.

−mshared−library−id=n
Specified the identification number of the ID based shared library being compiled. Specifying a value of 0 will generate more compact code, specifying other values will force the allocation of that number to the current library but is no more space or time efficient than omitting this option.

M68hc1x Options

These are the **−m** options defined for the 68hc11 and 68hc12 microcontrollers. The default values for these options depends on which style of microcontroller was selected when the compiler was configured; the defaults for the most common choices are given below.

−m6811
−m68hc11
Generate output for a 68HC11. This is the default when the compiler is configured for 68HC11−based systems.

−m6812
−m68hc12
Generate output for a 68HC12. This is the default when the compiler is configured for 68HC12−based systems.

−m68S12
−m68hcs12
Generate output for a 68HCS12.

−mauto−incdec
Enable the use of 68HC12 pre and post auto-increment and auto-decrement addressing modes.

−minmax
−nominmax
Enable the use of 68HC12 min and max instructions.

−mlong−calls
−mno−long−calls
Treat all calls as being far away (near). If calls are assumed to be far away, the compiler will use the `call` instruction to call a function and the `rtc` instruction for returning.

−mshort
Consider type `int` to be 16 bits wide, like `short int`.

−msoft−reg−count=*count*
Specify the number of pseudo-soft registers which are used for the code generation. The maximum number is 32. Using more pseudo-soft register may or may not result in better code depending on the program. The default is 4 for 68HC11 and 2 for 68HC12.

MCore Options

These are the **−m** options defined for the Motorola M*Core processors.

−mhardlit
−mno−hardlit
Inline constants into the code stream if it can be done in two instructions or less.

−mdiv
−mno−div
Use the divide instruction. (Enabled by default).

−mrelax−immediate
−mno−relax−immediate
Allow arbitrary sized immediates in bit operations.

−mwide−bitfields

−mno−wide−bitfields
Always treat bit-fields as int−sized.

−m4byte−functions
−mno−4byte−functions
Force all functions to be aligned to a four byte boundary.

−mcallgraph−data
−mno−callgraph−data
Emit callgraph information.

−mslow−bytes
−mno−slow−bytes
Prefer word access when reading byte quantities.

−mlittle−endian
−mbig−endian
Generate code for a little endian target.

−m210
−m340
Generate code for the 210 processor.

MIPS Options

−EB
Generate big-endian code.

−EL
Generate little-endian code. This is the default for **mips*el−*−*** configurations.

−march=*arch*
Generate code that will run on *arch*, which can be the name of a generic MIPS ISA, or the name of a particular processor. The ISA names are: **mips1**, **mips2**, **mips3**, **mips4**, **mips32**, **mips32r2**, and **mips64**. The processor names are: **4kc**, **4km**, **4kp**, **5kc**, **5kf**, **20kc**, **24k**, **24kc**, **24kf**, **24kx**, **m4k**, **orion**, **r2000**, **r3000**, **r3900**, **r4000**, **r4400**, **r4600**, **r4650**, **r6000**, **r8000**, **rm7000**, **rm9000**, **sb1**, **sr71000**, **vr4100**, **vr4111**, **vr4120**, **vr4130**, **vr4300**, **vr5000**, **vr5400** and **vr5500**. The special value **from-abi** selects the most compatible architecture for the selected ABI (that is, **mips1** for 32−bit ABIs and **mips3** for 64−bit ABIs).

In processor names, a final **000** can be abbreviated as **k** (for example, **−march=r2k**). Prefixes are optional, and **vr** may be written **r**.

GCC defines two macros based on the value of this option. The first is **_MIPS_ARCH**, which gives the name of target architecture, as a string. The second has the form **_MIPS_ARCH_***foo*, where *foo* is the capitalized value of **_MIPS_ARCH**. For example, **−march=r2000** will set **_MIPS_ARCH** to **"r2000"** and define the macro **_MIPS_ARCH_R2000**.

Note that the **_MIPS_ARCH** macro uses the processor names given above. In other words, it will have the full prefix and will not abbreviate **000** as **k**. In the case of **from-abi**, the macro names the resolved architecture (either **"mips1"** or **"mips3"**). It names the default architecture when no **−march** option is given.

−mtune=*arch*
Optimize for *arch*. Among other things, this option controls the way instructions are scheduled, and the perceived cost of arithmetic operations. The list of *arch* values is the same as for **−march**.

When this option is not used, GCC will optimize for the processor specified by **−march**. By using **−march** and **−mtune** together, it is possible to generate code that will run on a family of processors, but optimize the code for one particular member of that family.

−mtune defines the macros **_MIPS_TUNE** and **_MIPS_TUNE_***foo*, which work in the same way as the **−march** ones described above.

−mips1
Equivalent to **−march=mips1**.

−mips2
Equivalent to **−march=mips2**.

−mips3
Equivalent to **−march=mips3**.

−mips4
Equivalent to **−march=mips4**.

−mips32
Equivalent to **−march=mips32**.

−mips32r2
Equivalent to **−march=mips32r2**.

−mips64
Equivalent to **−march=mips64**.

−mips16
−mno−mips16
Generate (do not generate) MIPS16 code. If GCC is targetting a MIPS32 or MIPS64 architecture, it will make use of the MIPS16e ASE.

−mabi=32
−mabi=o64
−mabi=n32
−mabi=64
−mabi=eabi
Generate code for the given ABI.

Note that the EABI has a 32−bit and a 64−bit variant. GCC normally generates 64−bit code when you select a 64−bit architecture, but you can use **−mgp32** to get 32−bit code instead.

For information about the O64 ABI, see <**http://gcc.gnu.org/projects/mipso64−abi.html**>.

−mabicalls
−mno−abicalls
Generate (do not generate) SVR4−style position-independent code. **−mabicalls** is the default for SVR4−based systems.

−mxgot
−mno−xgot
Lift (do not lift) the usual restrictions on the size of the global offset table.

GCC normally uses a single instruction to load values from the GOT. While this is relatively efficient, it will only work if the GOT is smaller than about 64k. Anything larger will cause the linker to report an error such as:

```
relocation truncated to fit: R_MIPS_GOT16 foobar
```

If this happens, you should recompile your code with **−mxgot**. It should then work with very large GOTs, although it will also be less efficient, since it will take three instructions to fetch the value of a global symbol.

Note that some linkers can create multiple GOTs. If you have such a linker, you should only need to use **−mxgot** when a single object file accesses more than 64k's worth of GOT entries. Very few do.

These options have no effect unless GCC is generating position independent code.

−mgp32
Assume that general-purpose registers are 32 bits wide.

−mgp64
Assume that general-purpose registers are 64 bits wide.

−mfp32
Assume that floating-point registers are 32 bits wide.

−mfp64
Assume that floating-point registers are 64 bits wide.

−mhard−float
Use floating-point coprocessor instructions.

−msoft−float
Do not use floating-point coprocessor instructions. Implement floating-point calculations using library calls instead.

−msingle−float
Assume that the floating-point coprocessor only supports single-precision operations.

−mdouble−float
Assume that the floating-point coprocessor supports double-precision operations. This is the default.

−mdsp
−mno−dsp
Use (do not use) the MIPS DSP ASE.

−mpaired−single
−mno−paired−single
Use (do not use) paired-single floating-point instructions.
This option can only be used when generating 64−bit code and requires hardware floating-point support to be enabled.

−mips3d
−mno−mips3d
Use (do not use) the MIPS−3D ASE. The option **−mips3d** implies **−mpaired−single**.

−mlong64
Force `long` types to be 64 bits wide. See **−mlong32** for an explanation of the default and the way that the pointer size is determined.

−mlong32
Force `long`, `int`, and pointer types to be 32 bits wide.

The default size of `int`s, `long`s and pointers depends on the ABI. All the supported ABIs use 32−bit `int`s. The n64 ABI uses 64−bit `long`s, as does the 64−bit EABI; the others use 32−bit `long`s. Pointers are the same size as `long`s, or the same size as integer registers, whichever is smaller.

−msym32
−mno−sym32
Assume (do not assume) that all symbols have 32−bit values, regardless of the selected ABI. This option is useful in combination with **−mabi=64** and **−mno−abicalls** because it allows GCC to generate shorter and faster references to symbolic addresses.

−G *num*
Put global and static items less than or equal to *num* bytes into the small data or bss section instead of the normal data or bss section. This allows the data to be accessed using a single instruction.

All modules should be compiled with the same **−G** *num* value.

−membedded−data
−mno−embedded−data
Allocate variables to the read-only data section first if possible, then next in the small data section if possible, otherwise in data. This gives slightly slower code than the default, but reduces the amount of RAM required when executing, and thus may be preferred for some embedded systems.

−muninit−const−in−rodata
−mno−uninit−const−in−rodata
Put uninitialized `const` variables in the read-only data section. This option is only meaningful in conjunction with **−membedded−data**.

−msplit−addresses
−mno−split−addresses
Enable (disable) use of the `%hi()` and `%lo()` assembler relocation operators. This option has been superseded by **−mexplicit−relocs** but is retained for backwards compatibility.

−mexplicit−relocs
−mno−explicit−relocs
Use (do not use) assembler relocation operators when dealing with symbolic addresses. The alternative, selected by **−mno−explicit−relocs**, is to use assembler macros instead.

−mexplicit−relocs is the default if GCC was configured to use an assembler that supports relocation operators.

−mcheck−zero−division
−mno−check−zero−division
Trap (do not trap) on integer division by zero. The default is **−mcheck−zero−division**.

−mdivide−traps
−mdivide−breaks
MIPS systems check for division by zero by generating either a conditional trap or a break instruction. Using traps results in smaller code, but is only supported on MIPS II and later. Also, some versions of the Linux kernel have a bug that prevents trap from generating the proper signal (`SIGFPE`). Use **−mdivide−traps** to allow conditional traps on architectures that support them and **−mdivide−breaks** to force the use of breaks.

The default is usually **−mdivide−traps**, but this can be overridden at configure time using **−−with−divide=breaks**. Divide-by-zero checks can be completely disabled using **−mno−check−zero−division**.

−mmemcpy
−mno−memcpy
Force (do not force) the use of `memcpy()` for non-trivial block moves. The default is **−mno−memcpy**, which allows GCC to inline most constant-sized copies.

−mlong−calls
−mno−long−calls
Disable (do not disable) use of the `jal` instruction. Calling functions using `jal` is more efficient but requires the caller and callee to be in the same 256 megabyte segment.

This option has no effect on abicalls code. The default is **−mno−long−calls**.

−mmad
−mno−mad
Enable (disable) use of the `mad`, `madu` and `mul` instructions, as provided by the R4650 ISA.

−mfused−madd
−mno−fused−madd
Enable (disable) use of the floating point multiply-accumulate instructions, when they are available. The default is **−mfused−madd**.

When multiply-accumulate instructions are used, the intermediate product is calculated to infinite precision and is not subject to the FCSR Flush to Zero bit. This may be undesirable in some circumstances.

−nocpp
Tell the MIPS assembler to not run its preprocessor over user assembler files (with a **.s** suffix) when assembling them.

−mfix−r4000
−mno−fix−r4000
Work around certain R4000 CPU errata:

- A double-word or a variable shift may give an incorrect result if executed immediately after starting an integer division.
- A double-word or a variable shift may give an incorrect result if executed while an integer multiplication is in progress.
- An integer division may give an incorrect result if started in a delay slot of a taken branch or a jump.

−mfix−r4400
−mno−fix−r4400
Work around certain R4400 CPU errata:

- A double-word or a variable shift may give an incorrect result if executed immediately after starting an integer division.

−mfix−vr4120
−mno−fix−vr4120
Work around certain VR4120 errata:

- `dmultu` does not always produce the correct result.
- `div` and `ddiv` do not always produce the correct result if one of the operands is negative.

The workarounds for the division errata rely on special functions in *libgcc.a*. At present, these functions are only provided by the `mips64vr*−elf` configurations.

Other VR4120 errata require a nop to be inserted between certain pairs of instructions. These errata are handled by the assembler, not by GCC itself.

−mfix−vr4130
Work around the VR4130 `mflo`/`mfhi` errata. The workarounds are implemented by the assembler rather than by GCC, although GCC will avoid using `mflo` and `mfhi` if the VR4130 `macc`, `macchi`, `dmacc` and `dmacchi` instructions are available instead.

−mfix−sb1
−mno−fix−sb1
Work around certain SB−1 CPU core errata. (This flag currently works around the SB−1 revision 2 ‘‘F1’’ and ‘‘F2’’ floating point errata.)

−mflush−func=*func*
−mno−flush−func
Specifies the function to call to flush the I and D caches, or to not call any such function. If called, the function must take the same arguments as the common `_flush_func()`, that is, the address of the memory range for which the cache is being flushed, the size of the memory range, and the number 3 (to flush both caches). The default depends on the target GCC was configured for, but commonly is either **_flush_func** or **__cpu_flush**.

−mbranch−likely
−mno−branch−likely
Enable or disable use of Branch Likely instructions, regardless of the default for the selected architecture. By default, Branch Likely instructions may be generated if they are supported by the selected architecture. An exception is for the MIPS32 and MIPS64 architectures and processors which implement those architectures; for those, Branch Likely instructions will not be generated by default because the MIPS32 and MIPS64 architectures specifically deprecate their use.

−mfp−exceptions

−mno−fp−exceptions
Specifies whether FP exceptions are enabled. This affects how we schedule FP instructions for some processors. The default is that FP exceptions are enabled.

For instance, on the SB−1, if FP exceptions are disabled, and we are emitting 64−bit code, then we can use both FP pipes. Otherwise, we can only use one FP pipe.

−mvr4130−align
−mno−vr4130−align
The VR4130 pipeline is two-way superscalar, but can only issue two instructions together if the first one is 8−byte aligned. When this option is enabled, GCC will align pairs of instructions that it thinks should execute in parallel.

This option only has an effect when optimizing for the VR4130. It normally makes code faster, but at the expense of making it bigger. It is enabled by default at optimization level **−O3**.

MMIX Options

These options are defined for the MMIX:

−mlibfuncs
−mno−libfuncs
Specify that intrinsic library functions are being compiled, passing all values in registers, no matter the size.

−mepsilon
−mno−epsilon
Generate floating-point comparison instructions that compare with respect to the `rE` epsilon register.

−mabi=mmixware
−mabi=gnu
Generate code that passes function parameters and return values that (in the called function) are seen as registers `$0` and up, as opposed to the GNU ABI which uses global registers `$231` and up.

−mzero−extend
−mno−zero−extend
When reading data from memory in sizes shorter than 64 bits, use (do not use) zero-extending load instructions by default, rather than sign-extending ones.

−mknuthdiv
−mno−knuthdiv
Make the result of a division yielding a remainder have the same sign as the divisor. With the default, **−mno−knuthdiv**, the sign of the remainder follows the sign of the dividend. Both methods are arithmetically valid, the latter being almost exclusively used.

−mtoplevel−symbols
−mno−toplevel−symbols
Prepend (do not prepend) a `:` to all global symbols, so the assembly code can be used with the `PREFIX` assembly directive.

−melf
Generate an executable in the ELF format, rather than the default **mmo** format used by the **mmix** simulator.

−mbranch−predict
−mno−branch−predict
Use (do not use) the probable-branch instructions, when static branch prediction indicates a probable branch.

−mbase−addresses
−mno−base−addresses
Generate (do not generate) code that uses *base addresses*. Using a base address automatically generates a request (handled by the assembler and the linker) for a constant to be set up in a global register.

The register is used for one or more base address requests within the range 0 to 255 from the value held in the register. The generally leads to short and fast code, but the number of different data items that can be addressed is limited. This means that a program that uses lots of static data may require **−mno−base−addresses**.

−msingle−exit
−mno−single−exit
Force (do not force) generated code to have a single exit point in each function.

MN10300 Options

These **−m** options are defined for Matsushita MN10300 architectures:

−mmult−bug
Generate code to avoid bugs in the multiply instructions for the MN10300 processors. This is the default.

−mno−mult−bug
Do not generate code to avoid bugs in the multiply instructions for the MN10300 processors.

−mam33
Generate code which uses features specific to the AM33 processor.

−mno−am33
Do not generate code which uses features specific to the AM33 processor. This is the default.

−mreturn−pointer−on−d0
When generating a function which returns a pointer, return the pointer in both `a0` and `d0`. Otherwise, the pointer is returned only in a0, and attempts to call such functions without a prototype would result in errors. Note that this option is on by default; use **−mno−return−pointer−on−d0** to disable it.

−mno−crt0
Do not link in the C run-time initialization object file.

−mrelax
Indicate to the linker that it should perform a relaxation optimization pass to shorten branches, calls and absolute memory addresses. This option only has an effect when used on the command line for the final link step.

This option makes symbolic debugging impossible.

MT Options

These **−m** options are defined for Morpho MT architectures:

−march=*cpu-type*
Generate code that will run on *cpu-type*, which is the name of a system representing a certain processor type. Possible values for *cpu-type* are **ms1−64−001**, **ms1−16−002**, **ms1−16−003** and **ms2**.

When this option is not used, the default is **−march=ms1−16−002**.

−mbacc
Use byte loads and stores when generating code.

−mno−bacc
Do not use byte loads and stores when generating code.

−msim
Use simulator runtime

−mno−crt0
Do not link in the C run-time initialization object file *crti.o*. Other run-time initialization and termination files such as *startup.o* and *exit.o* are still included on the linker command line.

PDP−11 Options

These options are defined for the PDP−11:

−mfpu
Use hardware FPP floating point. This is the default. (FIS floating point on the PDP−11/40 is not supported.)

−msoft−float
Do not use hardware floating point.

−mac0
Return floating-point results in ac0 (fr0 in Unix assembler syntax).

−mno−ac0
Return floating-point results in memory. This is the default.

−m40
Generate code for a PDP−11/40.

−m45
Generate code for a PDP−11/45. This is the default.

−m10
Generate code for a PDP−11/10.

−mbcopy−builtin
Use inline `movmemhi` patterns for copying memory. This is the default.

−mbcopy
Do not use inline `movmemhi` patterns for copying memory.

−mint16
−mno−int32
Use 16−bit `int`. This is the default.

−mint32
−mno−int16
Use 32−bit `int`.

−mfloat64
−mno−float32
Use 64−bit `float`. This is the default.

−mfloat32
−mno−float64
Use 32−bit `float`.

−mabshi
Use `abshi2` pattern. This is the default.

−mno−abshi
Do not use `abshi2` pattern.

−mbranch−expensive
Pretend that branches are expensive. This is for experimenting with code generation only.

−mbranch−cheap
Do not pretend that branches are expensive. This is the default.

−msplit
Generate code for a system with split I&D.

−mno−split
Generate code for a system without split I&D. This is the default.

−munix−asm
Use Unix assembler syntax. This is the default when configured for **pdp11−*−bsd**.

−mdec−asm
Use DEC assembler syntax. This is the default when configured for any PDP−11 target other than **pdp11−*−bsd**.

PowerPC Options

These are listed under

IBM RS/6000 and PowerPC Options

These **−m** options are defined for the IBM RS/6000 and PowerPC:

−mpower
−mno−power
−mpower2
−mno−power2
−mpowerpc
−mno−powerpc
−mpowerpc−gpopt
−mno−powerpc−gpopt
−mpowerpc−gfxopt
−mno−powerpc−gfxopt
−mpowerpc64
−mno−powerpc64
−mmfcrf
−mno−mfcrf
−mpopcntb
−mno−popcntb
−mfprnd
−mno−fprnd
GCC supports two related instruction set architectures for the RS/6000 and PowerPC. The *POWER* instruction set are those instructions supported by the **rios** chip set used in the original RS/6000 systems and the *PowerPC* instruction set is the architecture of the Freescale MPC5xx, MPC6xx, MPC8xx microprocessors, and the IBM 4xx, 6xx, and follow-on microprocessors.

Neither architecture is a subset of the other. However there is a large common subset of instructions supported by both. An MQ register is included in processors supporting the POWER architecture.

You use these options to specify which instructions are available on the processor you are using. The default value of these options is determined when configuring GCC. Specifying the **−mcpu=***cpu_type* overrides the specification of these options. We recommend you use the **−mcpu=***cpu_type* option rather than the options listed above.

The **−mpower** option allows GCC to generate instructions that are found only in the POWER architecture and to use the MQ register. Specifying **−mpower2** implies **−power** and also allows GCC to generate instructions that are present in the POWER2 architecture but not the original POWER architecture.

The **−mpowerpc** option allows GCC to generate instructions that are found only in the 32−bit subset of the PowerPC architecture. Specifying **−mpowerpc−gpopt** implies **−mpowerpc** and also allows GCC to use the optional PowerPC architecture instructions in the General Purpose group, including floating-point square root. Specifying **−mpowerpc−gfxopt** implies **−mpowerpc** and also allows GCC to use the optional PowerPC architecture instructions in the Graphics group, including floating-point select.

The **−mmfcrf** option allows GCC to generate the move from condition register field instruction implemented on the POWER4 processor and other processors that support the PowerPC V2.01 architecture. The **−mpopcntb** option allows GCC to generate the popcount and double precision FP reciprocal estimate instruction implemented on the POWER5 processor and other processors that support the PowerPC V2.02 architecture. The **−mfprnd** option allows GCC to generate the FP round to integer instructions implemented on the POWER5+ processor and other processors that support the PowerPC V2.03

architecture.

The **−mpowerpc64** option allows GCC to generate the additional 64−bit instructions that are found in the full PowerPC64 architecture and to treat GPRs as 64−bit, doubleword quantities. GCC defaults to **−mno−powerpc64**.

If you specify both **−mno−power** and **−mno−powerpc**, GCC will use only the instructions in the common subset of both architectures plus some special AIX common-mode calls, and will not use the MQ register. Specifying both **−mpower** and **−mpowerpc** permits GCC to use any instruction from either architecture and to allow use of the MQ register; specify this for the Motorola MPC601.

−mnew−mnemonics
−mold−mnemonics
Select which mnemonics to use in the generated assembler code. With **−mnew−mnemonics**, GCC uses the assembler mnemonics defined for the PowerPC architecture. With **−mold−mnemonics** it uses the assembler mnemonics defined for the POWER architecture. Instructions defined in only one architecture have only one mnemonic; GCC uses that mnemonic irrespective of which of these options is specified.

GCC defaults to the mnemonics appropriate for the architecture in use. Specifying **−mcpu=***cpu_type* sometimes overrides the value of these option. Unless you are building a cross−compiler, you should normally not specify either **−mnew−mnemonics** or **−mold−mnemonics**, but should instead accept the default.

−mcpu=*cpu_type*
Set architecture type, register usage, choice of mnemonics, and instruction scheduling parameters for machine type *cpu_type*. Supported values for *cpu_type* are **401**, **403**, **405**, **405fp**, **440**, **440fp**, **505**, **601**, **602**, **603**, **603e**, **604**, **604e**, **620**, **630**, **740**, **7400**, **7450**, **750**, **801**, **821**, **823**, **860**, **970**, **8540**, **ec603e**, **G3**, **G4**, **G5**, **power**, **power2**, **power3**, **power4**, **power5**, **power5+**, **common**, **powerpc**, **powerpc64**, **rios**, **rios1**, **rios2**, **rsc**, and **rs64**.

−mcpu=common selects a completely generic processor. Code generated under this option will run on any POWER or PowerPC processor. GCC will use only the instructions in the common subset of both architectures, and will not use the MQ register. GCC assumes a generic processor model for scheduling purposes.

−mcpu=power, **−mcpu=power2**, **−mcpu=powerpc**, and **−mcpu=powerpc64** specify generic POWER, POWER2, pure 32−bit PowerPC (i.e., not MPC601), and 64−bit PowerPC architecture machine types, with an appropriate, generic processor model assumed for scheduling purposes.

The other options specify a specific processor. Code generated under those options will run best on that processor, and may not run at all on others.

The **−mcpu** options automatically enable or disable the following options: **−maltivec**, **−mfprnd**, **−mhard−float**, **−mmfcrf**, **−mmultiple**, **−mnew−mnemonics**, **−mpopcntb**, **−mpower**, **−mpower2**, **−mpowerpc64**, **−mpowerpc−gpopt**, **−mpowerpc−gfxopt**, **−mstring**. The particular options set for any particular CPU will vary between compiler versions, depending on what setting seems to produce optimal code for that CPU; it doesn't necessarily reflect the actual hardware's capabilities. If you wish to set an individual option to a particular value, you may specify it after the **−mcpu** option, like **−mcpu=970 −mno−altivec**.

On AIX, the **−maltivec** and **−mpowerpc64** options are not enabled or disabled by the **−mcpu** option at present because AIX does not have full support for these options. You may still enable or disable them individually if you're sure it'll work in your environment.

−mtune=*cpu_type*
Set the instruction scheduling parameters for machine type *cpu_type*, but do not set the architecture type, register usage, or choice of mnemonics, as **−mcpu=***cpu_type* would. The same values for *cpu_type* are used for **−mtune** as for **−mcpu**. If both are specified, the code generated will use the architecture, registers, and mnemonics set by **−mcpu**, but the scheduling parameters set by **−mtune**.

−mswdiv
−mno−swdiv
Generate code to compute division as reciprocal estimate and iterative refinement, creating opportunities for increased throughput. This feature requires: optional PowerPC Graphics instruction set for single precision and FRE instruction for double precision, assuming divides cannot generate user-visible traps, and the domain values not include Infinities, denormals or zero denominator.

−maltivec
−mno−altivec
Generate code that uses (does not use) AltiVec instructions, and also enable the use of built-in functions that allow more direct access to the AltiVec instruction set. You may also need to set **−mabi=altivec** to adjust the current ABI with AltiVec ABI enhancements.

−mvrsave
−mno−vrsave
Generate VRSAVE instructions when generating AltiVec code.

−msecure−plt
Generate code that allows ld and ld.so to build executables and shared libraries with non-exec .plt and .got sections. This is a PowerPC 32−bit SYSV ABI option.

−mbss−plt
Generate code that uses a BSS .plt section that ld.so fills in, and requires .plt and .got sections that are both writable and executable. This is a PowerPC 32−bit SYSV ABI option.

−misel
−mno−isel
This switch enables or disables the generation of ISEL instructions.

−misel=*yes/no*
This switch has been deprecated. Use **−misel** and **−mno−isel** instead.

−mspe
−mno−isel
This switch enables or disables the generation of SPE simd instructions.

−mspe=*yes/no*
This option has been deprecated. Use **−mspe** and **−mno−spe** instead.

−mfloat−gprs=*yes/single/double/no*
−mfloat−gprs
This switch enables or disables the generation of floating point operations on the general purpose registers for architectures that support it.

The argument *yes* or *single* enables the use of single-precision floating point operations.

The argument *double* enables the use of single and double-precision floating point operations.

The argument *no* disables floating point operations on the general purpose registers.

This option is currently only available on the MPC854x.

−m32
−m64
Generate code for 32−bit or 64−bit environments of Darwin and SVR4 targets (including GNU/Linux). The 32−bit environment sets int, long and pointer to 32 bits and generates code that runs on any PowerPC variant. The 64−bit environment sets int to 32 bits and long and pointer to 64 bits, and generates code for PowerPC64, as for **−mpowerpc64**.

−mfull−toc
−mno−fp−in−toc

−mno−sum−in−toc
−mminimal−toc
Modify generation of the TOC (Table Of Contents), which is created for every executable file. The **−mfull−toc** option is selected by default. In that case, GCC will allocate at least one TOC entry for each unique non-automatic variable reference in your program. GCC will also place floating-point constants in the TOC. However, only 16,384 entries are available in the TOC.

If you receive a linker error message that saying you have overflowed the available TOC space, you can reduce the amount of TOC space used with the **−mno−fp−in−toc** and **−mno−sum−in−toc** options. **−mno−fp−in−toc** prevents GCC from putting floating-point constants in the TOC and **−mno−sum−in−toc** forces GCC to generate code to calculate the sum of an address and a constant at run-time instead of putting that sum into the TOC. You may specify one or both of these options. Each causes GCC to produce very slightly slower and larger code at the expense of conserving TOC space.

If you still run out of space in the TOC even when you specify both of these options, specify **−mminimal−toc** instead. This option causes GCC to make only one TOC entry for every file. When you specify this option, GCC will produce code that is slower and larger but which uses extremely little TOC space. You may wish to use this option only on files that contain less frequently executed code.

−maix64
−maix32
Enable 64−bit AIX ABI and calling convention: 64−bit pointers, 64−bit `long` type, and the infrastructure needed to support them. Specifying **−maix64** implies **−mpowerpc64** and **−mpowerpc**, while **−maix32** disables the 64−bit ABI and implies **−mno−powerpc64**. GCC defaults to **−maix32**.

−mxl−compat
−mno−xl−compat
Produce code that conforms more closely to IBM XL compiler semantics when using AIX-compatible ABI. Pass floating-point arguments to prototyped functions beyond the register save area (RSA) on the stack in addition to argument FPRs. Do not assume that most significant double in 128−bit long double value is properly rounded when comparing values and converting to double. Use XL symbol names for long double support routines.

The AIX calling convention was extended but not initially documented to handle an obscure K&R C case of calling a function that takes the address of its arguments with fewer arguments than declared. IBM XL compilers access floating point arguments which do not fit in the RSA from the stack when a subroutine is compiled without optimization. Because always storing floating-point arguments on the stack is inefficient and rarely needed, this option is not enabled by default and only is necessary when calling subroutines compiled by IBM XL compilers without optimization.

−mpe
Support *IBM RS/6000 SP Parallel Environment* (PE). Link an application written to use message passing with special startup code to enable the application to run. The system must have PE installed in the standard location (*/usr/lpp/ppe.poe/*), or the *specs* file must be overridden with the **−specs=** option to specify the appropriate directory location. The Parallel Environment does not support threads, so the **−mpe** option and the **−pthread** option are incompatible.

−malign−natural
−malign−power
On AIX, 32−bit Darwin, and 64−bit PowerPC GNU/Linux, the option **−malign−natural** overrides the ABI-defined alignment of larger types, such as floating-point doubles, on their natural size-based boundary. The option **−malign−power** instructs GCC to follow the ABI-specified alignment rules. GCC defaults to the standard alignment defined in the ABI.

On 64−bit Darwin, natural alignment is the default, and **−malign−power** is not supported.

−msoft−float
−mhard−float
Generate code that does not use (uses) the floating-point register set. Software floating point emulation is provided if you use the **−msoft−float** option, and pass the option to GCC when linking.

−mmultiple
−mno−multiple
Generate code that uses (does not use) the load multiple word instructions and the store multiple word instructions. These instructions are generated by default on POWER systems, and not generated on PowerPC systems. Do not use **−mmultiple** on little endian PowerPC systems, since those instructions do not work when the processor is in little endian mode. The exceptions are PPC740 and PPC750 which permit the instructions usage in little endian mode.

−mstring
−mno−string
Generate code that uses (does not use) the load string instructions and the store string word instructions to save multiple registers and do small block moves. These instructions are generated by default on POWER systems, and not generated on PowerPC systems. Do not use **−mstring** on little endian PowerPC systems, since those instructions do not work when the processor is in little endian mode. The exceptions are PPC740 and PPC750 which permit the instructions usage in little endian mode.

−mupdate
−mno−update
Generate code that uses (does not use) the load or store instructions that update the base register to the address of the calculated memory location. These instructions are generated by default. If you use **−mno−update**, there is a small window between the time that the stack pointer is updated and the address of the previous frame is stored, which means code that walks the stack frame across interrupts or signals may get corrupted data.

−mfused−madd
−mno−fused−madd
Generate code that uses (does not use) the floating point multiply and accumulate instructions. These instructions are generated by default if hardware floating is used.

−mno−bit−align
−mbit−align
On System V.4 and embedded PowerPC systems do not (do) force structures and unions that contain bit-fields to be aligned to the base type of the bit−field.

For example, by default a structure containing nothing but 8 `unsigned` bit-fields of length 1 would be aligned to a 4 byte boundary and have a size of 4 bytes. By using **−mno−bit−align**, the structure would be aligned to a 1 byte boundary and be one byte in size.

−mno−strict−align
−mstrict−align
On System V.4 and embedded PowerPC systems do not (do) assume that unaligned memory references will be handled by the system.

−mrelocatable
−mno−relocatable
On embedded PowerPC systems generate code that allows (does not allow) the program to be relocated to a different address at runtime. If you use **−mrelocatable** on any module, all objects linked together must be compiled with **−mrelocatable** or **−mrelocatable−lib**.

−mrelocatable−lib
−mno−relocatable−lib
On embedded PowerPC systems generate code that allows (does not allow) the program to be relocated to a different address at runtime. Modules compiled with **−mrelocatable−lib** can be linked with either modules compiled without **−mrelocatable** and **−mrelocatable−lib** or with modules compiled with the **−mrelocatable** options.

−mno−toc
−mtoc
On System V.4 and embedded PowerPC systems do not (do) assume that register 2 contains a pointer to a global area pointing to the addresses used in the program.

−mlittle
−mlittle−endian
On System V.4 and embedded PowerPC systems compile code for the processor in little endian mode. The **−mlittle−endian** option is the same as **−mlittle**.

−mbig
−mbig−endian
On System V.4 and embedded PowerPC systems compile code for the processor in big endian mode. The **−mbig−endian** option is the same as **−mbig**.

−mdynamic−no−pic
On Darwin and Mac OS X systems, compile code so that it is not relocatable, but that its external references are relocatable. The resulting code is suitable for applications, but not shared libraries.

−mprioritize−restricted−insns=*priority*
This option controls the priority that is assigned to dispatch-slot restricted instructions during the second scheduling pass. The argument *priority* takes the value *0/1/2* to assign *no/highest/second−highest* priority to dispatch slot restricted instructions.

−msched−costly−dep=*dependence_type*
This option controls which dependences are considered costly by the target during instruction scheduling. The argument *dependence_type* takes one of the following values: *no*: no dependence is costly, *all*: all dependences are costly, *true_store_to_load*: a true dependence from store to load is costly, *store_to_load*: any dependence from store to load is costly, *number*: any dependence which latency >= *number* is costly.

−minsert−sched−nops=*scheme*
This option controls which nop insertion scheme will be used during the second scheduling pass. The argument *scheme* takes one of the following values: *no*: Don't insert nops. *pad*: Pad with nops any dispatch group which has vacant issue slots, according to the scheduler's grouping. *regroup_exact*: Insert nops to force costly dependent insns into separate groups. Insert exactly as many nops as needed to force an insn to a new group, according to the estimated processor grouping. *number*: Insert nops to force costly dependent insns into separate groups. Insert *number* nops to force an insn to a new group.

−mcall−sysv
On System V.4 and embedded PowerPC systems compile code using calling conventions that adheres to the March 1995 draft of the System V Application Binary Interface, PowerPC processor supplement. This is the default unless you configured GCC using **powerpc−*−eabiaix**.

−mcall−sysv−eabi
Specify both **−mcall−sysv** and **−meabi** options.

−mcall−sysv−noeabi
Specify both **−mcall−sysv** and **−mno−eabi** options.

−mcall−solaris
On System V.4 and embedded PowerPC systems compile code for the Solaris operating system.

−mcall−linux
On System V.4 and embedded PowerPC systems compile code for the Linux-based GNU system.

−mcall−gnu
On System V.4 and embedded PowerPC systems compile code for the Hurd-based GNU system.

−mcall−netbsd
On System V.4 and embedded PowerPC systems compile code for the NetBSD operating system.

−maix−struct−return
Return all structures in memory (as specified by the AIX ABI).

−msvr4−struct−return
Return structures smaller than 8 bytes in registers (as specified by the SVR4 ABI).

−mabi=*abi-type*
Extend the current ABI with a particular extension, or remove such extension. Valid values are *altivec*, *no-altivec*, *spe*, *no-spe*, *ibmlongdouble*, *ieeelongdouble*.

−mabi=spe
Extend the current ABI with SPE ABI extensions. This does not change the default ABI, instead it adds the SPE ABI extensions to the current ABI.

−mabi=no−spe
Disable Booke SPE ABI extensions for the current ABI.

−mabi=ibmlongdouble
Change the current ABI to use IBM extended precision long double. This is a PowerPC 32−bit SYSV ABI option.

−mabi=ieeelongdouble
Change the current ABI to use IEEE extended precision long double. This is a PowerPC 32−bit Linux ABI option.

−mprototype
−mno−prototype
On System V.4 and embedded PowerPC systems assume that all calls to variable argument functions are properly prototyped. Otherwise, the compiler must insert an instruction before every non prototyped call to set or clear bit 6 of the condition code register (*CR*) to indicate whether floating point values were passed in the floating point registers in case the function takes a variable arguments. With **−mprototype**, only calls to prototyped variable argument functions will set or clear the bit.

−msim
On embedded PowerPC systems, assume that the startup module is called *sim−crt0.o* and that the standard C libraries are *libsim.a* and *libc.a*. This is the default for **powerpc−*−eabisim**. configurations.

−mmvme
On embedded PowerPC systems, assume that the startup module is called *crt0.o* and the standard C libraries are *libmvme.a* and *libc.a*.

−mads
On embedded PowerPC systems, assume that the startup module is called *crt0.o* and the standard C libraries are *libads.a* and *libc.a*.

−myellowknife
On embedded PowerPC systems, assume that the startup module is called *crt0.o* and the standard C libraries are *libyk.a* and *libc.a*.

−mvxworks
On System V.4 and embedded PowerPC systems, specify that you are compiling for a VxWorks system.

−mwindiss
Specify that you are compiling for the WindISS simulation environment.

−memb
On embedded PowerPC systems, set the *PPC_EMB* bit in the ELF flags header to indicate that **eabi** extended relocations are used.

−meabi
−mno−eabi
On System V.4 and embedded PowerPC systems do (do not) adhere to the Embedded Applications Binary Interface (eabi) which is a set of modifications to the System V.4 specifications. Selecting **−meabi** means that the stack is aligned to an 8 byte boundary, a function `__eabi` is called to from `main` to set up the eabi environment, and the **−msdata** option can use both `r2` and `r13` to point to two separate small data areas. Selecting **−mno−eabi** means that the stack is aligned to a 16 byte boundary, do not call an initialization function from `main`, and the **−msdata** option will only use `r13` to point to a single small data area. The **−meabi** option is on by default if you configured GCC using one of the **powerpc*−*−eabi*** options.

−msdata=eabi
On System V.4 and embedded PowerPC systems, put small initialized `const` global and static data in the **.sdata2** section, which is pointed to by register `r2`. Put small initialized non−`const` global and static data in the **.sdata** section, which is pointed to by register `r13`. Put small uninitialized global and static data in the **.sbss** section, which is adjacent to the **.sdata** section. The **−msdata=eabi** option is incompatible with the **−mrelocatable** option. The **−msdata=eabi** option also sets the **−memb** option.

−msdata=sysv
On System V.4 and embedded PowerPC systems, put small global and static data in the **.sdata** section, which is pointed to by register `r13`. Put small uninitialized global and static data in the **.sbss** section, which is adjacent to the **.sdata** section. The **−msdata=sysv** option is incompatible with the **−mrelocatable** option.

−msdata=default
−msdata
On System V.4 and embedded PowerPC systems, if **−meabi** is used, compile code the same as **−msdata=eabi**, otherwise compile code the same as **−msdata=sysv**.

−msdata−data
On System V.4 and embedded PowerPC systems, put small global and static data in the **.sdata** section. Put small uninitialized global and static data in the **.sbss** section. Do not use register `r13` to address small data however. This is the default behavior unless other **−msdata** options are used.

−msdata=none
−mno−sdata
On embedded PowerPC systems, put all initialized global and static data in the **.data** section, and all uninitialized data in the **.bss** section.

−G *num*
On embedded PowerPC systems, put global and static items less than or equal to *num* bytes into the small data or bss sections instead of the normal data or bss section. By default, *num* is 8. The **−G** *num* switch is also passed to the linker. All modules should be compiled with the same **−G** *num* value.

−mregnames
−mno−regnames
On System V.4 and embedded PowerPC systems do (do not) emit register names in the assembly language output using symbolic forms.

−mlongcall
−mno−longcall
Default to making all function calls indirectly, using a register, so that functions which reside further than 32 megabytes (33,554,432 bytes) from the current location can be called. This setting can be overridden by the `shortcall` function attribute, or by `#pragma longcall(0)`.

Some linkers are capable of detecting out-of-range calls and generating glue code on the fly. On these systems, long calls are unnecessary and generate slower code. As of this writing, the AIX linker can do this, as can the GNU linker for PowerPC/64. It is planned to add this feature to the GNU linker for 32−bit PowerPC systems as well.

On Darwin/PPC systems, `#pragma longcall` will generate ‘‘jbsr callee, L42’’, plus a ‘‘branch island’’ (glue code). The two target addresses represent the callee and the ‘‘branch island’’. The Darwin/PPC linker will prefer the first address and generate a ‘‘bl callee’’ if the PPC ‘‘bl’’ instruction will reach the callee directly; otherwise, the linker will generate ‘‘bl L42’’ to call the ‘‘branch island’’. The ‘‘branch island’’ is appended to the body of the calling function; it computes the full 32−bit address of the callee and jumps to it.

On Mach-O (Darwin) systems, this option directs the compiler emit to the glue for every direct call, and the Darwin linker decides whether to use or discard it.

In the future, we may cause GCC to ignore all longcall specifications when the linker is known to generate glue.

−pthread
Adds support for multithreading with the *pthreads* library. This option sets flags for both the preprocessor and linker.

S/390 and zSeries Options

These are the **−m** options defined for the S/390 and zSeries architecture.

−mhard−float
−msoft−float
Use (do not use) the hardware floating-point instructions and registers for floating-point operations. When **−msoft−float** is specified, functions in *libgcc.a* will be used to perform floating-point operations. When **−mhard−float** is specified, the compiler generates IEEE floating-point instructions. This is the default.

−mlong−double−64
−mlong−double−128
These switches control the size of `long double` type. A size of 64bit makes the `long double` type equivalent to the `double` type. This is the default.

−mbackchain
−mno−backchain
Store (do not store) the address of the caller’s frame as backchain pointer into the callee’s stack frame. A backchain may be needed to allow debugging using tools that do not understand DWARF−2 call frame information. When **−mno−packed−stack** is in effect, the backchain pointer is stored at the bottom of the stack frame; when **−mpacked−stack** is in effect, the backchain is placed into the topmost word of the 96/160 byte register save area.

In general, code compiled with **−mbackchain** is call-compatible with code compiled with **−mmo−backchain**; however, use of the backchain for debugging purposes usually requires that the whole binary is built with **−mbackchain**. Note that the combination of **−mbackchain**, **−mpacked−stack** and **−mhard−float** is not supported. In order to build a linux kernel use **−msoft−float**.

The default is to not maintain the backchain.

−mpacked−stack
−mno−packed−stack
Use (do not use) the packed stack layout. When **−mno−packed−stack** is specified, the compiler uses the all fields of the 96/160 byte register save area only for their default purpose; unused fields still take up stack space. When **−mpacked−stack** is specified, register save slots are densely packed at the top of the register save area; unused space is reused for other purposes, allowing for more efficient use of the available stack space. However, when **−mbackchain** is also in effect, the topmost word of the save area is always used to store the backchain, and the return address register is always saved two words below the backchain.

As long as the stack frame backchain is not used, code generated with **−mpacked−stack** is call-compatible with code generated with **−mno−packed−stack**. Note that some non-FSF releases of GCC

2.95 for S/390 or zSeries generated code that uses the stack frame backchain at run time, not just for debugging purposes. Such code is not call-compatible with code compiled with **−mpacked−stack**. Also, note that the combination of **−mbackchain**, **−mpacked−stack** and **−mhard−float** is not supported. In order to build a linux kernel use **−msoft−float**.

The default is to not use the packed stack layout.

−msmall−exec
−mno−small−exec
Generate (or do not generate) code using the `bras` instruction to do subroutine calls. This only works reliably if the total executable size does not exceed 64k. The default is to use the `basr` instruction instead, which does not have this limitation.

−m64
−m31
When **−m31** is specified, generate code compliant to the GNU/Linux for S/390 ABI. When **−m64** is specified, generate code compliant to the GNU/Linux for zSeries ABI. This allows GCC in particular to generate 64−bit instructions. For the **s390** targets, the default is **−m31**, while the **s390x** targets default to **−m64**.

−mzarch
−mesa
When **−mzarch** is specified, generate code using the instructions available on z/Architecture. When **−mesa** is specified, generate code using the instructions available on ESA/390. Note that **−mesa** is not possible with **−m64**. When generating code compliant to the GNU/Linux for S/390 ABI, the default is **−mesa**. When generating code compliant to the GNU/Linux for zSeries ABI, the default is **−mzarch**.

−mmvcle
−mno−mvcle
Generate (or do not generate) code using the `mvcle` instruction to perform block moves. When **−mno−mvcle** is specified, use a `mvc` loop instead. This is the default unless optimizing for size.

−mdebug
−mno−debug
Print (or do not print) additional debug information when compiling. The default is to not print debug information.

−march=*cpu-type*
Generate code that will run on *cpu-type*, which is the name of a system representing a certain processor type. Possible values for *cpu-type* are **g5**, **g6**, **z900**, and **z990**. When generating code using the instructions available on z/Architecture, the default is **−march=z900**. Otherwise, the default is **−march=g5**.

−mtune=*cpu-type*
Tune to *cpu-type* everything applicable about the generated code, except for the ABI and the set of available instructions. The list of *cpu-type* values is the same as for **−march**. The default is the value used for **−march**.

−mtpf−trace
−mno−tpf−trace
Generate code that adds (does not add) in TPF OS specific branches to trace routines in the operating system. This option is off by default, even when compiling for the TPF OS.

−mfused−madd
−mno−fused−madd
Generate code that uses (does not use) the floating point multiply and accumulate instructions. These instructions are generated by default if hardware floating point is used.

−mwarn−framesize=*framesize*
Emit a warning if the current function exceeds the given frame size. Because this is a compile time check it doesn't need to be a real problem when the program runs. It is intended to identify functions

which most probably cause a stack overflow. It is useful to be used in an environment with limited stack size e.g. the linux kernel.

−mwarn−dynamicstack
Emit a warning if the function calls alloca or uses dynamically sized arrays. This is generally a bad idea with a limited stack size.

−mstack−guard=*stack-guard*
−mstack−size=*stack-size*
These arguments always have to be used in conjunction. If they are present the s390 back end emits additional instructions in the function prologue which trigger a trap if the stack size is *stack-guard* bytes above the *stack-size* (remember that the stack on s390 grows downward). These options are intended to be used to help debugging stack overflow problems. The additionally emitted code causes only little overhead and hence can also be used in production like systems without greater performance degradation. The given values have to be exact powers of 2 and *stack-size* has to be greater than *stack-guard* without exceeding 64k. In order to be efficient the extra code makes the assumption that the stack starts at an address aligned to the value given by *stack-size*.

SH Options

These **−m** options are defined for the SH implementations:

−m1
Generate code for the SH1.

−m2
Generate code for the SH2.

−m2e
Generate code for the SH2e.

−m3
Generate code for the SH3.

−m3e
Generate code for the SH3e.

−m4−nofpu
Generate code for the SH4 without a floating-point unit.

−m4−single−only
Generate code for the SH4 with a floating-point unit that only supports single-precision arithmetic.

−m4−single
Generate code for the SH4 assuming the floating-point unit is in single-precision mode by default.

−m4
Generate code for the SH4.

−m4a−nofpu
Generate code for the SH4al−dsp, or for a SH4a in such a way that the floating-point unit is not used.

−m4a−single−only
Generate code for the SH4a, in such a way that no double-precision floating point operations are used.

−m4a−single
Generate code for the SH4a assuming the floating-point unit is in single-precision mode by default.

−m4a
Generate code for the SH4a.

−m4al
Same as **−m4a−nofpu**, except that it implicitly passes **−dsp** to the assembler. GCC doesn't generate any DSP instructions at the moment.

−mb
Compile code for the processor in big endian mode.

−ml
Compile code for the processor in little endian mode.

−mdalign
Align doubles at 64−bit boundaries. Note that this changes the calling conventions, and thus some functions from the standard C library will not work unless you recompile it first with **−mdalign**.

−mrelax
Shorten some address references at link time, when possible; uses the linker option **−relax**.

−mbigtable
Use 32−bit offsets in `switch` tables. The default is to use 16−bit offsets.

−mfmovd
Enable the use of the instruction `fmovd`.

−mhitachi
Comply with the calling conventions defined by Renesas.

−mrenesas
Comply with the calling conventions defined by Renesas.

−mno−renesas
Comply with the calling conventions defined for GCC before the Renesas conventions were available. This option is the default for all targets of the SH toolchain except for **sh-symbianelf**.

−mnomacsave
Mark the `MAC` register as call−clobbered, even if **−mhitachi** is given.

−mieee
Increase IEEE-compliance of floating-point code. At the moment, this is equivalent to **−fno−finite−math−only**. When generating 16 bit SH opcodes, getting IEEE-conforming results for comparisons of NANs / infinities incurs extra overhead in every floating point comparison, therefore the default is set to **−ffinite−math−only**.

−misize
Dump instruction size and location in the assembly code.

−mpadstruct
This option is deprecated. It pads structures to multiple of 4 bytes, which is incompatible with the SH ABI.

−mspace
Optimize for space instead of speed. Implied by **−Os**.

−mprefergot
When generating position-independent code, emit function calls using the Global Offset Table instead of the Procedure Linkage Table.

−musermode
Generate a library function call to invalidate instruction cache entries, after fixing up a trampoline. This library function call doesn't assume it can write to the whole memory address space. This is the default when the target is `sh−*−linux*`.

−multcost=*number*
Set the cost to assume for a multiply insn.

−mdiv=*strategy*
Set the division strategy to use for SHmedia code. *strategy* must be one of: call, call2, fp, inv, inv:minlat, inv20u, inv20l, inv:call, inv:call2, inv:fp . ''fp'' performs the operation in floating point. This has a very high latency, but needs only a few instructions, so it might be a good choice if your code has enough easily exploitable ILP to allow the compiler to schedule the floating point instructions together

with other instructions. Division by zero causes a floating point exception. ‘‘inv’’ uses integer operations to calculate the inverse of the divisor, and then multiplies the dividend with the inverse. This strategy allows cse and hoisting of the inverse calculation. Division by zero calculates an unspecified result, but does not trap. ‘‘inv:minlat’’ is a variant of ‘‘inv’’ where if no cse / hoisting opportunities have been found, or if the entire operation has been hoisted to the same place, the last stages of the inverse calculation are intertwined with the final multiply to reduce the overall latency, at the expense of using a few more instructions, and thus offering fewer scheduling opportunities with other code. ‘‘call’’ calls a library function that usually implements the inv:minlat strategy. This gives high code density for m5−*media−nofpu compilations. ‘‘call2’’ uses a different entry point of the same library function, where it assumes that a pointer to a lookup table has already been set up, which exposes the pointer load to cse / code hoisting optimizations. ‘‘inv:call’’, ‘‘inv:call2’’ and ‘‘inv:fp’’ all use the ‘‘inv’’ algorithm for initial code generation, but if the code stays unoptimized, revert to the ‘‘call’’, ‘‘call2’’, or ‘‘fp’’ strategies, respectively. Note that the potentially-trapping side effect of division by zero is carried by a separate instruction, so it is possible that all the integer instructions are hoisted out, but the marker for the side effect stays where it is. A recombination to fp operations or a call is not possible in that case. ‘‘inv20u’’ and ‘‘inv20l’’ are variants of the ‘‘inv:minlat’’ strategy. In the case that the inverse calculation was nor separated from the multiply, they speed up division where the dividend fits into 20 bits (plus sign where applicable), by inserting a test to skip a number of operations in this case; this test slows down the case of larger dividends. inv20u assumes the case of a such a small dividend to be unlikely, and inv20l assumes it to be likely.

−mdivsi3_libfunc=*name*

Set the name of the library function used for 32 bit signed division to *name*. This only affect the name used in the call and inv:call division strategies, and the compiler will still expect the same sets of input/output/clobbered registers as if this option was not present.

−madjust−unroll

Throttle unrolling to avoid thrashing target registers. This option only has an effect if the gcc code base supports the TARGET_ADJUST_UNROLL_MAX target hook.

−mindexed−addressing

Enable the use of the indexed addressing mode for SHmedia32/SHcompact. This is only safe if the hardware and/or OS implement 32 bit wrap-around semantics for the indexed addressing mode. The architecture allows the implementation of processors with 64 bit MMU, which the OS could use to get 32 bit addressing, but since no current hardware implementation supports this or any other way to make the indexed addressing mode safe to use in the 32 bit ABI, the default is −mno−indexed−addressing.

−mgettrcost=*number*

Set the cost assumed for the gettr instruction to *number*. The default is 2 if **−mpt−fixed** is in effect, 100 otherwise.

−mpt−fixed

Assume pt* instructions won’t trap. This will generally generate better scheduled code, but is unsafe on current hardware. The current architecture definition says that ptabs and ptrel trap when the target anded with 3 is 3. This has the unintentional effect of making it unsafe to schedule ptabs / ptrel before a branch, or hoist it out of a loop. For example, __do_global_ctors, a part of libgcc that runs constructors at program startup, calls functions in a list which is delimited by −1. With the −mpt−fixed option, the ptabs will be done before testing against −1. That means that all the constructors will be run a bit quicker, but when the loop comes to the end of the list, the program crashes because ptabs loads −1 into a target register. Since this option is unsafe for any hardware implementing the current architecture specification, the default is −mno−pt−fixed. Unless the user specifies a specific cost with **−mgettrcost**, −mno−pt−fixed also implies **−mgettrcost=100**; this deters register allocation using target registers for storing ordinary integers.

−minvalid−symbols

Assume symbols might be invalid. Ordinary function symbols generated by the compiler will always be valid to load with movi/shori/ptabs or movi/shori/ptrel, but with assembler and/or linker tricks it is

possible to generate symbols that will cause ptabs / ptrel to trap. This option is only meaningful when **−mno−pt−fixed** is in effect. It will then prevent cross-basic-block cse, hoisting and most scheduling of symbol loads. The default is **−mno−invalid−symbols**.

SPARC Options

These **−m** options are supported on the SPARC:

−mno−app−regs
−mapp−regs
Specify **−mapp−regs** to generate output using the global registers 2 through 4, which the SPARC SVR4 ABI reserves for applications. This is the default.

To be fully SVR4 ABI compliant at the cost of some performance loss, specify **−mno−app−regs**. You should compile libraries and system software with this option.

−mfpu
−mhard−float
Generate output containing floating point instructions. This is the default.

−mno−fpu
−msoft−float
Generate output containing library calls for floating point. **Warning:** the requisite libraries are not available for all SPARC targets. Normally the facilities of the machine's usual C compiler are used, but this cannot be done directly in cross−compilation. You must make your own arrangements to provide suitable library functions for cross−compilation. The embedded targets **sparc−*−aout** and **sparclite−*−*** do provide software floating point support.

−msoft−float changes the calling convention in the output file; therefore, it is only useful if you compile *all* of a program with this option. In particular, you need to compile *libgcc.a*, the library that comes with GCC, with **−msoft−float** in order for this to work.

−mhard−quad−float
Generate output containing quad-word (long double) floating point instructions.

−msoft−quad−float
Generate output containing library calls for quad-word (long double) floating point instructions. The functions called are those specified in the SPARC ABI. This is the default.

As of this writing, there are no SPARC implementations that have hardware support for the quad-word floating point instructions. They all invoke a trap handler for one of these instructions, and then the trap handler emulates the effect of the instruction. Because of the trap handler overhead, this is much slower than calling the ABI library routines. Thus the **−msoft−quad−float** option is the default.

−mno−unaligned−doubles
−munaligned−doubles
Assume that doubles have 8 byte alignment. This is the default.

With **−munaligned−doubles**, GCC assumes that doubles have 8 byte alignment only if they are contained in another type, or if they have an absolute address. Otherwise, it assumes they have 4 byte alignment. Specifying this option avoids some rare compatibility problems with code generated by other compilers. It is not the default because it results in a performance loss, especially for floating point code.

−mno−faster−structs
−mfaster−structs
With **−mfaster−structs**, the compiler assumes that structures should have 8 byte alignment. This enables the use of pairs of `ldd` and `std` instructions for copies in structure assignment, in place of twice as many `ld` and `st` pairs. However, the use of this changed alignment directly violates the SPARC ABI. Thus, it's intended only for use on targets where the developer acknowledges that their resulting code will not be directly in line with the rules of the ABI.

−mimpure−text

−mimpure−text, used in addition to **−shared**, tells the compiler to not pass **−z text** to the linker when linking a shared object. Using this option, you can link position-dependent code into a shared object.

−mimpure−text suppresses the ‘‘relocations remain against allocatable but non-writable sections’’ linker error message. However, the necessary relocations will trigger copy−on−write, and the shared object is not actually shared across processes. Instead of using **−mimpure−text**, you should compile all source code with **−fpic** or **−fPIC**.

This option is only available on SunOS and Solaris.

−mcpu=*cpu_type*

Set the instruction set, register set, and instruction scheduling parameters for machine type *cpu_type*. Supported values for *cpu_type* are **v7**, **cypress**, **v8**, **supersparc**, **sparclite**, **f930**, **f934**, **hypersparc**, **sparclite86x**, **sparclet**, **tsc701**, **v9**, **ultrasparc**, and **ultrasparc3**.

Default instruction scheduling parameters are used for values that select an architecture and not an implementation. These are **v7**, **v8**, **sparclite**, **sparclet**, **v9**.

Here is a list of each supported architecture and their supported implementations.

```
v7:             cypress
v8:             supersparc, hypersparc
sparclite:      f930, f934, sparclite86x
sparclet:       tsc701
v9:             ultrasparc, ultrasparc3
```

By default (unless configured otherwise), GCC generates code for the V7 variant of the SPARC architecture. With **−mcpu=cypress**, the compiler additionally optimizes it for the Cypress CY7C602 chip, as used in the SPARCStation/SPARCServer 3xx series. This is also appropriate for the older SPARCStation 1, 2, IPX etc.

With **−mcpu=v8**, GCC generates code for the V8 variant of the SPARC architecture. The only difference from V7 code is that the compiler emits the integer multiply and integer divide instructions which exist in SPARC−V8 but not in SPARC−V7. With **−mcpu=supersparc**, the compiler additionally optimizes it for the SuperSPARC chip, as used in the SPARCStation 10, 1000 and 2000 series.

With **−mcpu=sparclite**, GCC generates code for the SPARClite variant of the SPARC architecture. This adds the integer multiply, integer divide step and scan (`ffs`) instructions which exist in SPARClite but not in SPARC−V7. With **−mcpu=f930**, the compiler additionally optimizes it for the Fujitsu MB86930 chip, which is the original SPARClite, with no FPU. With **−mcpu=f934**, the compiler additionally optimizes it for the Fujitsu MB86934 chip, which is the more recent SPARClite with FPU.

With **−mcpu=sparclet**, GCC generates code for the SPARClet variant of the SPARC architecture. This adds the integer multiply, multiply/accumulate, integer divide step and scan (`ffs`) instructions which exist in SPARClet but not in SPARC−V7. With **−mcpu=tsc701**, the compiler additionally optimizes it for the TEMIC SPARClet chip.

With **−mcpu=v9**, GCC generates code for the V9 variant of the SPARC architecture. This adds 64−bit integer and floating-point move instructions, 3 additional floating-point condition code registers and conditional move instructions. With **−mcpu=ultrasparc**, the compiler additionally optimizes it for the Sun UltraSPARC I/II chips. With **−mcpu=ultrasparc3**, the compiler additionally optimizes it for the Sun UltraSPARC III chip.

−mtune=*cpu_type*

Set the instruction scheduling parameters for machine type *cpu_type*, but do not set the instruction set or register set that the option **−mcpu=***cpu_type* would.

The same values for **−mcpu=***cpu_type* can be used for **−mtune=***cpu_type*, but the only useful values are those that select a particular cpu implementation. Those are **cypress**, **supersparc**, **hypersparc**, **f930**, **f934**, **sparclite86x**, **tsc701**, **ultrasparc**, and **ultrasparc3**.

−mv8plus
−mno−v8plus
With **−mv8plus**, GCC generates code for the SPARC−V8+ ABI. The difference from the V8 ABI is that the global and out registers are considered 64−bit wide. This is enabled by default on Solaris in 32−bit mode for all SPARC−V9 processors.

−mvis
−mno−vis
With **−mvis**, GCC generates code that takes advantage of the UltraSPARC Visual Instruction Set extensions. The default is **−mno−vis**.

These **−m** options are supported in addition to the above on SPARC−V9 processors in 64−bit environments:

−mlittle−endian
Generate code for a processor running in little-endian mode. It is only available for a few configurations and most notably not on Solaris and Linux.

−m32
−m64
Generate code for a 32−bit or 64−bit environment. The 32−bit environment sets int, long and pointer to 32 bits. The 64−bit environment sets int to 32 bits and long and pointer to 64 bits.

−mcmodel=medlow
Generate code for the Medium/Low code model: 64−bit addresses, programs must be linked in the low 32 bits of memory. Programs can be statically or dynamically linked.

−mcmodel=medmid
Generate code for the Medium/Middle code model: 64−bit addresses, programs must be linked in the low 44 bits of memory, the text and data segments must be less than 2GB in size and the data segment must be located within 2GB of the text segment.

−mcmodel=medany
Generate code for the Medium/Anywhere code model: 64−bit addresses, programs may be linked anywhere in memory, the text and data segments must be less than 2GB in size and the data segment must be located within 2GB of the text segment.

−mcmodel=embmedany
Generate code for the Medium/Anywhere code model for embedded systems: 64−bit addresses, the text and data segments must be less than 2GB in size, both starting anywhere in memory (determined at link time). The global register %g4 points to the base of the data segment. Programs are statically linked and PIC is not supported.

−mstack−bias
−mno−stack−bias
With **−mstack−bias**, GCC assumes that the stack pointer, and frame pointer if present, are offset by −2047 which must be added back when making stack frame references. This is the default in 64−bit mode. Otherwise, assume no such offset is present.

These switches are supported in addition to the above on Solaris:

−threads
Add support for multithreading using the Solaris threads library. This option sets flags for both the preprocessor and linker. This option does not affect the thread safety of object code produced by the compiler or that of libraries supplied with it.

−pthreads
Add support for multithreading using the POSIX threads library. This option sets flags for both the preprocessor and linker. This option does not affect the thread safety of object code produced by the compiler or that of libraries supplied with it.

−pthread
This is a synonym for **−pthreads**.

Options for System V

These additional options are available on System V Release 4 for compatibility with other compilers on those systems:

−G Create a shared object. It is recommended that **−symbolic** or **−shared** be used instead.

−Qy
Identify the versions of each tool used by the compiler, in a `.ident` assembler directive in the output.

−Qn
Refrain from adding `.ident` directives to the output file (this is the default).

−YP,*dirs*
Search the directories *dirs*, and no others, for libraries specified with **−l**.

−Ym,*dir*
Look in the directory *dir* to find the M4 preprocessor. The assembler uses this option.

TMS320C3x/C4x Options

These **−m** options are defined for TMS320C3x/C4x implementations:

−mcpu=*cpu_type*
Set the instruction set, register set, and instruction scheduling parameters for machine type *cpu_type*. Supported values for *cpu_type* are **c30**, **c31**, **c32**, **c40**, and **c44**. The default is **c40** to generate code for the TMS320C40.

−mbig−memory
−mbig
−msmall−memory
−msmall
Generates code for the big or small memory model. The small memory model assumed that all data fits into one 64K word page. At run-time the data page (DP) register must be set to point to the 64K page containing the .bss and .data program sections. The big memory model is the default and requires reloading of the DP register for every direct memory access.

−mbk
−mno−bk
Allow (disallow) allocation of general integer operands into the block count register BK.

−mdb
−mno−db
Enable (disable) generation of code using decrement and branch, DBcond(D), instructions. This is enabled by default for the C4x. To be on the safe side, this is disabled for the C3x, since the maximum iteration count on the C3x is 2^{23 + 1} (but who iterates loops more than 2^{23} times on the C3x?). Note that GCC will try to reverse a loop so that it can utilize the decrement and branch instruction, but will give up if there is more than one memory reference in the loop. Thus a loop where the loop counter is decremented can generate slightly more efficient code, in cases where the RPTB instruction cannot be utilized.

−mdp−isr−reload
−mparanoid
Force the DP register to be saved on entry to an interrupt service routine (ISR), reloaded to point to the data section, and restored on exit from the ISR. This should not be required unless someone has violated the small memory model by modifying the DP register, say within an object library.

−mmpyi

−mno−mpyi
For the C3x use the 24−bit MPYI instruction for integer multiplies instead of a library call to guarantee 32−bit results. Note that if one of the operands is a constant, then the multiplication will be performed using shifts and adds. If the **−mmpyi** option is not specified for the C3x, then squaring operations are performed inline instead of a library call.

−mfast−fix
−mno−fast−fix
The C3x/C4x FIX instruction to convert a floating point value to an integer value chooses the nearest integer less than or equal to the floating point value rather than to the nearest integer. Thus if the floating point number is negative, the result will be incorrectly truncated an additional code is necessary to detect and correct this case. This option can be used to disable generation of the additional code required to correct the result.

−mrptb
−mno−rptb
Enable (disable) generation of repeat block sequences using the RPTB instruction for zero overhead looping. The RPTB construct is only used for innermost loops that do not call functions or jump across the loop boundaries. There is no advantage having nested RPTB loops due to the overhead required to save and restore the RC, RS, and RE registers. This is enabled by default with **−O2**.

−mrpts=*count*
−mno−rpts
Enable (disable) the use of the single instruction repeat instruction RPTS. If a repeat block contains a single instruction, and the loop count can be guaranteed to be less than the value *count*, GCC will emit a RPTS instruction instead of a RPTB. If no value is specified, then a RPTS will be emitted even if the loop count cannot be determined at compile time. Note that the repeated instruction following RPTS does not have to be reloaded from memory each iteration, thus freeing up the CPU buses for operands. However, since interrupts are blocked by this instruction, it is disabled by default.

−mloop−unsigned
−mno−loop−unsigned
The maximum iteration count when using RPTS and RPTB (and DB on the C40) is 2^{31 + 1} since these instructions test if the iteration count is negative to terminate the loop. If the iteration count is unsigned there is a possibility than the 2^{31 + 1} maximum iteration count may be exceeded. This switch allows an unsigned iteration count.

−mti
Try to emit an assembler syntax that the TI assembler (asm30) is happy with. This also enforces compatibility with the API employed by the TI C3x C compiler. For example, long doubles are passed as structures rather than in floating point registers.

−mregparm
−mmemparm
Generate code that uses registers (stack) for passing arguments to functions. By default, arguments are passed in registers where possible rather than by pushing arguments on to the stack.

−mparallel−insns
−mno−parallel−insns
Allow the generation of parallel instructions. This is enabled by default with **−O2**.

−mparallel−mpy
−mno−parallel−mpy
Allow the generation of MPY||ADD and MPY||SUB parallel instructions, provided **−mparallel−insns** is also specified. These instructions have tight register constraints which can pessimize the code generation of large functions.

V850 Options

These **−m** options are defined for V850 implementations:

−mlong−calls
−mno−long−calls
Treat all calls as being far away (near). If calls are assumed to be far away, the compiler will always load the functions address up into a register, and call indirect through the pointer.

−mno−ep
−mep
Do not optimize (do optimize) basic blocks that use the same index pointer 4 or more times to copy pointer into the `ep` register, and use the shorter `sld` and `sst` instructions. The **−mep** option is on by default if you optimize.

−mno−prolog−function
−mprolog−function
Do not use (do use) external functions to save and restore registers at the prologue and epilogue of a function. The external functions are slower, but use less code space if more than one function saves the same number of registers. The **−mprolog−function** option is on by default if you optimize.

−mspace
Try to make the code as small as possible. At present, this just turns on the **−mep** and **−mprolog−function** options.

−mtda=*n*
Put static or global variables whose size is *n* bytes or less into the tiny data area that register `ep` points to. The tiny data area can hold up to 256 bytes in total (128 bytes for byte references).

−msda=*n*
Put static or global variables whose size is *n* bytes or less into the small data area that register `gp` points to. The small data area can hold up to 64 kilobytes.

−mzda=*n*
Put static or global variables whose size is *n* bytes or less into the first 32 kilobytes of memory.

−mv850
Specify that the target processor is the V850.

−mbig−switch
Generate code suitable for big switch tables. Use this option only if the assembler/linker complain about out of range branches within a switch table.

−mapp−regs
This option will cause r2 and r5 to be used in the code generated by the compiler. This setting is the default.

−mno−app−regs
This option will cause r2 and r5 to be treated as fixed registers.

−mv850e1
Specify that the target processor is the V850E1. The preprocessor constants **__v850e1__** and **__v850e__** will be defined if this option is used.

−mv850e
Specify that the target processor is the V850E. The preprocessor constant **__v850e__** will be defined if this option is used.

If neither **−mv850** nor **−mv850e** nor **−mv850e1** are defined then a default target processor will be chosen and the relevant **__v850*__** preprocessor constant will be defined.

The preprocessor constants **__v850** and **__v851__** are always defined, regardless of which processor variant is the target.

−mdisable−callt
This option will suppress generation of the CALLT instruction for the v850e and v850e1 flavors of the v850 architecture. The default is **−mno−disable−callt** which allows the CALLT instruction to be used.

VAX Options

These **−m** options are defined for the VAX:

−munix
Do not output certain jump instructions (`aobleq` and so on) that the Unix assembler for the VAX cannot handle across long ranges.

−mgnu
Do output those jump instructions, on the assumption that you will assemble with the GNU assembler.

−mg
Output code for g−format floating point numbers instead of d−format.

x86−64 Options

These are listed under

Xstormy16 Options

These options are defined for Xstormy16:

−msim
Choose startup files and linker script suitable for the simulator.

Xtensa Options

These options are supported for Xtensa targets:

−mconst16
−mno−const16
Enable or disable use of `CONST16` instructions for loading constant values. The `CONST16` instruction is currently not a standard option from Tensilica. When enabled, `CONST16` instructions are always used in place of the standard `L32R` instructions. The use of `CONST16` is enabled by default only if the `L32R` instruction is not available.

−mfused−madd
−mno−fused−madd
Enable or disable use of fused multiply/add and multiply/subtract instructions in the floating-point option. This has no effect if the floating-point option is not also enabled. Disabling fused multiply/add and multiply/subtract instructions forces the compiler to use separate instructions for the multiply and add/subtract operations. This may be desirable in some cases where strict IEEE 754−compliant results are required: the fused multiply add/subtract instructions do not round the intermediate result, thereby producing results with *more* bits of precision than specified by the IEEE standard. Disabling fused multiply add/subtract instructions also ensures that the program output is not sensitive to the compiler's ability to combine multiply and add/subtract operations.

−mtext−section−literals
−mno−text−section−literals
Control the treatment of literal pools. The default is **−mno−text−section−literals**, which places literals in a separate section in the output file. This allows the literal pool to be placed in a data RAM/ROM, and it also allows the linker to combine literal pools from separate object files to remove redundant literals and improve code size. With **−mtext−section−literals**, the literals are interspersed in the text section in order to keep them as close as possible to their references. This may be necessary for large assembly files.

−mtarget−align
−mno−target−align
When this option is enabled, GCC instructs the assembler to automatically align instructions to reduce branch penalties at the expense of some code density. The assembler attempts to widen density instructions to align branch targets and the instructions following call instructions. If there are not enough preceding safe density instructions to align a target, no widening will be performed. The default is **−mtarget−align**. These options do not affect the treatment of auto-aligned instructions like

LOOP, which the assembler will always align, either by widening density instructions or by inserting no-op instructions.

−mlongcalls
−mno−longcalls
When this option is enabled, GCC instructs the assembler to translate direct calls to indirect calls unless it can determine that the target of a direct call is in the range allowed by the call instruction. This translation typically occurs for calls to functions in other source files. Specifically, the assembler translates a direct CALL instruction into an L32R followed by a CALLX instruction. The default is **−mno−longcalls**. This option should be used in programs where the call target can potentially be out of range. This option is implemented in the assembler, not the compiler, so the assembly code generated by GCC will still show direct call instructions−−−look at the disassembled object code to see the actual instructions. Note that the assembler will use an indirect call for every cross-file call, not just those that really will be out of range.

zSeries Options

These are listed under

Options for Code Generation Conventions

These machine-independent options control the interface conventions used in code generation.

Most of them have both positive and negative forms; the negative form of **−ffoo** would be **−fno−foo**. In the table below, only one of the forms is listed−−−the one which is not the default. You can figure out the other form by either removing **no−** or adding it.

−fbounds−check
For front-ends that support it, generate additional code to check that indices used to access arrays are within the declared range. This is currently only supported by the Java and Fortran 77 front−ends, where this option defaults to true and false respectively.

−ftrapv
This option generates traps for signed overflow on addition, subtraction, multiplication operations.

−fwrapv
This option instructs the compiler to assume that signed arithmetic overflow of addition, subtraction and multiplication wraps around using twos-complement representation. This flag enables some optimizations and disables others. This option is enabled by default for the Java front−end, as required by the Java language specification.

−fexceptions
Enable exception handling. Generates extra code needed to propagate exceptions. For some targets, this implies GCC will generate frame unwind information for all functions, which can produce significant data size overhead, although it does not affect execution. If you do not specify this option, GCC will enable it by default for languages like C++ which normally require exception handling, and disable it for languages like C that do not normally require it. However, you may need to enable this option when compiling C code that needs to interoperate properly with exception handlers written in C++. You may also wish to disable this option if you are compiling older C++ programs that don't use exception handling.

−fnon−call−exceptions
Generate code that allows trapping instructions to throw exceptions. Note that this requires platform-specific runtime support that does not exist everywhere. Moreover, it only allows *trapping* instructions to throw exceptions, i.e. memory references or floating point instructions. It does not allow exceptions to be thrown from arbitrary signal handlers such as SIGALRM.

−funwind−tables
Similar to **−fexceptions**, except that it will just generate any needed static data, but will not affect the generated code in any other way. You will normally not enable this option; instead, a language processor that needs this handling would enable it on your behalf.

−fasynchronous−unwind−tables
Generate unwind table in dwarf2 format, if supported by target machine. The table is exact at each instruction boundary, so it can be used for stack unwinding from asynchronous events (such as debugger or garbage collector).

−fpcc−struct−return
Return ‘‘short’’ `struct` and `union` values in memory like longer ones, rather than in registers. This convention is less efficient, but it has the advantage of allowing intercallability between GCC-compiled files and files compiled with other compilers, particularly the Portable C Compiler (pcc).

The precise convention for returning structures in memory depends on the target configuration macros.

Short structures and unions are those whose size and alignment match that of some integer type.

Warning: code compiled with the **−fpcc−struct−return** switch is not binary compatible with code compiled with the **−freg−struct−return** switch. Use it to conform to a non-default application binary interface.

−freg−struct−return
Return `struct` and `union` values in registers when possible. This is more efficient for small structures than **−fpcc−struct−return**.

If you specify neither **−fpcc−struct−return** nor **−freg−struct−return**, GCC defaults to whichever convention is standard for the target. If there is no standard convention, GCC defaults to **−fpcc−struct−return**, except on targets where GCC is the principal compiler. In those cases, we can choose the standard, and we chose the more efficient register return alternative.

Warning: code compiled with the **−freg−struct−return** switch is not binary compatible with code compiled with the **−fpcc−struct−return** switch. Use it to conform to a non-default application binary interface.

−fshort−enums
Allocate to an `enum` type only as many bytes as it needs for the declared range of possible values. Specifically, the `enum` type will be equivalent to the smallest integer type which has enough room.

Warning: the **−fshort−enums** switch causes GCC to generate code that is not binary compatible with code generated without that switch. Use it to conform to a non-default application binary interface.

−fshort−double
Use the same size for `double` as for `float`.

Warning: the **−fshort−double** switch causes GCC to generate code that is not binary compatible with code generated without that switch. Use it to conform to a non-default application binary interface.

−fshort−wchar
Override the underlying type for **wchar_t** to be **short unsigned int** instead of the default for the target. This option is useful for building programs to run under WINE.

Warning: the **−fshort−wchar** switch causes GCC to generate code that is not binary compatible with code generated without that switch. Use it to conform to a non-default application binary interface.

−fshared−data
Requests that the data and non−`const` variables of this compilation be shared data rather than private data. The distinction makes sense only on certain operating systems, where shared data is shared between processes running the same program, while private data exists in one copy per process.

−fno−common
In C, allocate even uninitialized global variables in the data section of the object file, rather than generating them as common blocks. This has the effect that if the same variable is declared (without `extern`) in two different compilations, you will get an error when you link them. The only reason this might be useful is if you wish to verify that the program will work on other systems which always work this way.

−fno−ident
Ignore the **#ident** directive.

−finhibit−size−directive
Don't output a `.size` assembler directive, or anything else that would cause trouble if the function is split in the middle, and the two halves are placed at locations far apart in memory. This option is used when compiling *crtstuff.c*; you should not need to use it for anything else.

−fverbose−asm
Put extra commentary information in the generated assembly code to make it more readable. This option is generally only of use to those who actually need to read the generated assembly code (perhaps while debugging the compiler itself).

−fno−verbose−asm, the default, causes the extra information to be omitted and is useful when comparing two assembler files.

−fpic
Generate position-independent code (PIC) suitable for use in a shared library, if supported for the target machine. Such code accesses all constant addresses through a global offset table (GOT). The dynamic loader resolves the GOT entries when the program starts (the dynamic loader is not part of GCC; it is part of the operating system). If the GOT size for the linked executable exceeds a machine-specific maximum size, you get an error message from the linker indicating that **−fpic** does not work; in that case, recompile with **−fPIC** instead. (These maximums are 8k on the SPARC and 32k on the m68k and RS/6000. The 386 has no such limit.)

Position-independent code requires special support, and therefore works only on certain machines. For the 386, GCC supports PIC for System V but not for the Sun 386i. Code generated for the IBM RS/6000 is always position−independent.

−fPIC
If supported for the target machine, emit position-independent code, suitable for dynamic linking and avoiding any limit on the size of the global offset table. This option makes a difference on the m68k, PowerPC and SPARC.

Position-independent code requires special support, and therefore works only on certain machines.

−fpie
−fPIE
These options are similar to **−fpic** and **−fPIC**, but generated position independent code can be only linked into executables. Usually these options are used when **−pie** GCC option will be used during linking.

−fno−jump−tables
Do not use jump tables for switch statements even where it would be more efficient than other code generation strategies. This option is of use in conjunction with **−fpic** or **−fPIC** for building code which forms part of a dynamic linker and cannot reference the address of a jump table. On some targets, jump tables do not require a GOT and this option is not needed.

−ffixed−*reg*
Treat the register named *reg* as a fixed register; generated code should never refer to it (except perhaps as a stack pointer, frame pointer or in some other fixed role).

reg must be the name of a register. The register names accepted are machine-specific and are defined in the `REGISTER_NAMES` macro in the machine description macro file.

This flag does not have a negative form, because it specifies a three-way choice.

−fcall−used−*reg*
Treat the register named *reg* as an allocable register that is clobbered by function calls. It may be allocated for temporaries or variables that do not live across a call. Functions compiled this way will not save and restore the register *reg*.

It is an error to used this flag with the frame pointer or stack pointer. Use of this flag for other registers that have fixed pervasive roles in the machine's execution model will produce disastrous results.

This flag does not have a negative form, because it specifies a three-way choice.

−fcall−saved−*reg*
Treat the register named *reg* as an allocable register saved by functions. It may be allocated even for temporaries or variables that live across a call. Functions compiled this way will save and restore the register *reg* if they use it.

It is an error to used this flag with the frame pointer or stack pointer. Use of this flag for other registers that have fixed pervasive roles in the machine's execution model will produce disastrous results.

A different sort of disaster will result from the use of this flag for a register in which function values may be returned.

This flag does not have a negative form, because it specifies a three-way choice.

−fpack−struct[=*n***]**
Without a value specified, pack all structure members together without holes. When a value is specified (which must be a small power of two), pack structure members according to this value, representing the maximum alignment (that is, objects with default alignment requirements larger than this will be output potentially unaligned at the next fitting location.

Warning: the **−fpack−struct** switch causes GCC to generate code that is not binary compatible with code generated without that switch. Additionally, it makes the code suboptimal. Use it to conform to a non-default application binary interface.

−finstrument−functions
Generate instrumentation calls for entry and exit to functions. Just after function entry and just before function exit, the following profiling functions will be called with the address of the current function and its call site. (On some platforms, `__builtin_return_address` does not work beyond the current function, so the call site information may not be available to the profiling functions otherwise.)

```
void __cyg_profile_func_enter (void *this_fn,
                               void *call_site);
void __cyg_profile_func_exit  (void *this_fn,
                               void *call_site);
```

The first argument is the address of the start of the current function, which may be looked up exactly in the symbol table.

This instrumentation is also done for functions expanded inline in other functions. The profiling calls will indicate where, conceptually, the inline function is entered and exited. This means that addressable versions of such functions must be available. If all your uses of a function are expanded inline, this may mean an additional expansion of code size. If you use **extern inline** in your C code, an addressable version of such functions must be provided. (This is normally the case anyways, but if you get lucky and the optimizer always expands the functions inline, you might have gotten away without providing static copies.)

A function may be given the attribute `no_instrument_function`, in which case this instrumentation will not be done. This can be used, for example, for the profiling functions listed above, high-priority interrupt routines, and any functions from which the profiling functions cannot safely be called (perhaps signal handlers, if the profiling routines generate output or allocate memory).

−fstack−check
Generate code to verify that you do not go beyond the boundary of the stack. You should specify this flag if you are running in an environment with multiple threads, but only rarely need to specify it in a single-threaded environment since stack overflow is automatically detected on nearly all systems if there is only one stack.

Note that this switch does not actually cause checking to be done; the operating system must do that.

The switch causes generation of code to ensure that the operating system sees the stack being extended.

−fstack−limit−register=*reg*
−fstack−limit−symbol=*sym*
−fno−stack−limit
Generate code to ensure that the stack does not grow beyond a certain value, either the value of a register or the address of a symbol. If the stack would grow beyond the value, a signal is raised. For most targets, the signal is raised before the stack overruns the boundary, so it is possible to catch the signal without taking special precautions.

For instance, if the stack starts at absolute address **0x80000000** and grows downwards, you can use the flags **−fstack−limit−symbol=__stack_limit** and **−Wl,−−defsym,__stack_limit=0x7ffe0000** to enforce a stack limit of 128KB. Note that this may only work with the GNU linker.

−fargument−alias
−fargument−noalias
−fargument−noalias−global
Specify the possible relationships among parameters and between parameters and global data.

−fargument−alias specifies that arguments (parameters) may alias each other and may alias global storage.**−fargument−noalias** specifies that arguments do not alias each other, but may alias global storage.**−fargument−noalias−global** specifies that arguments do not alias each other and do not alias global storage.

Each language will automatically use whatever option is required by the language standard. You should not need to use these options yourself.

−fleading−underscore
This option and its counterpart, **−fno−leading−underscore**, forcibly change the way C symbols are represented in the object file. One use is to help link with legacy assembly code.

Warning: the **−fleading−underscore** switch causes GCC to generate code that is not binary compatible with code generated without that switch. Use it to conform to a non-default application binary interface. Not all targets provide complete support for this switch.

−ftls−model=*model*
Alter the thread-local storage model to be used. The *model* argument should be one of `global−dynamic`, `local−dynamic`, `initial−exec` or `local−exec`.

The default without **−fpic** is `initial−exec`; with **−fpic** the default is `global−dynamic`.

−fvisibility=*default*|*internal*|*hidden*|*protected*
Set the default ELF image symbol visibility to the specified option−−−all symbols will be marked with this unless overridden within the code. Using this feature can very substantially improve linking and load times of shared object libraries, produce more optimized code, provide near-perfect API export and prevent symbol clashes. It is **strongly** recommended that you use this in any shared objects you distribute.

Despite the nomenclature, `default` always means public ie; available to be linked against from outside the shared object. `protected` and `internal` are pretty useless in real-world usage so the only other commonly used option will be `hidden`. The default if **−fvisibility** isn't specified is `default`, i.e., make every symbol public−−−this causes the same behavior as previous versions of GCC.

A good explanation of the benefits offered by ensuring ELF symbols have the correct visibility is given by "How To Write Shared Libraries" by Ulrich Drepper (which can be found at <**http://people.redhat.com/~drepper/**>)−−−however a superior solution made possible by this option to marking things hidden when the default is public is to make the default hidden and mark things public. This is the norm with DLL's on Windows and with **−fvisibility=hidden** and `__attribute__ ((visibility("default")))` instead of `__declspec(dllexport)` you get almost identical semantics

with identical syntax. This is a great boon to those working with cross-platform projects.

For those adding visibility support to existing code, you may find **#pragma GCC visibility** of use. This works by you enclosing the declarations you wish to set visibility for with (for example) **#pragma GCC visibility push(hidden)** and **#pragma GCC visibility pop**. Bear in mind that symbol visibility should be viewed **as part of the API interface contract** and thus all new code should always specify visibility when it is not the default ie; declarations only for use within the local DSO should **always** be marked explicitly as hidden as so to avoid PLT indirection overheads−−−making this abundantly clear also aids readability and self-documentation of the code. Note that due to ISO C++ specification requirements, operator new and operator delete must always be of default visibility.

An overview of these techniques, their benefits and how to use them is at <**http://gcc.gnu.org/wiki/Visibility**>.

ENVIRONMENT

This section describes several environment variables that affect how GCC operates. Some of them work by specifying directories or prefixes to use when searching for various kinds of files. Some are used to specify other aspects of the compilation environment.

Note that you can also specify places to search using options such as **−B**, **−I** and **−L**. These take precedence over places specified using environment variables, which in turn take precedence over those specified by the configuration of GCC.

LANG
LC_CTYPE
LC_MESSAGES
LC_ALL

These environment variables control the way that GCC uses localization information that allow GCC to work with different national conventions. GCC inspects the locale categories **LC_CTYPE** and **LC_MESSAGES** if it has been configured to do so. These locale categories can be set to any value supported by your installation. A typical value is **en_GB.UTF−8** for English in the United Kingdom encoded in UTF−8.

The **LC_CTYPE** environment variable specifies character classification. GCC uses it to determine the character boundaries in a string; this is needed for some multibyte encodings that contain quote and escape characters that would otherwise be interpreted as a string end or escape.

The **LC_MESSAGES** environment variable specifies the language to use in diagnostic messages.

If the **LC_ALL** environment variable is set, it overrides the value of **LC_CTYPE** and **LC_MESSAGES**; otherwise, **LC_CTYPE** and **LC_MESSAGES** default to the value of the **LANG** environment variable. If none of these variables are set, GCC defaults to traditional C English behavior.

TMPDIR

If **TMPDIR** is set, it specifies the directory to use for temporary files. GCC uses temporary files to hold the output of one stage of compilation which is to be used as input to the next stage: for example, the output of the preprocessor, which is the input to the compiler proper.

GCC_EXEC_PREFIX

If **GCC_EXEC_PREFIX** is set, it specifies a prefix to use in the names of the subprograms executed by the compiler. No slash is added when this prefix is combined with the name of a subprogram, but you can specify a prefix that ends with a slash if you wish.

If **GCC_EXEC_PREFIX** is not set, GCC will attempt to figure out an appropriate prefix to use based on the pathname it was invoked with.

If GCC cannot find the subprogram using the specified prefix, it tries looking in the usual places for the subprogram.

The default value of **GCC_EXEC_PREFIX** is *prefix/lib/gcc/* where *prefix* is the value of `prefix` when you ran the *configure* script.

Other prefixes specified with **−B** take precedence over this prefix.

This prefix is also used for finding files such as *crt0.o* that are used for linking.

In addition, the prefix is used in an unusual way in finding the directories to search for header files. For each of the standard directories whose name normally begins with **/usr/local/lib/gcc** (more precisely, with the value of **GCC_INCLUDE_DIR**), GCC tries replacing that beginning with the specified prefix to produce an alternate directory name. Thus, with **−Bfoo/**, GCC will search *foo/bar* where it would normally search */usr/local/lib/bar*. These alternate directories are searched first; the standard directories come next.

COMPILER_PATH
The value of **COMPILER_PATH** is a colon-separated list of directories, much like **PATH**. GCC tries the directories thus specified when searching for subprograms, if it can't find the subprograms using **GCC_EXEC_PREFIX**.

LIBRARY_PATH
The value of **LIBRARY_PATH** is a colon-separated list of directories, much like **PATH**. When configured as a native compiler, GCC tries the directories thus specified when searching for special linker files, if it can't find them using **GCC_EXEC_PREFIX**. Linking using GCC also uses these directories when searching for ordinary libraries for the **−l** option (but directories specified with **−L** come first).

LANG
This variable is used to pass locale information to the compiler. One way in which this information is used is to determine the character set to be used when character literals, string literals and comments are parsed in C and C++. When the compiler is configured to allow multibyte characters, the following values for **LANG** are recognized:

C−JIS
Recognize JIS characters.

C−SJIS
Recognize SJIS characters.

C−EUCJP
Recognize EUCJP characters.

If **LANG** is not defined, or if it has some other value, then the compiler will use mblen and mbtowc as defined by the default locale to recognize and translate multibyte characters.

Some additional environments variables affect the behavior of the preprocessor.

CPATH
C_INCLUDE_PATH
CPLUS_INCLUDE_PATH
OBJC_INCLUDE_PATH
Each variable's value is a list of directories separated by a special character, much like **PATH**, in which to look for header files. The special character, `PATH_SEPARATOR`, is target-dependent and determined at GCC build time. For Microsoft Windows-based targets it is a semicolon, and for almost all other targets it is a colon.

CPATH specifies a list of directories to be searched as if specified with **−I**, but after any paths given with **−I** options on the command line. This environment variable is used regardless of which language is being preprocessed.

The remaining environment variables apply only when preprocessing the particular language indicated. Each specifies a list of directories to be searched as if specified with **−isystem**, but after any paths given with **−isystem** options on the command line.

In all these variables, an empty element instructs the compiler to search its current working directory. Empty elements can appear at the beginning or end of a path. For instance, if the value of **CPATH** is `:/special/include`, that has the same effect as **−I. −I/special/include**.

DEPENDENCIES_OUTPUT

If this variable is set, its value specifies how to output dependencies for Make based on the non-system header files processed by the compiler. System header files are ignored in the dependency output.

The value of **DEPENDENCIES_OUTPUT** can be just a file name, in which case the Make rules are written to that file, guessing the target name from the source file name. Or the value can have the form *file target*, in which case the rules are written to file *file* using *target* as the target name.

In other words, this environment variable is equivalent to combining the options **−MM** and **−MF**, with an optional **−MT** switch too.

SUNPRO_DEPENDENCIES

This variable is the same as **DEPENDENCIES_OUTPUT** (see above), except that system header files are not ignored, so it implies **−M** rather than **−MM**. However, the dependence on the main input file is omitted.

BUGS

For instructions on reporting bugs, see <**http://gcc.gnu.org/bugs.html**>.

FOOTNOTES

1. On some systems, **gcc −shared** needs to build supplementary stub code for constructors to work. On multi-libbed systems, **gcc −shared** must select the correct support libraries to link against. Failing to supply the correct flags may lead to subtle defects. Supplying them in cases where they are not necessary is innocuous.

SEE ALSO

gpl (7), *gfdl* (7), *fsf−funding* (7), *cpp* (1), *gcov* (1), *as* (1), *ld* (1), *gdb* (1), *adb* (1), *dbx* (1), *sdb* (1) and the Info entries for *gcc*, *cpp*, *as*, *ld*, *binutils* and *gdb*.

AUTHOR

See the Info entry for **gcc**, or <**http://gcc.gnu.org/onlinedocs/gcc/Contributors.html**>, for contributors to GCC.

COPYRIGHT

NAME

gcore — get core images of running process

SYNOPSIS

gcore [**-c** *corename*] *pid* [*pid ...*]

DESCRIPTION

gcore creates a core image of the specified processes suitable for use with gdb(1). By default, the core is written to the file "<progname>.core", where "⟨progname⟩" is the program name of the process corresponding to the pid that is given on the command line. This filename can be changed by supplying the **-c** *corename* argument, or setting "proc.<pid>.corename" with sysctl(8).

FILES

<progname>.core The core image.

SEE ALSO

gdb(1), sysctl(8)

HISTORY

gcore appeared in BSD 4.2, disappeared in NetBSD 1.2, and reappeared in NetBSD 2.0.

BUGS

The process is not stopped while the core file is generated, so it might not be consistent.

NAME

gcov − coverage testing tool

SYNOPSIS

gcov [**−v**|**−−version**] [**−h**|**−−help**]
[**−a**|**−−all−blocks**]
[**−b**|**−−branch−probabilities**]
[**−c**|**−−branch−counts**]
[**−n**|**−−no−output**]
[**−l**|**−−long−file−names**]
[**−p**|**−−preserve−paths**]
[**−f**|**−−function−summaries**]
[**−o**|**−−object−directory** *directory*|*file*] *sourcefile*
[**−u**|**−−unconditional−branches**]

DESCRIPTION

gcov is a test coverage program. Use it in concert with GCC to analyze your programs to help create more efficient, faster running code and to discover untested parts of your program. You can use **gcov** as a profiling tool to help discover where your optimization efforts will best affect your code. You can also use **gcov** along with the other profiling tool, **gprof**, to assess which parts of your code use the greatest amount of computing time.

Profiling tools help you analyze your code's performance. Using a profiler such as **gcov** or **gprof**, you can find out some basic performance statistics, such as:

- how often each line of code executes
- what lines of code are actually executed
- how much computing time each section of code uses

Once you know these things about how your code works when compiled, you can look at each module to see which modules should be optimized. **gcov** helps you determine where to work on optimization.

Software developers also use coverage testing in concert with testsuites, to make sure software is actually good enough for a release. Testsuites can verify that a program works as expected; a coverage program tests to see how much of the program is exercised by the testsuite. Developers can then determine what kinds of test cases need to be added to the testsuites to create both better testing and a better final product.

You should compile your code without optimization if you plan to use **gcov** because the optimization, by combining some lines of code into one function, may not give you as much information as you need to look for 'hot spots' where the code is using a great deal of computer time. Likewise, because **gcov** accumulates statistics by line (at the lowest resolution), it works best with a programming style that places only one statement on each line. If you use complicated macros that expand to loops or to other control structures, the statistics are less helpful−−−they only report on the line where the macro call appears. If your complex macros behave like functions, you can replace them with inline functions to solve this problem.

gcov creates a logfile called *sourcefile.gcov* which indicates how many times each line of a source file *sourcefile.c* has executed. You can use these logfiles along with **gprof** to aid in fine-tuning the performance of your programs. **gprof** gives timing information you can use along with the information you get from **gcov**.

gcov works only on code compiled with GCC. It is not compatible with any other profiling or test coverage mechanism.

OPTIONS

−h
−−help
Display help about using **gcov** (on the standard output), and exit without doing any further processing.

−v

−−version
Display the **gcov** version number (on the standard output), and exit without doing any further processing.

−a
−−all−blocks
Write individual execution counts for every basic block. Normally gcov outputs execution counts only for the main blocks of a line. With this option you can determine if blocks within a single line are not being executed.

−b
−−branch−probabilities
Write branch frequencies to the output file, and write branch summary info to the standard output. This option allows you to see how often each branch in your program was taken. Unconditional branches will not be shown, unless the **−u** option is given.

−c
−−branch−counts
Write branch frequencies as the number of branches taken, rather than the percentage of branches taken.

−n
−−no−output
Do not create the **gcov** output file.

−l
−−long−file−names
Create long file names for included source files. For example, if the header file *x.h* contains code, and was included in the file *a.c*, then running **gcov** on the file *a.c* will produce an output file called *a.c##x.h.gcov* instead of *x.h.gcov*. This can be useful if *x.h* is included in multiple source files. If you use the **−p** option, both the including and included file names will be complete path names.

−p
−−preserve−paths
Preserve complete path information in the names of generated *.gcov* files. Without this option, just the filename component is used. With this option, all directories are used, with / characters translated to # characters, . directory components removed and .. components renamed to ^. This is useful if sourcefiles are in several different directories. It also affects the **−l** option.

−f
−−function−summaries
Output summaries for each function in addition to the file level summary.

−o *directory|file*
−−object−directory *directory*
−−object−file *file*
Specify either the directory containing the gcov data files, or the object path name. The *.gcno*, and *.gcda* data files are searched for using this option. If a directory is specified, the data files are in that directory and named after the source file name, without its extension. If a file is specified here, the data files are named after that file, without its extension. If this option is not supplied, it defaults to the current directory.

−u
−−unconditional−branches
When branch probabilities are given, include those of unconditional branches. Unconditional branches are normally not interesting.

gcov should be run with the current directory the same as that when you invoked the compiler. Otherwise it will not be able to locate the source files. **gcov** produces files called *mangledname.gcov* in the current directory. These contain the coverage information of the source file they correspond to. One *.gcov* file is

produced for each source file containing code, which was compiled to produce the data files. The *mangledname* part of the output file name is usually simply the source file name, but can be something more complicated if the **−l** or **−p** options are given. Refer to those options for details.

The *.gcov* files contain the **:** separated fields along with program source code. The format is

```
<execution_count>:<line_number>:<source line text>
```

Additional block information may succeed each line, when requested by command line option. The *execution_count* is **−** for lines containing no code and ##### for lines which were never executed. Some lines of information at the start have *line_number* of zero.

The preamble lines are of the form

```
-:0:<tag>:<value>
```

The ordering and number of these preamble lines will be augmented as **gcov** development progresses **−−−** do not rely on them remaining unchanged. Use *tag* to locate a particular preamble line.

The additional block information is of the form

```
<tag> <information>
```

The *information* is human readable, but designed to be simple enough for machine parsing too.

When printing percentages, 0% and 100% are only printed when the values are *exactly* 0% and 100% respectively. Other values which would conventionally be rounded to 0% or 100% are instead printed as the nearest non-boundary value.

When using **gcov**, you must first compile your program with two special GCC options: **−fprofile−arcs −ftest−coverage**. This tells the compiler to generate additional information needed by gcov (basically a flow graph of the program) and also includes additional code in the object files for generating the extra profiling information needed by gcov. These additional files are placed in the directory where the object file is located.

Running the program will cause profile output to be generated. For each source file compiled with **−fprofile−arcs**, an accompanying *.gcda* file will be placed in the object file directory.

Running **gcov** with your program's source file names as arguments will now produce a listing of the code along with frequency of execution for each line. For example, if your program is called *tmp.c*, this is what you see when you use the basic **gcov** facility:

```
$ gcc -fprofile-arcs -ftest-coverage tmp.c
$ a.out
$ gcov tmp.c
90.00% of 10 source lines executed in file tmp.c
Creating tmp.c.gcov.
```

The file *tmp.c.gcov* contains output from **gcov**. Here is a sample:

```
        -:    0:Source:tmp.c
        -:    0:Graph:tmp.gcno
        -:    0:Data:tmp.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:#include <stdio.h>
        -:    2:
        -:    3:int main (void)
        1:    4:{
        1:    5:  int i, total;
        -:    6:
        1:    7:  total = 0;
        -:    8:
       11:    9:  for (i = 0; i < 10; i++)
       10:   10:    total += i;
        -:   11:
        1:   12:  if (total != 45)
    #####:   13:    printf ("Failure\n");
        -:   14:  else
        1:   15:    printf ("Success\n");
        1:   16:  return 0;
        -:   17:}
```

When you use the **−a** option, you will get individual block counts, and the output looks like this:

```
        -:    0:Source:tmp.c
        -:    0:Graph:tmp.gcno
        -:    0:Data:tmp.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:#include <stdio.h>
        -:    2:
        -:    3:int main (void)
        1:    4:{
        1:    4-block  0
        1:    5:  int i, total;
        -:    6:
        1:    7:  total = 0;
        -:    8:
       11:    9:  for (i = 0; i < 10; i++)
       11:    9-block  0
       10:   10:    total += i;
       10:   10-block  0
        -:   11:
        1:   12:  if (total != 45)
        1:   12-block  0
    #####:   13:    printf ("Failure\n");
    $$$$$:   13-block  0
        -:   14:  else
        1:   15:    printf ("Success\n");
        1:   15-block  0
        1:   16:  return 0;
        1:   16-block  0
        -:   17:}
```

In this mode, each basic block is only shown on one line — the last line of the block. A multi-line block

will only contribute to the execution count of that last line, and other lines will not be shown to contain code, unless previous blocks end on those lines. The total execution count of a line is shown and subsequent lines show the execution counts for individual blocks that end on that line. After each block, the branch and call counts of the block will be shown, if the **−b** option is given.

Because of the way GCC instruments calls, a call count can be shown after a line with no individual blocks. As you can see, line 13 contains a basic block that was not executed.

When you use the **−b** option, your output looks like this:

```
$ gcov -b tmp.c
90.00% of 10 source lines executed in file tmp.c
80.00% of 5 branches executed in file tmp.c
80.00% of 5 branches taken at least once in file tmp.c
50.00% of 2 calls executed in file tmp.c
Creating tmp.c.gcov.
```

Here is a sample of a resulting *tmp.c.gcov* file:

```
            -:    0:Source:tmp.c
            -:    0:Graph:tmp.gcno
            -:    0:Data:tmp.gcda
            -:    0:Runs:1
            -:    0:Programs:1
            -:    1:#include <stdio.h>
            -:    2:
            -:    3:int main (void)
function main called 1 returned 1 blocks executed 75%
            1:    4:{
            1:    5:  int i, total;
            -:    6:
            1:    7:  total = 0;
            -:    8:
           11:    9:  for (i = 0; i < 10; i++)
branch  0 taken 91% (fallthrough)
branch  1 taken 9%
           10:   10:    total += i;
            -:   11:
            1:   12:  if (total != 45)
branch  0 taken 0% (fallthrough)
branch  1 taken 100%
    #####:   13:    printf ("Failure\n");
call    0 never executed
            -:   14:  else
            1:   15:    printf ("Success\n");
call    0 called 1 returned 100%
            1:   16:  return 0;
            -:   17:}
```

For each function, a line is printed showing how many times the function is called, how many times it returns and what percentage of the function's blocks were executed.

For each basic block, a line is printed after the last line of the basic block describing the branch or call that ends the basic block. There can be multiple branches and calls listed for a single source line if there are multiple basic blocks that end on that line. In this case, the branches and calls are each given a number. There is no simple way to map these branches and calls back to source constructs. In general, though, the lowest numbered branch or call will correspond to the leftmost construct on the source line.

For a branch, if it was executed at least once, then a percentage indicating the number of times the branch

was taken divided by the number of times the branch was executed will be printed. Otherwise, the message ''never executed'' is printed.

For a call, if it was executed at least once, then a percentage indicating the number of times the call returned divided by the number of times the call was executed will be printed. This will usually be 100%, but may be less for functions call `exit` or `longjmp`, and thus may not return every time they are called.

The execution counts are cumulative. If the example program were executed again without removing the *.gcda* file, the count for the number of times each line in the source was executed would be added to the results of the previous run(s). This is potentially useful in several ways. For example, it could be used to accumulate data over a number of program runs as part of a test verification suite, or to provide more accurate long-term information over a large number of program runs.

The data in the *.gcda* files is saved immediately before the program exits. For each source file compiled with **−fprofile−arcs**, the profiling code first attempts to read in an existing *.gcda* file; if the file doesn't match the executable (differing number of basic block counts) it will ignore the contents of the file. It then adds in the new execution counts and finally writes the data to the file.

Using gcov with GCC Optimization

If you plan to use **gcov** to help optimize your code, you must first compile your program with two special GCC options: **−fprofile−arcs −ftest−coverage**. Aside from that, you can use any other GCC options; but if you want to prove that every single line in your program was executed, you should not compile with optimization at the same time. On some machines the optimizer can eliminate some simple code lines by combining them with other lines. For example, code like this:

```
if (a != b)
  c = 1;
else
  c = 0;
```

can be compiled into one instruction on some machines. In this case, there is no way for **gcov** to calculate separate execution counts for each line because there isn't separate code for each line. Hence the **gcov** output looks like this if you compiled the program with optimization:

```
100:   12:if (a != b)
100:   13:  c = 1;
100:   14:else
100:   15:  c = 0;
```

The output shows that this block of code, combined by optimization, executed 100 times. In one sense this result is correct, because there was only one instruction representing all four of these lines. However, the output does not indicate how many times the result was 0 and how many times the result was 1.

Inlineable functions can create unexpected line counts. Line counts are shown for the source code of the inlineable function, but what is shown depends on where the function is inlined, or if it is not inlined at all.

If the function is not inlined, the compiler must emit an out of line copy of the function, in any object file that needs it. If *fileA.o* and *fileB.o* both contain out of line bodies of a particular inlineable function, they will also both contain coverage counts for that function. When *fileA.o* and *fileB.o* are linked together, the linker will, on many systems, select one of those out of line bodies for all calls to that function, and remove or ignore the other. Unfortunately, it will not remove the coverage counters for the unused function body. Hence when instrumented, all but one use of that function will show zero counts.

If the function is inlined in several places, the block structure in each location might not be the same. For instance, a condition might now be calculable at compile time in some instances. Because the coverage of all the uses of the inline function will be shown for the same source lines, the line counts themselves might seem inconsistent.

SEE ALSO

gpl (7), *gfdl* (7), *fsf−funding* (7), *gcc* (1) and the Info entry for *gcc*.

COPYRIGHT

NAME

gdb − The GNU Debugger

SYNOPSIS

gdb [**−help**] [**−nx**] [**−q**] [**−batch**] [**−cd=***dir*] [**−f**] [**−b** *bps*] [**−tty=***dev*] [**−s** *symfile*] [**−e** *prog*] [**−se** *prog*] [**−c** *core*] [**−x** *cmds*] [**−d** *dir*] [*prog*[*core* | *procID]]*

DESCRIPTION

The purpose of a debugger such as GDB is to allow you to see what is going on ‘‘inside’’ another program while it executes—or what another program was doing at the moment it crashed.

GDB can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

You can use GDB to debug programs written in C, C++, and Modula-2. Fortran support will be added when a GNU Fortran compiler is ready.

GDB is invoked with the shell command **gdb**. Once started, it reads commands from the terminal until you tell it to exit with the GDB command **quit**. You can get online help from **gdb** itself by using the command **help**.

You can run **gdb** with no arguments or options; but the most usual way to start GDB is with one argument or two, specifying an executable program as the argument:

gdb program

You can also start with both an executable program and a core file specified:

gdb program core

You can, instead, specify a process ID as a second argument, if you want to debug a running process:

gdb program 1234

would attach GDB to process **1234** (unless you also have a file named ‘ **1234** ’; GDB does check for a core file first).

Here are some of the most frequently needed GDB commands:

break [*file***:**]*function*
Set a breakpoint at *function* (in *file*).

run [*arglist*]
Start your program (with *arglist*, if specified).

bt Backtrace: display the program stack.

print *expr*
Display the value of an expression.

c Continue running your program (after stopping, e.g. at a breakpoint).

next Execute next program line (after stopping); step *over* any function calls in the line.

edit [*file***:**]*function*
look at the program line where it is presently stopped.

list [*file***:**]*function*
type the text of the program in the vicinity of where it is presently stopped.

step Execute next program line (after stopping); step *into* any function calls in the line.

help [*name*]
Show information about GDB command *name*, or general information about using GDB.

quit Exit from GDB.

For full details on GDB, see *Using GDB: A Guide to the GNU Source-Level Debugger*, by Richard M. Stallman and Roland H. Pesch. The same text is available online as the **gdb** entry in the **info** program.

OPTIONS

Any arguments other than options specify an executable file and core file (or process ID); that is, the first argument encountered with no associated option flag is equivalent to a ‘ **−se** ’ option, and the second, if any, is equivalent to a ‘ **−c** ’ option if it’s the name of a file. Many options have both long and short forms; both are shown here. The long forms are also recognized if you truncate them, so long as enough of the option is present to be unambiguous. (If you prefer, you can flag option arguments with ‘ **+** ’ rather than ‘ **−** ’, though we illustrate the more usual convention.)

All the options and command line arguments you give are processed in sequential order. The order makes a difference when the ‘ **−x** ’ option is used.

−help

−h List all options, with brief explanations.

−symbols= *file*

−s *file* Read symbol table from file *file*.

−write Enable writing into executable and core files.

−exec= *file*

−e *file* Use file *file* as the executable file to execute when appropriate, and for examining pure data in conjunction with a core dump.

−se= *file*
Read symbol table from file *file* and use it as the executable file.

−core= *file*

−c *file* Use file *file* as a core dump to examine.

−command= *file*

−x *file* Execute GDB commands from file *file*.

−directory=*directory*

−d *directory*
Add *directory* to the path to search for source files.

−nx

−n Do not execute commands from any ‘ **.gdbinit** ’ initialization files. Normally, the commands in these files are executed after all the command options and arguments have been processed.

−quiet

−q ‘‘Quiet’’. Do not print the introductory and copyright messages. These messages are also suppressed in batch mode.

−batch Run in batch mode. Exit with status **0** after processing all the command files specified with ‘ **−x** ’ (and ‘ **.gdbinit** ’, if not inhibited). Exit with nonzero status if an error occurs in executing the GDB commands in the command files.

Batch mode may be useful for running GDB as a filter, for example to download and run a program on another computer; in order to make this more useful, the message

Program exited normally.

(which is ordinarily issued whenever a program running under GDB control terminates) is not issued when running in batch mode.

−cd=*directory*
Run GDB using *directory* as its working directory, instead of the current directory.

−fullname

−f Emacs sets this option when it runs GDB as a subprocess. It tells GDB to output the full file name and line number in a standard, recognizable fashion each time a stack frame is displayed (which includes each time the program stops). This recognizable format looks like two ‘ **32** ’ characters, followed by the file name, line number and character position separated by colons, and a newline. The Emacs-to-GDB interface program uses the two ‘ **32** ’ characters as a signal to display the source code for the frame.

−b *bps* Set the line speed (baud rate or bits per second) of any serial interface used by GDB for remote debugging.

−tty=*device*
Run using *device* for your program’s standard input and output.

SEE ALSO

‘ **gdb** ’ entry in **info**; *Using GDB: A Guide to the GNU Source-Level Debugger*, Richard M. Stallman and Roland H. Pesch, July 1991.

COPYING

NAME

gdbserver − Remote Server for the GNU Debugger

SYNOPSIS

gdbserver

tty prog [**args...**]

gdbserver tty **--attach** PID

DESCRIPTION

GDBSERVER is a program that allows you to run GDB on a different machine than the one which is running the program being debugged.

Usage (server (target) side):

First, you need to have a copy of the program you want to debug put onto the target system. The program can be stripped to save space if needed, as GDBserver doesn't care about symbols. All symbol handling is taken care of by the GDB running on the host system.

To use the server, you log on to the target system, and run the 'gdbserver' program. You must tell it (a) how to communicate with GDB, (b) the name of your program, and (c) its arguments. The general syntax is:

```
target> gdbserver COMM PROGRAM [ARGS ...]
```

For example, using a serial port, you might say:

```
target> gdbserver /dev/com1 emacs foo.txt
```

This tells gdbserver to debug emacs with an argument of foo.txt, and to communicate with GDB via /dev/com1. Gdbserver now waits patiently for the host GDB to communicate with it.

To use a TCP connection, you could say:

```
target> gdbserver host:2345 emacs foo.txt
```

This says pretty much the same thing as the last example, except that we are going to communicate with the host GDB via TCP. The 'host:2345' argument means that we are expecting to see a TCP connection from 'host' to local TCP port 2345. (Currently, the 'host' part is ignored.) You can choose any number you want for the port number as long as it does not conflict with any existing TCP ports on the target system. This same port number must be used in the host GDBs 'target remote' command, which will be described shortly. Note that if you chose a port number that conflicts with another service, gdbserver will print an error message and exit.

On some targets, gdbserver can also attach to running programs. This is accomplished via the --attach argument. The syntax is:

```
target> gdbserver COMM --attach PID
```

PID is the process ID of a currently running process. It isn't necessary to point gdbserver at a binary for the running process.

Usage (host side):

You need an unstripped copy of the target program on your host system, since GDB needs to examine it's symbol tables and such. Start up GDB as you normally would, with the target program as the first

argument. (You may need to use the --baud option if the serial line is running at anything except 9600 baud.) Ie: ‘gdb TARGET-PROG’, or ‘gdb --baud BAUD TARGET-PROG’. After that, the only new command you need to know about is ‘target remote’. It’s argument is either a device name (usually a serial device, like ‘/dev/ttyb’), or a HOST:PORT descriptor. For example:

```
(gdb) target remote /dev/ttyb
```

communicates with the server via serial line /dev/ttyb, and:

```
(gdb) target remote the-target:2345
```

communicates via a TCP connection to port 2345 on host ‘the-target’, where you previously started up gdbserver with the same port number. Note that for TCP connections, you must start up gdbserver prior to using the ‘target remote’ command, otherwise you may get an error that looks something like ‘Connection refused’.

OPTIONS

You have to supply the name of the program to debug and the tty to communicate on; the remote GDB will do everything else. Any remaining arguments will be passed to the program verbatim.

SEE ALSO

‘ **gdb** ’ entry in **info**; *Using GDB: A Guide to the GNU Source-Level Debugger*, Richard M. Stallman and Roland H. Pesch, July 1991.

COPYING

NAME

genassym — emit an assym.h file

SYNOPSIS

genassym [**-c**] [**-f**] *C compiler invocation*

DESCRIPTION

genassym is a shell script normally used during the kernel build process to create an assym.h file. This file defines a number of cpp constants derived from the configuration information **genassym** reads from stdin. The generated file is used by kernel sources written in assembler to gain access to information (e.g. structure offsets and sizes) normally only known to the C compiler. Arguments to **genassym** are usually of the form *${CC} ${CFLAGS} ${CPPFLAGS}* where *${CC}* is the C compiler used to compile the kernel, while *${CFLAGS}* and *${CPPFLAGS}* are flag arguments to the C compiler. The script creates a C source file from its input. Then the C compiler is called according to the script's arguments to compile this file. Normally **genassym** instructs the C compiler to create an assembler source from the constructed C source. The resulting file is then processed to extract the information needed to create the assym.h file. The **-c** flag instructs **genassym** to create slightly different code, generate an executable from this code and run it. In both cases the assym.h file is written to stdout. The **-f** flag instructs **genassym** to create forth code.

DIAGNOSTICS

Either self-explanatory, or generated by one of the programs called from the script.

SEE ALSO

`genassym.cf`(5)

HISTORY

The **genassym** command appeared in NetBSD 1.3 as genassym.sh in `/usr/src/sys/kern`. It became a userland utility in NetBSD 4.0.

NAME

gencat — generates a Native Language Support (NLS) message catalog file

SYNOPSIS

gencat [**-?**] *catalog-file* *message-file* [*message-file ...*]

DESCRIPTION

The **gencat** command reads one or more files containing message strings that will be displayed using the catgets(3) library call. From these files it generates a message catalog which is loaded dynamically by the Native Language Support (NLS) library at run time.

The message description files are text files in the format described below.

The message catalog file is a binary file. If it already exists, it will be truncated when **gencat** is run.

Error messages are grouped into sets, and a program can load a particular set depending on which type, or language, of messages is desired.

The **-?** option flag prints the usage message.

MESSAGE FILE FORMAT

Empty lines and leading blanks are ignored.

$set NN Determines the set to be used for all subsequent messages. *NN* is an integer greater than 0.

$delset NN Removes a set from the catalog. *NN* is an integer greater than 0.

If a set was created earlier in the current file, or in a file previously read by the **gencat** command, this command will remove it.

$quote C Sets a quote character to be used around the messages. *C* may be any character other than white space.

If this is specified, then messages must begin and end with the quote character. By default no quote character is used. If none is specified, then the current quote character is unset. This is useful when messages must contain leading white space.

NN message Defines a message. *NN* is an integer greater than 0.

The message is read until the end of the line or a quote character (if one is specified). If no message is provided, the message with the number *NN* is removed from the catalog. If no "set" has been created, this command generates an error.

Messages may contain any characters, however the "\" is special as an escape character, where the following instances are allowed:

```
\\      Generates a single backslash.
\n      Generates a newline (as defined by the C compiler).
\t      Generates a tab (as defined by the C compiler).
\v      Generates a vertical tab (as defined by the C compiler).
\b      Generates a backspace (as defined by the C compiler).
\r      Generates a carriage return (as defined by the C compiler).
\f      Generates a form feed (as defined by the C compiler).
\NNN    Generates the character corresponding to the specified
        octal number.
\EOL    A backslash at the end of line continues the message onto
        the next line.
\quote  A backslash preceding the current quote character generates
```

the quote character.

SEE ALSO

catclose(3), catgets(3), catopen(3), nls(7)

AUTHORS

The Native Language Support (NLS) message catalog facility was contributed by J.T. Conklin ⟨jtc@NetBSD.org⟩. This page was written by
Kee Hinckley ⟨nazgul@somewhere.com⟩.

NAME

getconf — get configuration values

SYNOPSIS

getconf *system_var*
getconf -a
getconf *path_var pathname*
getconf -a *pathname*

DESCRIPTION

The **getconf** utility writes the current value of a configurable system limit or option variable to the standard output.

The *system_var* argument specifies the system variable to be queried. The names of the system variables are from sysconf(3) with the leading "_SC_" removed.

The *path_var* argument specifies the pathname variable to be queried for the specified *pathname* argument. The names of the pathname variables are from pathconf(2) with the leading "_PC_" removed.

When invoked with the option **-a**, **getconf** writes a list of all applicable variables and their values to the standard output, in the format "*name* = *value*".

EXIT STATUS

The **getconf** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

pathconf(2), confstr(3), sysconf(3)

STANDARDS

The **getconf** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2").

NAME

getent — get entries from administrative database

SYNOPSIS

getent *database* [*key ...*]

DESCRIPTION

The **getent** program retrieves and displays entries from the administrative database specified by *database*, using the lookup order specified in nsswitch.conf(5). The display format for a given *database* is as per the "traditional" file format for that database.

database may be one of:

Database	Display format
disktab	entry
ethers	address name
gettytab	entry
group	group:passwd:gid:[member[,member]...]
hosts	address name [alias ...]
netgroup	(host,user,domain) [...]
networks	name network [alias ...]
passwd	user:passwd:uid:gid:gecos:home_dir:shell
printcap	entry
protocols	name protocol [alias ...]
rpc	name number [alias ...]
services	name port/protocol [alias ...]
shells	/path/to/shell
termcap	entry

If one or more *key* arguments are provided, they will be looked up in *database* using the appropriate function. For example, **passwd** supports a numeric UID or user name; **hosts** supports an IPv4 address, IPv6 address, or host name; and **services** supports a service name, service name/protocol name, numeric port, or numeric port/protocol name.

If no *key* is provided and *database* supports enumeration, all entries for *database* will be retrieved using the appropriate enumeration function and printed.

For cgetcap(3) style databases (**disktab**, **printcap**, **termcap**) specifying a key, lists the entry for that key, and specifying more arguments after the key are used as fields in that key, and only the values of the keys are returned. For boolean keys true is returned if the key is found. If a key is not found, then false is always returned.

DIAGNOSTICS

getent exits 0 on success, 1 if there was an error in the command syntax, 2 if one of the specified key names was not found in *database*, or 3 if there is no support for enumeration on *database*.

SEE ALSO

cgetcap(3), disktab(5), ethers(5), gettytab(5), group(5), hosts(5), networks(5), nsswitch.conf(5), passwd(5), protocols(5), printcap(5), rpc(5), services(5), shells(5), termcap(5)

HISTORY

A **getent** command appeared in NetBSD 3.0. It was based on the command of the same name in Solaris and Linux.

NAME

getextattr, **lsextattr**, **rmextattr**, **setextattr** — manipulate extended attributes

SYNOPSIS

getextattr [**-fhqsx**] *attrnamespace attrname filename ...*
lsextattr [**-fhq**] *attrnamespace filename ...*
rmextattr [**-fhq**] *attrnamespace attrname filename ...*
setextattr [**-fhnq**] *attrnamespace attrname attrvalue filename ...*

DESCRIPTION

These utilities are user tools to manipulate the named extended attributes on files and directories. The *attrnamespace* argument should be the namespace of the attribute to retrieve: legal values are **user** and **system**. The *attrname* argument should be the name of the attribute, *filename* the name of the target file or directory, *attrvalue* a string to store in the attribute.

The following options are available:

-f (Force.) Ignore errors on individual filenames and continue with the remaining arguments.

-h (No follow.) If the file is a symbolic link, perform the operation on the link itself rather than the file that the link points to.

-n (NUL-terminate.) NUL-terminate the extent content written out.

-q (Quiet.) Do not print out the pathname and suppress error messages.

-s (Stringify.) Escape nonprinting characters and put quotes around the output.

-x (Hex.) Print the output in hexadecimal.

EXAMPLES

```
setextattr system md5 ‘md5 -q /boot/kernel/kernel‘ /boot/kernel/kernel
getextattr system md5 /boot/kernel/kernel
lsextattr system /boot/kernel/kernel
rmextattr system md5 /boot/kernel/kernel
```

SEE ALSO

extattr(2), extattr(3), extattrctl(8), extattr(9)

HISTORY

Extended attribute support was developed as part of the TrustedBSD Project, and introduced in FreeBSD 5.0 and NetBSD 3.0. It was developed to support security extensions requiring additional labels to be associated with each file or directory.

AUTHORS

Robert N M Watson
Poul-Henning Kamp

BUGS

The **setextattr** utility can only be used to set attributes to strings.

NAME

getopt — parse command options

SYNOPSIS

```
args=`getopt optstring $*`
set -- `getopt optstring $*`
```

DESCRIPTION

getopt is used to break up options in command lines for easy parsing by shell procedures, and to check for legal options. [Optstring] is a string of recognized option letters (see getopt(3)); if a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by white space. The special option "--" is used to delimit the end of the options. **getopt** will place "--" in the arguments at the end of the options, or recognize it if used explicitly. The shell arguments (**$1 $2** ...) are reset so that each option is preceded by a "-" and in its own shell argument; each option argument is also in its own shell argument.

EXAMPLES

The following code fragment shows how one might process the arguments for a command that can take the options [a] and [b], and the option [c], which requires an argument.

```
args=`getopt abc: $*`
if [ $? -ne 0 ]; then
        echo 'Usage: ...'
        exit 2
fi
set -- $args
while [ $# -gt 0 ]; do
        case "$1" in
                -a|-b)
                        flag=$1
                        ;;
                -c)
                        carg=$2; shift
                        ;;
                --)
                        shift; break
                        ;;
        esac
        shift
done
```

This code will accept any of the following as equivalent:

```
cmd -acarg file file
cmd -a -c arg file file
cmd -carg -a file file
cmd -a -carg -- file file
```

IEEE Std 1003.2 ("POSIX.2") mandates that the sh(1) set command return the value of 0 for the exit status. Therefore, the exit status of the **getopt** command is lost when **getopt** and the sh(1) set command are used on the same line. The example given is one way to detect errors found by **getopt**.

DIAGNOSTICS

getopt prints an error message on the standard error output when it encounters an option letter not included in [optstring].

SEE ALSO

sh(1), getopt(3)

HISTORY

Written by Henry Spencer, working from a Bell Labs manual page. Behavior believed identical to the Bell version.

BUGS

Whatever getopt(3) has.

Arguments containing white space or embedded shell metacharacters generally will not survive intact; this looks easy to fix but isn't.

The error message for an invalid option is identified as coming from **getopt** rather than from the shell procedure containing the invocation of **getopt**; this again is hard to fix.

The precise best way to use the **set** command to set the arguments without disrupting the value(s) of shell options varies from one shell version to another.

NAME

gettextize − install or upgrade gettext infrastructure

SYNOPSIS

gettextize [*OPTION*]... [*package-dir*]

DESCRIPTION

Prepares a source package to use gettext.

OPTIONS

−−help print this help and exit

−−version
print version information and exit

−c, **−−copy**
copy files instead of making symlinks

−f, **−−force**
force writing of new files even if old exist

−−intl install libintl in a subdirectory

−−no−changelog
don’t update or create ChangeLog files

−n, **−−dry−run**
print modifications but don’t perform them

AUTHOR

Written by Ulrich Drepper

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **gettextize** is maintained as a Texinfo manual. If the **info** and **gettextize** programs are properly installed at your site, the command

info gettextize

should give you access to the complete manual.

NAME

gprof − display call graph profile data

SYNOPSIS

gprof [−[abcDhilLrsTvwxyz]] [−[ACeEfFJnNOpPqQZ][*name*]]
[−I *dirs*] [−d[*num*]] [−k *from/to*]
[−m *min-count*] [−R *map_file*] [−t *table-length*]
[−−[no−]annotated−source[=*name*]]
[−−[no−]exec−counts[=*name*]]
[−−[no−]flat−profile[=*name*]] [−−[no−]graph[=*name*]]
[−−[no−]time=*name*] [−−all−lines] [−−brief]
[−−debug[=*level*]] [−−function−ordering]
[−−file−ordering] [−−directory−path=*dirs*]
[−−display−unused−functions] [−−file−format=*name*]
[−−file−info] [−−help] [−−line] [−−min−count=*n*]
[−−no−static] [−−print−path] [−−separate−files]
[−−static−call−graph] [−−sum] [−−table−length=*len*]
[−−traditional] [−−version] [−−width=*n*]
[−−ignore−non−functions] [−−demangle[=*STYLE*]]
[−−no−demangle] [*image-file*] [*profile-file ...*]

DESCRIPTION

`gprof` produces an execution profile of C, Pascal, or Fortran77 programs. The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file (*gmon.out* default) which is created by programs that are compiled with the **−pg** option of `cc`, `pc`, and `f77`. The **−pg** option also links in versions of the library routines that are compiled for profiling. `Gprof` reads the given object file (the default is `a.out`) and establishes the relation between its symbol table and the call graph profile from *gmon.out*. If more than one profile file is specified, the `gprof` output shows the sum of the profile information in the given profile files.

`Gprof` calculates the amount of time spent in each routine. Next, these times are propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle.

Several forms of output are available from the analysis.

The *flat profile* shows how much time your program spent in each function, and how many times that function was called. If you simply want to know which functions burn most of the cycles, it is stated concisely here.

The *call graph* shows, for each function, which functions called it, which other functions it called, and how many times. There is also an estimate of how much time was spent in the subroutines of each function. This can suggest places where you might try to eliminate function calls that use a lot of time.

The *annotated source* listing is a copy of the program's source code, labeled with the number of times each line of the program was executed.

OPTIONS

These options specify which of several output formats `gprof` should produce.

Many of these options take an optional *symspec* to specify functions to be included or excluded. These options can be specified multiple times, with different symspecs, to include or exclude sets of symbols.

Specifying any of these options overrides the default (**−p −q**), which prints a flat profile and call graph analysis for all functions.

`−A[`*`symspec`*`]`
`−−annotated−source[=`*`symspec`*`]`

The **−A** option causes `gprof` to print annotated source code. If *symspec* is specified, print output only for matching symbols.

`-b`
`--brief`
: If the **-b** option is given, `gprof` doesn't print the verbose blurbs that try to explain the meaning of all of the fields in the tables. This is useful if you intend to print out the output, or are tired of seeing the blurbs.

`-C[`*`symspec`*`]`
`--exec-counts[=`*`symspec`*`]`
: The **-C** option causes `gprof` to print a tally of functions and the number of times each was called. If *symspec* is specified, print tally only for matching symbols.

  If the profile data file contains basic-block count records, specifying the **-l** option, along with **-C**, will cause basic-block execution counts to be tallied and displayed.

`-i`
`--file-info`
: The **-i** option causes `gprof` to display summary information about the profile data file(s) and then exit. The number of histogram, call graph, and basic-block count records is displayed.

`-I` *`dirs`*
`--directory-path=`*`dirs`*
: The **-I** option specifies a list of search directories in which to find source files. Environment variable *GPROF_PATH* can also be used to convey this information. Used mostly for annotated source output.

`-J[`*`symspec`*`]`
`--no-annotated-source[=`*`symspec`*`]`
: The **-J** option causes `gprof` not to print annotated source code. If *symspec* is specified, `gprof` prints annotated source, but excludes matching symbols.

`-L`
`--print-path`
: Normally, source filenames are printed with the path component suppressed. The **-L** option causes `gprof` to print the full pathname of source filenames, which is determined from symbolic debugging information in the image file and is relative to the directory in which the compiler was invoked.

`-p[`*`symspec`*`]`
`--flat-profile[=`*`symspec`*`]`
: The **-p** option causes `gprof` to print a flat profile. If *symspec* is specified, print flat profile only for matching symbols.

`-P[`*`symspec`*`]`
`--no-flat-profile[=`*`symspec`*`]`
: The **-P** option causes `gprof` to suppress printing a flat profile. If *symspec* is specified, `gprof` prints a flat profile, but excludes matching symbols.

`-q[`*`symspec`*`]`
`--graph[=`*`symspec`*`]`
: The **-q** option causes `gprof` to print the call graph analysis. If *symspec* is specified, print call graph only for matching symbols and their children.

`-Q[`*`symspec`*`]`
`--no-graph[=`*`symspec`*`]`
: The **-Q** option causes `gprof` to suppress printing the call graph. If *symspec* is specified, `gprof` prints a call graph, but excludes matching symbols.

`-t`
`--table-length=`*`num`*
: The **-t** option causes the *num* most active source lines in each source file to be listed when source annotation is enabled. The default is 10.

`-y`

`--separate-files`
This option affects annotated source output only. Normally, `gprof` prints annotated source files to standard-output. If this option is specified, annotated source for a file named *path/filename* is generated in the file *filename-ann*. If the underlying filesystem would truncate *filename-ann* so that it overwrites the original *filename*, `gprof` generates annotated source in the file *filename.ann* instead (if the original file name has an extension, that extension is *replaced* with *.ann*).

`-Z[`*`symspec`*`]`
`--no-exec-counts[=`*`symspec`*`]`
The **-Z** option causes `gprof` not to print a tally of functions and the number of times each was called. If *symspec* is specified, print tally, but exclude matching symbols.

`-r`
`--function-ordering`
The **--function-ordering** option causes `gprof` to print a suggested function ordering for the program based on profiling data. This option suggests an ordering which may improve paging, tlb and cache behavior for the program on systems which support arbitrary ordering of functions in an executable.

The exact details of how to force the linker to place functions in a particular order is system dependent and out of the scope of this manual.

`-R` *`map_file`*
`--file-ordering` *`map_file`*
The **--file-ordering** option causes `gprof` to print a suggested .o link line ordering for the program based on profiling data. This option suggests an ordering which may improve paging, tlb and cache behavior for the program on systems which do not support arbitrary ordering of functions in an executable.

Use of the **-a** argument is highly recommended with this option.

The *map_file* argument is a pathname to a file which provides function name to object file mappings. The format of the file is similar to the output of the program `nm`.

```
c-parse.o:00000000 T yyparse
c-parse.o:00000004 C yyerrflag
c-lang.o:00000000 T maybe_objc_method_name
c-lang.o:00000000 T print_lang_statistics
c-lang.o:00000000 T recognize_objc_keyword
c-decl.o:00000000 T print_lang_identifier
c-decl.o:00000000 T print_lang_type
...
```

To create a *map_file* with GNU `nm`, type a command like `nm --extern-only --defined-only -v --print-file-name program-name`.

`-T`
`--traditional`
The **-T** option causes `gprof` to print its output in ''traditional'' BSD style.

`-w` *`width`*
`--width=`*`width`*
Sets width of output lines to *width*. Currently only used when printing the function index at the bottom of the call graph.

`-x`
`--all-lines`
This option affects annotated source output only. By default, only the lines at the beginning of a basic-block are annotated. If this option is specified, every line in a basic-block is annotated by repeating the annotation for the first line. This behavior is similar to `tcov`'s **-a**.

`--demangle[=`*`style`*`]`
`--no-demangle`
These options control whether C++ symbol names should be demangled when printing output. The default is to demangle symbols. The `--no-demangle` option may be used to turn off demangling. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

Analysis Options

`-a`
`--no-static`
The **-a** option causes `gprof` to suppress the printing of statically declared (private) functions. (These are functions whose names are not listed as global, and which are not visible outside the file/function/block where they were defined.) Time spent in these functions, calls to/from them, etc, will all be attributed to the function that was loaded directly before it in the executable file. This option affects both the flat profile and the call graph.

`-c`
`--static-call-graph`
The **-c** option causes the call graph of the program to be augmented by a heuristic which examines the text space of the object file and identifies function calls in the binary machine code. Since normal call graph records are only generated when functions are entered, this option identifies children that could have been called, but never were. Calls to functions that were not compiled with profiling enabled are also identified, but only if symbol table entries are present for them. Calls to dynamic library routines are typically *not* found by this option. Parents or children identified via this heuristic are indicated in the call graph with call counts of **0**.

`-D`
`--ignore-non-functions`
The **-D** option causes `gprof` to ignore symbols which are not known to be functions. This option will give more accurate profile data on systems where it is supported (Solaris and HPUX for example).

`-k` *`from/to`*
The **-k** option allows you to delete from the call graph any arcs from symbols matching symspec *from* to those matching symspec *to*.

`-l`
`--line`
The **-l** option enables line-by-line profiling, which causes histogram hits to be charged to individual source code lines, instead of functions. If the program was compiled with basic-block counting enabled, this option will also identify how many times each line of code was executed. While line-by-line profiling can help isolate where in a large function a program is spending its time, it also significantly increases the running time of `gprof`, and magnifies statistical inaccuracies.

`-m` *`num`*
`--min-count=`*`num`*
This option affects execution count output only. Symbols that are executed less than *num* times are suppressed.

`-n[`*`symspec`*`]`
`--time[=`*`symspec`*`]`
The **-n** option causes `gprof`, in its call graph analysis, to only propagate times for symbols matching *symspec*.

`-N[`*`symspec`*`]`
`--no-time[=`*`symspec`*`]`
The **-n** option causes `gprof`, in its call graph analysis, not to propagate times for symbols matching *symspec*.

`-z`
`--display-unused-functions`
If you give the **-z** option, `gprof` will mention all functions in the flat profile, even those that were never called, and that had no time spent in them. This is useful in conjunction with the **-c** option for discovering which routines were never called.

Miscellaneous Options

`-d[`*num*`]`
`--debug[=`*num*`]`
The **-d** *num* option specifies debugging options. If *num* is not specified, enable all debugging.

`-h`
`--help`
The **-h** option prints command line usage.

`-O`*name*
`--file-format=`*name*
Selects the format of the profile data files. Recognized formats are **auto** (the default), **bsd**, **4.4bsd**, **magic**, and **prof** (not yet supported).

`-s`
`--sum`
The **-s** option causes `gprof` to summarize the information in the profile data files it read in, and write out a profile data file called *gmon.sum*, which contains all the information from the profile data files that `gprof` read in. The file *gmon.sum* may be one of the specified input files; the effect of this is to merge the data in the other input files into *gmon.sum*.

Eventually you can run `gprof` again without **-s** to analyze the cumulative data in the file *gmon.sum*.

`-v`
`--version`
The **-v** flag causes `gprof` to print the current version number, and then exit.

Deprecated Options

These options have been replaced with newer versions that use symspecs.

`-e` *function_name*
The **-e** *function* option tells `gprof` to not print information about the function *function_name* (and its children...) in the call graph. The function will still be listed as a child of any functions that call it, but its index number will be shown as **[not printed]**. More than one **-e** option may be given; only one *function_name* may be indicated with each **-e** option.

`-E` *function_name*
The `-E` *function* option works like the `-e` option, but time spent in the function (and children who were not called from anywhere else), will not be used to compute the percentages-of-time for the call graph. More than one **-E** option may be given; only one *function_name* may be indicated with each **-E** option.

`-f` *function_name*
The **-f** *function* option causes `gprof` to limit the call graph to the function *function_name* and its children (and their children...). More than one **-f** option may be given; only one *function_name* may be indicated with each **-f** option.

`-F` *function_name*
The **-F** *function* option works like the `-f` option, but only time spent in the function and its children (and their children...) will be used to determine total-time and percentages-of-time for the call graph. More than one **-F** option may be given; only one *function_name* may be indicated with each **-F** option. The **-F** option overrides the **-E** option.

FILES

a.out
: the namelist and text space.

gmon.out
: dynamic call graph and profile.

gmon.sum
: summarized dynamic call graph and profile.

BUGS

The granularity of the sampling is shown, but remains statistical at best. We assume that the time for each execution of a function can be expressed by the total time for the function divided by the number of times the function is called. Thus the time propagated along the call graph arcs to the function's parents is directly proportional to the number of times that arc is traversed.

Parents that are not themselves profiled will have the time of their profiled children propagated to them, but they will appear to be spontaneously invoked in the call graph listing, and will not have their time propagated further. Similarly, signal catchers, even though profiled, will appear to be spontaneous (although for more obscure reasons). Any profiled children of signal catchers should have their times propagated properly, unless the signal catcher was invoked during the execution of the profiling routine, in which case all is lost.

The profiled program must call `exit`(2) or return normally for the profiling information to be saved in the *gmon.out* file.

SEE ALSO

monitor (3), *profil* (2), *cc* (1), *prof* (1), and the Info entry for *gprof*.

''An Execution Profiler for Modular Programs'', by S. Graham, P. Kessler, M. McKusick; Software − Practice and Experience, Vol. 13, pp. 671−685, 1983.

''gprof: A Call Graph Execution Profiler'', by S. Graham, P. Kessler, M. McKusick; Proceedings of the SIGPLAN '82 Symposium on Compiler Construction, SIGPLAN Notices, Vol. 17, No 6, pp. 120−126, June 1982.

COPYRIGHT

NAME

grep, egrep, fgrep − print lines matching a pattern

SYNOPSIS

grep [*options*] *PATTERN* [*FILE*...]
grep [*options*] [**−e** *PATTERN* | **−f** *FILE*] [*FILE*...]

DESCRIPTION

Grep searches the named input *FILE*s (or standard input if no files are named, or the file name − is given) for lines containing a match to the given *PATTERN*. By default, **grep** prints the matching lines.

In addition, two variant programs **egrep** and **fgrep** are available. **Egrep** is the same as **grep −E**. **Fgrep** is the same as **grep −F**.

OPTIONS

−A *NUM*, **−−after-context=***NUM*
Print *NUM* lines of trailing context after matching lines. Places a line containing −− between contiguous groups of matches.

−a, **−−text**
Process a binary file as if it were text; this is equivalent to the **−−binary-files=text** option.

−B *NUM*, **−−before-context=***NUM*
Print *NUM* lines of leading context before matching lines. Places a line containing −− between contiguous groups of matches.

−b, **−−byte-offset**
Print the byte offset within the input file before each line of output.

−−binary-files=*TYPE*
If the first few bytes of a file indicate that the file contains binary data, assume that the file is of type *TYPE*. By default, *TYPE* is **binary**, and **grep** normally outputs either a one-line message saying that a binary file matches, or no message if there is no match. If *TYPE* is **without-match**, **grep** assumes that a binary file does not match; this is equivalent to the **−I** option. If *TYPE* is **text**, **grep** processes a binary file as if it were text; this is equivalent to the **−a** option. *Warning:* **grep −−binary-files=text** might output binary garbage, which can have nasty side effects if the output is a terminal and if the terminal driver interprets some of it as commands.

−C *NUM*, **−−context=***NUM*
Print *NUM* lines of output context. Places a line containing −− between contiguous groups of matches.

−c, **−−count**
Suppress normal output; instead print a count of matching lines for each input file. With the **−v**, **−−invert-match** option (see below), count non-matching lines.

−−colour[=*WHEN*], **−−color[=***WHEN*]
Surround the matching string with the marker find in **GREP_COLOR** environment variable. WHEN may be ‘never’, ‘always’, or ‘auto’

−D *ACTION*, **−−devices=***ACTION*
If an input file is a device, FIFO or socket, use *ACTION* to process it. By default, *ACTION* is **read**, which means that devices are read just as if they were ordinary files. If *ACTION* is **skip**, devices are silently skipped.

−d *ACTION*, **−−directories=***ACTION*
If an input file is a directory, use *ACTION* to process it. By default, *ACTION* is **read**, which means that directories are read just as if they were ordinary files. If *ACTION* is **skip**, directories are silently skipped. If *ACTION* is **recurse**, **grep** reads all files under each directory, recursively; this is equivalent to the **−r** option.

−E, **−−extended-regexp**
Interpret *PATTERN* as an extended regular expression (see below).

−e *PATTERN*, **−−regexp=***PATTERN*
Use *PATTERN* as the pattern; useful to protect patterns beginning with **−**. May be specified more than once.

−F, **−−fixed-strings**
Interpret *PATTERN* as a list of fixed strings, separated by newlines, any of which is to be matched.

−f *FILE*, **−−file=***FILE*
Obtain patterns from *FILE*, one per line. The empty file contains zero patterns, and therefore matches nothing.

−G, **−−basic-regexp**
Interpret *PATTERN* as a basic regular expression (see below). This is the default.

−H, **−−with-filename**
Print the filename for each match.

−h, **−−no-filename**
Suppress the prefixing of filenames on output when multiple files are searched.

−−help Output a brief help message.

−I Process a binary file as if it did not contain matching data; this is equivalent to the **−−binary-files=without-match** option.

−i, **−−ignore-case**
Ignore case distinctions in both the *PATTERN* and the input files.

−L, **−−files-without-match**
Suppress normal output; instead print the name of each input file from which no output would normally have been printed. The scanning will stop on the first match.

−l, **−−files-with-matches**
Suppress normal output; instead print the name of each input file from which output would normally have been printed. The scanning will stop on the first match.

−−label=*LABEL*
Displays input actually coming from standard input as input coming from file *LABEL*. This is especially useful for tools like zgrep, e.g. **gzip -cd foo.gz |grep --label=foo something**

−−line-buffered
Use line buffering, it can be a performance penalty.

−m *NUM*, **−−max-count=***NUM*
Stop reading a file after *NUM* matching lines. If the input is standard input from a regular file, and *NUM* matching lines are output, **grep** ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When **grep** stops after *NUM* matching lines, it outputs any trailing context lines. When the **−c** or **−−count** option is also used, **grep** does not output a count greater than *NUM*. When the **−v** or **−−invert-match** option is also used, **grep** stops after outputting *NUM* non-matching lines.

−−mmap
If possible, use the **mmap**(2) system call to read input, instead of the default **read**(2) system call. In some situations, **−−mmap** yields better performance. However, **−−mmap** can cause undefined behavior (including core dumps) if an input file shrinks while **grep** is operating, or if an I/O error occurs.

−n, **−−line-number**
Prefix each line of output with the line number within its input file.

−o, **−−only-matching**
Show only the part of a matching line that matches *PATTERN*.

−P, **−−perl-regexp**
Interpret *PATTERN* as a Perl regular expression.

−q, **−−quiet**, **−−silent**
Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected. Also see the **−s** or **−−no-messages** option.

−R, **−r**, **−−recursive**
Read all files under each directory, recursively; this is equivalent to the **−d recurse** option.

−−include=PATTERN
Recurse in directories only searching file matching *PATTERN*.

−−exclude=PATTERN
Recurse in directories skip file matching *PATTERN*.

−s, **−−no-messages**
Suppress error messages about nonexistent or unreadable files. Portability note: unlike GNU **grep**, traditional **grep** did not conform to POSIX.2, because traditional **grep** lacked a **−q** option and its **−s** option behaved like GNU **grep**'s **−q** option. Shell scripts intended to be portable to traditional **grep** should avoid both **−q** and **−s** and should redirect output to /dev/null instead.

−U, **−−binary**
Treat the file(s) as binary. By default, under MS-DOS and MS-Windows, **grep** guesses the file type by looking at the contents of the first 32KB read from the file. If **grep** decides the file is a text file, it strips the CR characters from the original file contents (to make regular expressions with **^** and **$** work correctly). Specifying **−U** overrules this guesswork, causing all files to be read and passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some regular expressions to fail. This option has no effect on platforms other than MS-DOS and MS-Windows.

−u, **−−unix-byte-offsets**
Report Unix-style byte offsets. This switch causes **grep** to report byte offsets as if the file were Unix-style text file, i.e. with CR characters stripped off. This will produce results identical to running **grep** on a Unix machine. This option has no effect unless **−b** option is also used; it has no effect on platforms other than MS-DOS and MS-Windows.

−V, **−−version**
Print the version number of **grep** to standard error. This version number should be included in all bug reports (see below).

−v, **−−invert-match**
Invert the sense of matching, to select non-matching lines.

−w, **−−word-regexp**
Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore.

−x, **−−line-regexp**
Select only those matches that exactly match the whole line.

−y Obsolete synonym for **−i**.

−Z, **−−null**
Output a zero byte (the ASCII **NUL** character) instead of the character that normally follows a file name. For example, **grep −lZ** outputs a zero byte after each file name instead of the usual newline. This option makes the output unambiguous, even in the presence of file names containing unusual characters like newlines. This option can be used with commands like **find −print0**, **perl**

−0, **sort −z**, and **xargs −0** to process arbitrary file names, even those that contain newline characters.

REGULAR EXPRESSIONS

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

Grep understands two different versions of regular expression syntax: “basic” and “extended.” In GNU **grep**, there is no difference in available functionality using either syntax. In other implementations, basic regular expressions are less powerful. The following description applies to extended regular expressions; differences for basic regular expressions are summarized afterwards.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any metacharacter with special meaning may be quoted by preceding it with a backslash.

A *bracket expression* is a list of characters enclosed by **[** and **]**. It matches any single character in that list; if the first character of the list is the caret ^ then it matches any character *not* in the list. For example, the regular expression **[0123456789]** matches any single digit.

Within a bracket expression, a *range expression* consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters, inclusive, using the locale’s collating sequence and character set. For example, in the default C locale, **[a−d]** is equivalent to **[abcd]**. Many locales sort characters in dictionary order, and in these locales **[a−d]** is typically not equivalent to **[abcd]**; it might be equivalent to **[aBbCcDd]**, for example. To obtain the traditional interpretation of bracket expressions, you can use the C locale by setting the **LC_ALL** environment variable to the value **C**.

Finally, certain named classes of characters are predefined within bracket expressions, as follows. Their names are self explanatory, and they are **[:alnum:]**, **[:alpha:]**, **[:cntrl:]**, **[:digit:]**, **[:graph:]**, **[:lower:]**, **[:print:]**, **[:punct:]**, **[:space:]**, **[:upper:]**, and **[:xdigit:].** For example, **[[:alnum:]]** means **[0−9A−Za−z]**, except the latter form depends upon the C locale and the ASCII character encoding, whereas the former is independent of locale and character set. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket list.) Most metacharacters lose their special meaning inside lists. To include a literal **]** place it first in the list. Similarly, to include a literal ^ place it anywhere but first. Finally, to include a literal − place it last.

The period **.** matches any single character. The symbol **\w** is a synonym for **[[:alnum:]]** and **\W** is a synonym for **[^[:alnum:]]**.

The caret ^ and the dollar sign **$** are metacharacters that respectively match the empty string at the beginning and end of a line. The symbols \< and \> respectively match the empty string at the beginning and end of a word. The symbol **\b** matches the empty string at the edge of a word, and **\B** matches the empty string provided it’s *not* at the edge of a word.

A regular expression may be followed by one of several repetition operators:

? The preceding item is optional and matched at most once.
***** The preceding item will be matched zero or more times.
+ The preceding item will be matched one or more times.
{*n***}** The preceding item is matched exactly *n* times.
{*n***,}** The preceding item is matched *n* or more times.
{*n***,***m***}** The preceding item is matched at least *n* times, but not more than *m* times.

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated subexpressions.

Two regular expressions may be joined by the infix operator |; the resulting regular expression matches any string matching either subexpression.

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole subexpression may be enclosed in parentheses to override these precedence rules.

The backreference *n*, where *n* is a single digit, matches the substring previously matched by the *n*th

parenthesized subexpression of the regular expression.

In basic regular expressions the metacharacters **?**, **+**, **{**, |, **(**, and **)** lose their special meaning; instead use the backslashed versions **\?**, **\+**, **\{**, \|, **\(**, and **\)**.

Traditional **egrep** did not support the **{** metacharacter, and some **egrep** implementations support **\{** instead, so portable scripts should avoid **{** in **egrep** patterns and should use **[{]** to match a literal **{**.

GNU **egrep** attempts to support traditional usage by assuming that **{** is not special if it would be the start of an invalid interval specification. For example, the shell command **egrep '{1'** searches for the two-character string **{1** instead of reporting a syntax error in the regular expression. POSIX.2 allows this behavior as an extension, but portable scripts should avoid it.

ENVIRONMENT VARIABLES

Grep's behavior is affected by the following environment variables.

A locale **LC_** *foo* is specified by examining the three environment variables **LC_ALL**, **LC_***foo*, **LANG**, in that order. The first of these variables that is set specifies the locale. For example, if **LC_ALL** is not set, but **LC_MESSAGES** is set to **pt_BR**, then Brazilian Portuguese is used for the **LC_MESSAGES** locale. The C locale is used if none of these environment variables are set, or if the locale catalog is not installed, or if **grep** was not compiled with national language support (NLS).

GREP_OPTIONS
: This variable specifies default options to be placed in front of any explicit options. For example, if **GREP_OPTIONS** is **'--binary-files=without-match --directories=skip'**, **grep** behaves as if the two options **--binary-files=without-match** and **--directories=skip** had been specified before any explicit options. Option specifications are separated by whitespace. A backslash escapes the next character, so it can be used to specify an option containing whitespace or a backslash.

GREP_COLOR
: Specifies the marker for highlighting.

LC_ALL, **LC_COLLATE**, **LANG**
: These variables specify the **LC_COLLATE** locale, which determines the collating sequence used to interpret range expressions like **[a-z]**.

LC_ALL, **LC_CTYPE**, **LANG**
: These variables specify the **LC_CTYPE** locale, which determines the type of characters, e.g., which characters are whitespace.

LC_ALL, **LC_MESSAGES**, **LANG**
: These variables specify the **LC_MESSAGES** locale, which determines the language that **grep** uses for messages. The default C locale uses American English messages.

POSIXLY_CORRECT
: If set, **grep** behaves as POSIX.2 requires; otherwise, **grep** behaves more like other GNU programs. POSIX.2 requires that options that follow file names must be treated as file names; by default, such options are permuted to the front of the operand list and are treated as options. Also, POSIX.2 requires that unrecognized options be diagnosed as "illegal", but since they are not really against the law the default is to diagnose them as "invalid". **POSIXLY_CORRECT** also disables **_***N***_GNU_nonoption_argv_flags_**, described below.

_*N***_GNU_nonoption_argv_flags_**
: (Here *N* is **grep**'s numeric process ID.) If the *i*th character of this environment variable's value is **1**, do not consider the *i*th operand of **grep** to be an option, even if it appears to be one. A shell can put this variable in the environment for each command it runs, specifying which operands are the results of file name wildcard expansion and therefore should not be treated as options. This behavior is available only with the GNU C library, and only when **POSIXLY_CORRECT** is not set.

DIAGNOSTICS

Normally, exit status is 0 if selected lines are found and 1 otherwise. But the exit status is 2 if an error occurred, unless the **−q** or **−−quiet** or **−−silent** option is used and a selected line is found.

BUGS

Email bug reports to **bug-gnu-utils@gnu.org**. Be sure to include the word “grep” somewhere in the “Subject:” field.

Large repetition counts in the ***{n,m}*** construct may cause grep to use lots of memory. In addition, certain other obscure regular expressions require exponential time and space, and may cause **grep** to run out of memory.

Backreferences are very slow, and may require exponential time.

NAME

grep, egrep, fgrep, zgrep — file pattern searcher

SYNOPSIS

grep [**-ABC** *num*] [**-EFGHILPSVZabchilnqrsvwxz**] [**-d** *action*] [**-e** *pattern*] [**-f** *file*] [*file . . .*]

DESCRIPTION

The **grep** utilities search the given input files selecting lines that match one or more patterns. By default, a pattern matches an input line if any regular expression (RE) in the pattern matches the input line without its trailing newline. An empty expression matches every line. Each input line that matches at least one of the patterns is written to the standard output.

The **grep** utility is used for simple patterns and ex(1) or ed(1) style regular expressions. The **egrep** utility can handle extended regular expressions and multi-line patterns. The **fgrep** utility is quick but can handle only fixed patterns consisting of one or more lines, allowing any of the pattern lines to match a portion of the input. The **zgrep** utility acts like grep, but accepts input files compressed with the compress(1) or gzip(1) compression utilities.

The following options are available:

-A *num*
: Print *num* lines of trailing context after each match.

-B *num*
: Print *num* lines of leading context before each match.

-C *num*
: Print *num* lines of leading and trailing context before and after each match respectively.

-D *action*
: Use the specified action when processing input files that are devices, FIFOs or sockets. The default action is **read**, which means that that devices are treated as ordinary files. If the action is **skip** then devices will not be processed at all.

-E
: Force **grep** to behave as **egrep.**

-F
: Force **grep** to behave as **fgrep.**

-G
: Force **grep** to behave as **grep.**

-H
: Always print filename headers with output lines.

-I
: If the file being processed is binary, assume that it does not contain matching data.

-L
: Only the names of files not containing selected lines are written to standard output. Pathnames are listed once per file searched. If the standard input is searched, the pathname '**-**' is written.

-P
: If **-R** is specified, no symbolic links are followed.

-S
: If **-R** is specified, all symbolic links are followed.

-V
: Display version information.

-Z
: When printing filenames, output a null character after the filename instead of a newline character.

-a
: Treat all files as text.

-b
: The block number on the disk in which a matched pattern is located is displayed in front of the respective matched line.

-c Only a count of selected lines is written to standard output.

-d *action*
Use the specified action when processing input files that are directories. The default action is **read**, which means that directories are treated as ordinary files. If the action is **skip** then directories will not be processed at all. If the action is **recurse** then **grep** will read all files under each directory recursively.

-e *expression*
Specify a pattern used during the search of the input. Multiple **-e** options can be used to specify multiple patterns; an input line is selected if it matches any of the specified patterns.

-f *pattern_file*
The pattern is read from the specified file. Trailing newlines in the pattern file are ignored. (**Egrep** and **fgrep** only).

-h Never print filename headers with output lines.

-i Perform case insensitive matching.

-l Only the names of files containing selected lines are written to standard output. Pathnames are listed once per file searched. If the standard input is searched, the pathname '-' is written.

-n Each output line is preceded by its relative line number in the file, starting at line 1. The line number counter is reset for each file processed. This option is ignored if **-c**, **-l**, or **-s** is specified.

-q Suppress normal output.

-r Recursively search subdirectories listed.

-s Silent mode. Nonexistent and unreadable files are ignored.

-v Selected lines are those *not* matching the specified patterns.

-w The expression is searched for as a word (as if surrounded by '\<' and '\>', see ex(1)).

-x Only input lines selected against an entire fixed string or regular expression are considered to be matching lines.

-z When printing matched lines, output a null character after the line instead of a newline character.

If no file arguments are specified, the standard input is used.

The **grep** utility exits with one of the following values:

0 One or more lines were selected.
1 No lines were selected.
>1 An error occurred.

EXTENDED REGULAR EXPRESSIONS

The following characters are interpreted by **egrep**:

$ Align the match from the end of the line.
^ Align the match from the beginning of the line.
| Add another pattern (see example below).
? Match 1 or less sequential repetitions of the pattern.
+ Match 1 or more sequential repetitions of the pattern.
* Match 0 or more sequential repetitions of the pattern.

[] Match any single character or range of characters enclosed in the brackets.
**** Escape special characters which have meaning to **egrep**, the set of {$,.,^,[,],|,?,+,*,(,)}.

EXAMPLES

To find all occurrences of the word patricia in a file:

```
grep patricia myfile
```

To find all occurrences of the pattern `.Pp` at the beginning of a line:

```
grep '^\.Pp'
```

The apostrophes assure the entire expression is evaluated by **grep** instead of by the user's shell. The caret `^` matches the null string at the beginning of a line, and the `\` escapes the `.` which would otherwise match any character.

A simple example of an extended regular expression:

```
egrep '19|20|25' calendar
```

Peruses the file calendar looking for either 19, 20 or 25.

SEE ALSO

ed(1), ex(1), sed(1), re_format(7), regex(3), regexp(3)

HISTORY

The **grep** command appeared in Version 6 AT&T UNIX.

NAME

grfinfo — display information about grf graphics devices

SYNOPSIS

grfinfo [**-at**] *file*

DESCRIPTION

The **grfinfo** utility displays information about grf graphics frame buffer devices. By default, only the frame buffer type is displayed. *file* is the device file for the graphics frame buffer.

The options are as follows:

-a Display all possible information.

-t Display only the type, even if an error occurs.

EXIT STATUS

The **grfinfo** utility exits 0 on success, and >0 if an error occurs. If the **-t** option is used and an error occurs, no error message is displayed, and the type is displayed as "none".

SEE ALSO

grf(4)

NAME

groups — show group memberships

SYNOPSIS

groups [*user*]

DESCRIPTION

The **groups** utility has been obsoleted by the id(1) utility, and is equivalent to "**id -Gn** [*user*]". The command "**id -p**" is suggested for normal interactive use.

The **groups** utility displays the groups to which you (or the optionally specified user) belong.

The **groups** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

id(1)

NAME

gzexe — create auto-decompressing executables

SYNOPSIS

gzexe [**-d**] *file ...*

DESCRIPTION

The **gzexe** utility uses gzip(1) to compress executables, producing executables that decompress on-the-fly when executed. This saves disk space, at the cost of slower execution times. The original executables are saved by copying each of them to a file with the same name with a '~' suffix appended. After verifying that the compressed executables work as expected, the backup files can be removed.

The options are as follows:

-d Decompress executables previously compressed by **gzexe**.

The **gzexe** program refuses to compress non-regular or non-executable files, files with a setuid or setgid bit set, files that are already compressed using **gzexe** or programs it needs to perform on-the-fly decompression: sh(1), mktemp(1), rm(1), echo(1), tail(1), gzip(1), and chmod(1).

SEE ALSO

gzip(1)

CAVEATS

The **gzexe** utility replaces files by overwriting them with the generated compressed executable. To be able to do this, it is required that the original files are writable.

NAME

gzip — compression/decompression tool using Lempel-Ziv coding (LZ77)

SYNOPSIS

gzip [**-cdfhlNnqrtVv**] [**-S** *suffix*] *file* [*file* [...]]
gunzip [**-cfhNqrtVv**] [**-S** *suffix*] *file* [*file* [...]]
zcat [**-fhV**] *file* [*file* [...]]

DESCRIPTION

The **gzip** program compresses and decompresses files using Lempel-Ziv coding (LZ77). If no *files* are specified, **gzip** will compress from standard input, or decompress to standard output. When in compression mode, each *file* will be replaced with another file with the suffix, set by the **-S** *suffix* option, added, if possible. In decompression mode, each *file* will be checked for existence, as will the file with the suffix added.

If invoked as **gunzip** then the **-d** option is enabled. If invoked as **zcat** or **gzcat** then both the **-c** and **-d** options are enabled.

This version of **gzip** is also capable of decompressing files compressed using compress(1) or bzip2(1).

OPTIONS

The following options are available:

-1, **--fast**

-2

-3

-4

-5

-6

-7

-8

-9, **--best** These options change the compression level used, with the **-1** option being the fastest, with less compression, and the **-9** option being the slowest, with optimal compression. The default compression level is 6.

-c, **--stdout**, **--to-stdout**
This option specifies that output will go to the standard output stream, leaving files intact.

-d, **--decompress**, **--uncompress**
This option selects decompression rather than compression.

-f, **--force** This option turns on force mode. This allows files with multiple links, overwriting of pre-existing files, reading from or writing to a terminal, and when combined with the **-c** option, allowing non-compressed data to pass through unchanged.

-h, **--help** This option prints a usage summary and exits.

-l, **--list** This option displays information about the file's compressed and uncompressed size, ratio, uncompressed name. With the **-v** option, it also displays the compression method, CRC, date and time embedded in the file.

-N, **--name** This option causes the stored filename in the input file to be used as the output file.

-n, **--no-name** This option stops the filename from being stored in the output file.

-q, **--quiet** With this option, no warnings or errors are printed.

-r, **--recursive** This option is used to **gzip** the files in a directory tree individually, using the fts(3) library.

-S *suffix*, **--suffix** *suffix*
This option changes the default suffix from .gz to *suffix*.

-t, **--test** This option will test compressed files for integrity.

-V, **--version** This option prints the version of the **gzip** program.

-v, **--verbose** This option turns on verbose mode, which prints the compression ratio for each file compressed.

ENVIRONMENT

If the environment variable GZIP is set, it is parsed as a white-space separated list of options handled before any options on the command line. Options on the command line will override anything in GZIP.

SEE ALSO

bzip2(1), compress(1), fts(3), zlib(3)

HISTORY

The **gzip** program was originally written by Jean-loup Gailly, licensed under the GNU Public Licence. Matthew R. Green wrote a simple front end for NetBSD 1.3 distribution media, based on the freely re-distributable zlib library. It was enhanced to be mostly feature-compatible with the original GNU **gzip** program for NetBSD 2.0.

This manual documents NetBSD **gzip** version 20040427.

AUTHORS

This implementation of **gzip** was written by Matthew R. Green ⟨mrg@eterna.com.au⟩.

NAME

head — display first lines of a file

SYNOPSIS

head [**-qv**] [**-n** *count*] [**-c** *byte_count*] [*file ...*]

DESCRIPTION

This filter displays the first *count* lines of each of the specified files, or of the standard input if no files are specified. If *count* is omitted it defaults to 10. If **-c** *byte_count* is specified, **head** counts bytes instead of lines.

If more than a single file is specified, or the **-v** option is used, each file is preceded by a header consisting of the string "==> XXX <==" where "XXX" is the name of the file. The **-q** flag disables the printing of the header in all cases.

The **head** utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

The historic command line syntax of **head** is supported by this implementation.

This command is mostly compatible with GNU extensions to **head**.

SEE ALSO

tail(1)

STANDARDS

The **head** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2").

HISTORY

The **head** utility appeared in 3.0BSD. It was enhanced to include the **-c**, **-q**, and **-v** options for NetBSD 2.1.

NAME

hesinfo — find out what is stored in the Hesiod database

SYNOPSIS

hesinfo [**-bl**] *HesiodName HesiodNameType*

DESCRIPTION

hesinfo takes two arguments, a name to be resolved and a string, known as a *HesiodNameType*. It then prints the information returned by the Hesiod nameserver.

The value returned by **hesinfo** is of the type *HesiodNameType*.

hesinfo understands the following options:

-l
Selects long format.

-b
Prints the fully-qualified string passed to the nameserver.

VALID Hesiod_Names

The following types of identifiers may be used in the *HesiodName* argument to **hesinfo**. These values will be resolved by accessing the hesiod(3) database.

⟨username⟩	the 8-character-or-less string used to identify users or classes (e.g. joeuser, root, 1.00, etc). Used with the *Hesiod_Name_Types* passwd, pobox, and filsys.
⟨uid⟩	the id number assigned to a user.
⟨groupid⟩	the id number assigned to a group.
⟨groupname⟩	a name identifying a unique group.
⟨file-system-name⟩	the name of an athena file system.
⟨rvd server⟩:⟨pack⟩	the name of an rvd's server and pack separated by a colon.
⟨nfs server⟩:⟨partition⟩	the name of an nfs server and its partition separated by a colon.
⟨workstation-name⟩	the machine name of an Athena workstation (e.g. E40-343-3).
⟨service-name⟩	name of an Athena service (e.g. Zephyr).
⟨service-type⟩	name of Unix service (valid entries are defined in /etc/services).
⟨printer-name⟩	name of a printer.
⟨printer-cluster-name⟩	name of an Athena print cluster.
⟨foo⟩	some **hesinfo** calls (e.g., prclusterlist) do not require a specific *HesiodName* argument. However, you must include a dummy string (e.g., "foo") for **hesinfo** to work properly.

VALID Hesiod_Name_Types

The following symbols are valid substitutions for the *HesiodNameType* argument to **hesinfo**.

passwd returns string suitable for inclusion in /etc/passwd, searching with ⟨username⟩.

pobox returns information on the pobox assigned to the user specified by *HesiodName*, searching with ⟨username⟩.

uid returns string suitable for inclusion in `/etc/passwd`, searching with ⟨uid⟩.

gid returns string suitable for inclusion in `/etc/group`, searching with ⟨groupid⟩.

group returns string suitable for inclusion in `/etc/group`, searching with ⟨groupname⟩.

grplist returns subgroups included in superset defined by ⟨groupname⟩.

filsys returns file system type, export point, server, mount mode, and import point for the following valid *HesiodNames* (see above) - ⟨`file system name`⟩, ⟨`username`⟩, ⟨`rvd server`⟩:⟨pack⟩, and ⟨`nfs server`⟩:⟨partition⟩.

cluster returns information about the local cluster the workstation, specified by ⟨workstation name⟩. Included is information about the local file and print servers. This information is accesses by `clusterinfo` at boot time.

sloc returns network name of service host for ⟨service-name⟩.

service returns Internet protocol type and protocol service port for ⟨service-type⟩.

pcap returns a valid entry for `/etc/printcap` for ⟨printer-name⟩.

prcluserlist
returns a list of print clusters.

prcluster returns a list of printers in a cluster specified by ⟨printer-cluster-name⟩.

FILES

`/etc/hesiod.conf`

SEE ALSO

`hesiod`(3), `named`(8)

Hesiod - Project Athena Technical Plan -- Name Service.

AUTHORS

Steve Dyer, IBM/Project Athena

NAME

hexdump — ascii, decimal, hexadecimal, octal dump

SYNOPSIS

hexdump [**-bcCdovx**] [**-e** *format_string*] [**-f** *format_file*] [**-n** *length*] [**-s** *skip*] *file ...*

DESCRIPTION

The hexdump utility is a filter which displays the specified files, or the standard input, if no files are specified, in a user specified format.

The options are as follows:

-b *One-byte octal display*. Display the input offset in hexadecimal, followed by sixteen space-separated, three column, zero-filled, bytes of input data, in octal, per line.

-c *One-byte character display*. Display the input offset in hexadecimal, followed by sixteen space-separated, three column, space-filled, characters of input data per line.

-C *Canonical hex+ASCII display*. Display the input offset in hexadecimal, followed by sixteen space-separated, two column, hexadecimal bytes, followed by the same sixteen bytes in %_p format enclosed in ‘‘|’’ characters.

-d *Two-byte decimal display*. Display the input offset in hexadecimal, followed by eight space-separated, five column, zero-filled, two-byte units of input data, in unsigned decimal, per line.

-e *format_string*
Specify a format string to be used for displaying data.

-f *format_file*
Specify a file that contains one or more newline separated format strings. Empty lines and lines whose first non-blank character is a hash mark (**#**) are ignored.

-n *length*
Interpret only *length* bytes of input.

-o *Two-byte octal display*. Display the input offset in hexadecimal, followed by eight space-separated, six column, zero-filled, two byte quantities of input data, in octal, per line.

-s *offset*
Skip *offset* bytes from the beginning of the input. By default, *offset* is interpreted as a decimal number. With a leading **0x** or **0X**, *offset* is interpreted as a hexadecimal number, otherwise, with a leading **0**, *offset* is interpreted as an octal number. Appending the character **b**, **k**, or **m** to *offset* causes it to be interpreted as a multiple of 512, 1024, or 1048576, respectively.

-v The **-v** option causes hexdump to display all input data. Without the **-v** option, any number of groups of output lines, which would be identical to the immediately preceding group of output lines (except for the input offsets), are replaced with a line containing a single asterisk.

-x *Two-byte hexadecimal display*. Display the input offset in hexadecimal, followed by eight, space separated, four column, zero-filled, two-byte quantities of input data, in hexadecimal, per line.

For each input file, **hexdump** sequentially copies the input to standard output, transforming the data according to the format strings specified by the **-e** and **-f** options, in the order that they were specified.

Formats

A format string contains any number of format units, separated by whitespace. A format unit contains up to three items: an iteration count, a byte count, and a format.

The iteration count is an optional positive integer, which defaults to one. Each format is applied iteration count times.

The byte count is an optional positive integer. If specified it defines the number of bytes to be interpreted by each iteration of the format.

If an iteration count and/or a byte count is specified, a single slash must be placed after the iteration count and/or before the byte count to disambiguate them. Any whitespace before or after the slash is ignored.

The format is required and must be surrounded by double quote (" ") marks. It is interpreted as a fprintf-style format string (see `fprintf`(3)), with the following exceptions:

- An asterisk (*) may not be used as a field width or precision.
- A byte count or field precision *is* required for each ‘‘s’’ conversion character (unlike the `fprintf`(3) default which prints the entire string if the precision is unspecified).
- The conversion characters ‘‘h’’, ‘‘l’’, ‘‘n’’, ‘‘p’’ and ‘‘q’’ are not supported.
- The single character escape sequences described in the C standard are supported:

NUL	\0
<alert character>	\a
<backspace>	\b
<form-feed>	\f
<newline>	\n
<carriage return>	\r
<tab>	\t
<vertical tab>	\v

Hexdump also supports the following additional conversion strings:

_a[**dox**] Display the input offset, cumulative across input files, of the next byte to be displayed. The appended characters **d**, **o**, and **x** specify the display base as decimal, octal or hexadecimal respectively.

_A[**dox**] Identical to the **_a** conversion string except that it is only performed once, when all of the input data has been processed.

_c Output characters in the default character set. Nonprinting characters are displayed in three character, zero-padded octal, except for those representable by standard escape notation (see above), which are displayed as two character strings.

_p Output characters in the default character set. Nonprinting characters are displayed as a single “**.**”.

_u Output US ASCII characters, with the exception that control characters are displayed using the following, lower-case, names. Characters greater than 0xff, hexadecimal, are displayed as hexadecimal strings.

000 nul	001 soh	002 stx	003 etx	004 eot	005 enq
006 ack	007 bel	008 bs	009 ht	00A lf	00B vt
00C ff	00D cr	00E so	00F si	010 dle	011 dc1

012 dc2 013 dc3 014 dc4 015 nak 016 syn 017 etb
018 can 019 em 01A sub 01B esc 01C fs 01D gs
01E rs 01F us 0FF del

The default and supported byte counts for the conversion characters are as follows:

`%_c`, `%_p`, `%_u`, `%c`	One byte counts only.
`%d`, `%i`, `%o`, `%u`, `%X`, `%x`	Four byte default, one, two, four and eight byte counts supported.
`%E`, `%e`, `%f`, `%G`, `%g`	Eight byte default, four byte counts supported.

The amount of data interpreted by each format string is the sum of the data required by each format unit, which is the iteration count times the byte count, or the iteration count times the number of bytes required by the format if the byte count is not specified.

The input is manipulated in ‘‘blocks’’, where a block is defined as the largest amount of data specified by any format string. Format strings interpreting less than an input block’s worth of data, whose last format unit both interprets some number of bytes and does not have a specified iteration count, have the iteration count incremented until the entire input block has been processed or there is not enough data remaining in the block to satisfy the format string.

If, either as a result of user specification or hexdump modifying the iteration count as described above, an iteration count is greater than one, no trailing whitespace characters are output during the last iteration.

It is an error to specify a byte count as well as multiple conversion characters or strings unless all but one of the conversion characters or strings is **_a** or **_A**.

If, as a result of the specification of the **−n** option or end-of-file being reached, input data only partially satisfies a format string, the input block is zero-padded sufficiently to display all available data (i.e. any format units overlapping the end of data will display some number of the zero bytes).

Further output by such format strings is replaced by an equivalent number of spaces. An equivalent number of spaces is defined as the number of spaces output by an **s** conversion character with the same field width and precision as the original conversion character or conversion string but with any “+”, “ ”, “#” conversion flag characters removed, and referencing a NULL string.

If no format strings are specified, the default display is equivalent to specifying the **−x** option.

hexdump exits 0 on success and >0 if an error occurred.

EXAMPLES

Display the input in perusal format:

```
"%06.6_ao "  12/1 "%3_u "
"\t\t" "%_p "
"\n"
```

Implement the −x option:

```
"%07.7_Ax\n"
"%07.7_ax  " 8/2 "%04x " "\n"
```

NAME

host − DNS lookup utility

SYNOPSIS

host [**−aCdlnrsTwv**] [**−c** *class*] [**−N** *ndots*] [**−R** *number*] [**−t** *type*] [**−W** *wait*] [**−m** *flag*] [**−4**] [**−6**] {name} [server]

DESCRIPTION

host is a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa. When no arguments or options are given, **host** prints a short summary of its command line arguments and options.

name is the domain name that is to be looked up. It can also be a dotted−decimal IPv4 address or a colon−delimited IPv6 address, in which case **host** will by default perform a reverse lookup for that address. *server* is an optional argument which is either the name or IP address of the name server that **host** should query instead of the server or servers listed in */etc/resolv.conf*.

The **−a** (all) option is equivalent to setting the **−v** option and asking **host** to make a query of type ANY.

When the **−C** option is used, **host** will attempt to display the SOA records for zone *name* from all the listed authoritative name servers for that zone. The list of name servers is defined by the NS records that are found for the zone.

The **−c** option instructs to make a DNS query of class *class*. This can be used to lookup Hesiod or Chaosnet class resource records. The default class is IN (Internet).

Verbose output is generated by **host** when the **−d** or **−v** option is used. The two options are equivalent. They have been provided for backwards compatibility. In previous versions, the **−d** option switched on debugging traces and **−v** enabled verbose output.

List mode is selected by the **−l** option. This makes **host** perform a zone transfer for zone *name*. Transfer the zone printing out the NS, PTR and address records (A/AAAA). If combined with **−a** all records will be printed.

The **−i** option specifies that reverse lookups of IPv6 addresses should use the IP6.INT domain as defined in RFC1886. The default is to use IP6.ARPA.

The **−N** option sets the number of dots that have to be in *name* for it to be considered absolute. The default value is that defined using the ndots statement in */etc/resolv.conf*, or 1 if no ndots statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the **search** or **domain** directive in */etc/resolv.conf*.

The number of UDP retries for a lookup can be changed with the **−R** option. *number* indicates how many times **host** will repeat a query that does not get answered. The default number of retries is 1. If *number* is negative or zero, the number of retries will default to 1.

Non−recursive queries can be made via the **−r** option. Setting this option clears the **RD** — recursion desired — bit in the query which **host** makes. This should mean that the name server receiving the query will not attempt to resolve *name*. The **−r** option enables **host** to mimic the behaviour of a name server by making non−recursive queries and expecting to receive answers to those queries that are usually referrals to other name servers.

By default **host** uses UDP when making queries. The **−T** option makes it use a TCP connection when querying the name server. TCP will be automatically selected for queries that require it, such as zone transfer (AXFR) requests.

The **−4** option forces **host** to only use IPv4 query transport. The **−6** option forces **host** to only use IPv6 query transport.

The **−t** option is used to select the query type. *type* can be any recognised query type: CNAME, NS, SOA, SIG, KEY, AXFR, etc. When no query type is specified, **host** automatically selects an appropriate query type. By default it looks for A records, but if the **−C** option was given, queries will be made for SOA records, and if *name* is a dotted−decimal IPv4 address or colon−delimited IPv6 address, **host** will query for

PTR records. If a query type of IXFR is chosen the starting serial number can be specified by appending an equal followed by the starting serial number (e.g. −t IXFR=12345678).

The time to wait for a reply can be controlled through the **−W** and **−w** options. The **−W** option makes **host** wait for *wait* seconds. If *wait* is less than one, the wait interval is set to one second. When the **−w** option is used, **host** will effectively wait forever for a reply. The time to wait for a response will be set to the number of seconds given by the hardware's maximum value for an integer quantity.

The **−s** option tells **host** *not* to send the query to the next nameserver if any server responds with a SERVFAIL response, which is the reverse of normal stub resolver behaviour.

The **−m** can be used to set the memory usage debugging flags *record*, *usage* and *trace*.

IDN SUPPORT

If **host** has been built with IDN (internationalized domain name) support, it can accept and display non−ASCII domain names. **host** appropriately converts character encoding of domain name before sending a request to DNS server or displaying a reply from the server. If you'd like to turn off the IDN support for some reason, defines the **IDN_DISABLE** environment variable. The IDN support is disabled if the variable is set when **host** runs.

FILES

/etc/resolv.conf

SEE ALSO

dig(1), **named**(8).

COPYRIGHT

NAME

hostapd_cli − hostapd command-line interface

SYNOPSIS

hostapd_cli [-p<path>] [-i<ifname>] [-hv] [command..]

DESCRIPTION

This manual page documents briefly the **hostapd_cli** utility.

hostapd_cli is a command-line interface for the **hostapd** daemon.

hostapd is a user space daemon for access point and authentication servers. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators and RADIUS authentication server. For more information about **hostapd** refer to the **hostapd**(8) man page.

OPTIONS

A summary of options is included below. For a complete description, run **hostapd_cli** from the command line.

−p<path>
Path to find control sockets.

Default: /var/run/hostapd

−i<ifname>
Interface to listen on.

Default: first interface found in socket path.

−h Show usage.

−v Show hostapd_cli version.

COMMANDS

A summary of commands is included below. For a complete description, run **hostapd_cli** from the command line.

mib Get MIB variables (dot1x, dot11, radius).

sta <addr>
Get MIB variables for one station.

all_sta Get MIB variables for all stations.

help Get usage help.

interface [ifname]
Show interfaces/select interface.

level <debug level>
Change debug level.

license Show full **hostapd_cli** license.

quit Exit hostapd_cli.

SEE ALSO

hostapd(8).

AUTHOR

hostapd_cli was written by Jouni Malinen <j@w1.fi>.

This manual page was written by Faidon Liambotis <faidon@cube.gr>, for the Debian project (but may be used by others).

NAME

hostname — set or print name of current host system

SYNOPSIS

hostname [**-s**] [*name-of-host*]

DESCRIPTION

hostname prints the name of the current host. The super-user can set the host name by supplying an argument; this is usually done in the network initialization script `/etc/rc.d/network`, normally run at boot time.

Options:

-s Trims off any domain information from the printed name.

SEE ALSO

`domainname`(1), `gethostname`(3), `sethostname`(3)

HISTORY

The **hostname** utility appeared in 4.2BSD.

NAME

iconv — codeset conversion utility

SYNOPSIS

iconv [**-cs**] **-f** *from_name* **-t** *to_name* [*file ...*]
iconv -f *from_name* [**-cs**][**-t** *to_name*][*file ...*]
iconv -t *to_name* [**-cs**][**-f** *from_name*][*file ...*]
iconv -l

DESCRIPTION

The **iconv** utility converts the codeset of *file* (or from standard input if no file is specified) from codeset *from_name* to codeset *to_name* and outputs the converted text on standard output.

The following options are available:

-c Prevent output of any invalid characters. By default, **iconv** outputs an "invalid character" specified by the *to_name* codeset when it encounts a character which is valid in the *from_name* codeset but does not have a corresponding character in the *to_name* codeset.

-f Specifies the source codeset name as *from_name*.

-l Lists available codeset names. Note that not all combinations of *from_name* and *to_name* are valid.

-s Silent. By default, **iconv** outputs the number of "invalid characters" to standard error if they exist. This option prevents this behaviour.

-t Specifies the destination codeset name as *to_name*.

EXIT STATUS

The **iconv** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

iconv(3)

STANDARDS

iconv conform to IEEE Std 1003.1-2001 ("POSIX.1").

HISTORY

iconv first appeared in NetBSD 2.0.

NAME

id — return user identity

SYNOPSIS

id [*user*]
id -G [**-n**] [*user*]
id -g [**-nr**] [*user*]
id -p [*user*]
id -u [**-nr**] [*user*]

DESCRIPTION

The **id** utility displays the user and group names and numeric IDs, of the calling process, to the standard output. If the real and effective IDs are different, both are displayed, otherwise only the real ID is displayed.

If a *user* (login name or user ID) is specified, the user and group IDs of that user are displayed. In this case, the real and effective IDs are assumed to be the same.

The options are as follows:

-G Display the different group IDs (effective, real and supplementary) as white-space separated numbers, in no particular order.

-g Display the effective group ID as a number.

-n Display the name of the user or group ID for the **-G**, **-g** and **-u** options instead of the number. If any of the ID numbers cannot be mapped into names, the number will be displayed as usual.

-p Make the output human-readable. If the user name returned by `getlogin`(2) is different from the login name referenced by the user ID, the name returned by `getlogin`(2) is displayed, preceded by the keyword ‘‘login’’. The user ID as a name is displayed, preceded by the keyword ‘‘uid’’. If the effective user ID is different from the real user ID, the real user ID is displayed as a name, preceded by the keyword ‘‘euid’’. If the effective group ID is different from the real group ID, the real group ID is displayed as a name, preceded by the keyword ‘‘rgid’’. The list of groups to which the user belongs is then displayed as names, preceded by the keyword ‘‘groups’’. Each display is on a separate line.

-r Display the real ID for the **-g** and **-u** options instead of the effective ID.

-u Display the effective user ID as a number.

The **id** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

`who`(1)

STANDARDS

The **id** function is expected to conform to IEEE Std 1003.2 (“POSIX.2”).

HISTORY

The historic `groups`(1) command is equivalent to “**id -Gn** [*user*]”.

The historic `whoami`(1) command is equivalent to “**id -un**”.

The **id** command first appeared in 4.4BSD.

NAME

ident − identify RCS keyword strings in files

SYNOPSIS

ident [**−q**] [**−V**] [*file* . . .]

DESCRIPTION

ident searches for all instances of the pattern **$***keyword***:** *text* **$** in the named files or, if no files are named, the standard input.

These patterns are normally inserted automatically by the RCS command **co**(1), but can also be inserted manually. The option **−q** suppresses the warning given if there are no patterns in a file. The option **−V** prints **ident**'s version number.

ident works on text files as well as object files and dumps. For example, if the C program in **f.c** contains

```
#include <stdio.h>
static char const rcsid[] =
  "$Id: f.c,v  $";
int main() { return printf("%s\n", rcsid) == EOF; }
```

and **f.c** is compiled into **f.o**, then the command

```
ident  f.c  f.o
```

will output

```
f.c:
   $Id: f.c,v  $
f.o:
   $Id: f.c,v  $
```

If a C program defines a string like **rcsid** above but does not use it, **lint**(1) may complain, and some C compilers will optimize away the string. The most reliable solution is to have the program use the **rcsid** string, as shown in the example above.

ident finds all instances of the **$***keyword***:** *text* **$** pattern, even if *keyword* is not actually an RCS-supported keyword. This gives you information about nonstandard keywords like **$XConsortium$**.

KEYWORDS

Here is the list of keywords currently maintained by **co**(1). All times are given in Coordinated Universal Time (UTC, sometimes called GMT) by default, but if the files were checked out with **co**'s **−z***zone* option, times are given with a numeric time zone indication appended.

$Author$
The login name of the user who checked in the revision.

$Date$ The date and time the revision was checked in.

$Header$
A standard header containing the full pathname of the RCS file, the revision number, the date and time, the author, the state, and the locker (if locked).

Id Same as **$Header$**, except that the RCS filename is without a path.

$Locker$
The login name of the user who locked the revision (empty if not locked).

Log The log message supplied during checkin. For **ident**'s purposes, this is equivalent to **$RCSfile$**.

$Name$
The symbolic name used to check out the revision, if any.

$RCSfile$
The name of the RCS file without a path.

$Revision$
The revision number assigned to the revision.

$Source$
The full pathname of the RCS file.

$State$
The state assigned to the revision with the **−s** option of **rcs**(1) or **ci**(1).

co(1) represents the following characters in keyword values by escape sequences to keep keyword strings well-formed.

char	*escape sequence*
tab	**\t**
newline	**\n**
space	**\040**
$	**\044**
\	\\

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

ci(1), co(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rlog(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

idnconv − codeset converter for named.conf and zone master files

SYNOPSIS

idnconv [*options..*] [*file...*]

DESCRIPTION

idnconv is a codeset converter for named configuration files and zone master files. **idnconv** performs codeset conversion specified either by the command-line arguments or by the configuration file, and writes the converted text to stdout.

If file name is specified, **idnconv** converts the contents of the file. Otherwise, **idnconv** converts *stdin*.

Since **idnconv** is specifically designed for converting internatinalized domain names, it may not be suitable as a general codeset converter.

OPERATION MODES

idnconv has two operation modes.

One is a mode to convert local-encoded domain names to IDN-encoded one. Usually this mode is used for preparing domain names to be listed in named configuration files or zone master files. In this mode, the following processes are performed in addition to the codeset (encoding) conversion.

–
local mapping

–
standard domain name preperation (NAMEPREP)

The other mode is a reverse conversion, from IDN-encoded domain name to local-encoded domain names. In this mode, local mapping and NAMEPREP are not performed since IDN-encoded names should already be normalized. Instead, a check is done in order to make sure the IDN-encoded domain name is properly NAMEPREP'ed. If it is not, the name will be output in IDN encoding, not in the local encoding.

OPTIONS

Normally **idnconv** reads system's default configuration file (idn.conf) and performs conversion or name preparation according to the parameters specified in the file. You can override the setting in the configuration file by various command line options below.

−in *in-code*, **−i** *in-code*
Specify the codeset name of the input text. Any of the following codeset names can be specified.

- Any codeset names which *iconv_open()* library function accepts
- `Punycode`
- `UTF-8`
- Any alias names for the above, defined by the codeset alias file.

If this option is not specified, the default codeset is determined from the locale in normal conversion mode. In reverse conversion mode, the default codeset is the IDN encoding specified by the configuration file (''idn-encoding'' entry).

−out *out-code*, **−o** *out-code*
Specify the codeset name of the output text. *out-code* can be any codeset name that can be specified for **−in** option.

If this option is not specified, the default is the IDN encoding specified by the configuration file (''idn-encoding'' entry) in normal conversion mode. In reverse conversion mode, the default codeset is determined from the locale.

−conf *path*, **−c** *path*
Specify the pathname of idnkit configuration file (''idn.conf''). If not specified, system's default file is used, unless −noconf option is specified.

−noconf, **−C**
Specify that no configuration file is to be used.

−reverse, **−r**
Specify reverse conversion mode.
If this option is not specified, the normal conversion mode is used.

−nameprep *version*, **−n** *version*
Specify the version of NAMEPREP. The following is a list of currently available versions.

`RFC3491`
Perform NAMEPREP according to the RFC3491 ‘‘rfc-3491.txt’’.

−nonameprep, **−N**
Specify to skip NAMEPREP process (or NAMEPREP verification process in the reverse conversion mode). This option implies -nounassigncheck and -nobidicheck.

−localmap *map*
Specify the name of local mapping rule. Currently, following maps are available.

`RFC3491`
Use the list of mappings specified by RFC3491.

`filemap:`*path*
Use list of mappings specified by mapfile *path*. See idn.conf(5) for the format of a mapfile.

This option can be specified more than once. In that case, each mapping will be performed in the order of the specification.

−nounassigncheck, **−U**
Skip unassigned codepoint check.

−nobidicheck, **−B**
Skip bidi character check.

−nolengthcheck
Do not check label length of normal conversion result. This option is only meaningful in the normal conversion mode.

−noasciicheck, **−A**
Do not check ASCII range characters. This option is only meaningful in the normal conversion mode.

−noroundtripcheck
Do not perform round trip check. This option is only meaningful in the reverse conversion mode.

−delimiter *codepoint*
Specify the character to be mapped to domain name delimiter (period). This option can be specified more than once in order to specify multiple characters.
This option is only meaningful in the normal conversion mode.

−whole, **−w**
Perform local mapping, nameprep and conversion to output codeset for the entire input text. If this option is not specified, only non-ASCII characters and their surrounding texts will be processed. See ‘‘NORAML CONVERSION MECHANISM’’ and ‘‘REVERSE CONVERSION MECHANISM’’ for details.

−alias *path*, **−a** *path*
Specify a codeset alias file. It is a simple text file, where each line has a pair of alias name and real name separated by one or more white spaces like below:

alias-codeset-name *real-codeset-name*

Lines starting with ‘‘#’’ are treated as comments.

−flush
Force line-buffering mode.

−version, **−v**
Print version information and quit.

LOCAL CODESET

idnconv guesses local codeset from locale and environment variables. See the ‘‘LOCAL CODESET’’ section in idn.conf(5) for more details.

NORMAL CONVERSION MECHANISM

idnconv performs conversion line by line. Here describes how **idnconv** does its job for each line.

1. read a line from input text

2. convert the line to UTF-8
   idnconv converts the line from local encoding to UTF-8.

3. find internationalized domain names
   If the −whole (or −w) option is specified, the entire line is assumed as an internationalized domain name. Otherwise, **idnconv** recognizes any character sequences having the following properties in the line as internationalized domain names.
   - containing at least one non-ASCII character, and
   - consisting of legal domain name characters (alphabets, digits, hypens), non-ASCII characters and period.

4. convert internationalized domain names to ACE
   For each internationalized domain name found in the line, **idnconv** converts the name to ACE. The details about the conversion procedure is:

   4.1. delimiter mapping
   Substibute certain characters specified as domain name delimiter with period.

   4.2. local mapping
   Perform local mapping. If the local mapping is specified by command line option −localmap, the specified mapping rule is applied. Otherwise, find the mapping rule from the configuration file which matches to the TLD of the name, and perform mapping according to the matched rule.
   This step is skipped if the −nolocalmap (or −L) option is specified.

   4.3. NAMEPREP
   Perform name preparation (NAMEPREP). Mapping, normalization, prohibited character checking, unassigned codepoint checking, bidirectional character checking are done in that order. If the prohibited character check, unassigned codepoint check, or bidi character check fails, the normal conversion procedure aborts.
   This step is skipped if the −nonameprep (or −N) option is specified.

   4.4. ASCII character checking
   Checks ASCII range character in the domain name. the normal conversion procedure aborts, if the domain name has a label beginning or end with hyphen (U+002D) or it contains ASCII range character except for alphanumeric and hyphen,
   This step is skipped if the −noasciicheck (or −A) option is specified.

   4.5. ACE conversion
   Convert the string to ACE.

   4.6. label length checking
   The normal conversion procedure aborts, if the domain name has an empty label or too long label (64 characters or more).
   This step is skipped if the −nolengthcheck option is specified.

5. output the result

REVERSE CONVERSION MECHANISM

This is like the normal conversion mechanism, but they are not symmetric. **idnconv** does its job for each line.

1. read a line from input text

2. convert the line to UTF-8
   idnconv converts the line from local encoding to UTF-8.

3. find internationalized domain names
   If the −whole (or −w) option is specified, the entire line is assumed as an internationalized domain name. Otherwise, **idnconv** decodes any valid ASCII domain names including ACE names in the line.

4. convert domain names to local encoding
   Then, **idnconv** decodes the domain names. The decode procedure consists of the following steps.

   4.1. Delimiter mapping
   Substibute certain characters specified as domain name delimiter with period.

   4.2. NAMEPREP
   Perform name preparation (NAMEPREP) for each label in the domain name. Mapping, normalization, prohibited character checking, unassigned codepoint checking, bidirectional character checking are done in that order. If the prohibited character check, unassigned codepoint check, or bidi character check fails, disqualified labels are restored to original input strings and further conversion on those labels are not performed.
   This step is skipped if the −nonameprep (or −N) option is specified.

   4.3. ACE conversion
   Convert the string from ACE to UTF-8.

   4.4. Round trip checkning
   For each label, perform the normal conversion and compare it with the result of the step 4.2. This check succeeds, if they are equivalent strings. In case of failure, disqualified labels are restored to original input strings and further conversion on those labels are not performed.
   This step is skipped if the −noroundtripcheck option is specified.

   4.5. local encoding conversion
   Convert the result of the step 4.3. from UTF-8 to local encoding. If a label in the domain name contains a character which cannot be represented in the local encoding, the label is restored to the original input string.

5. output the result

FILE MANAGEMENT

Maybe the best way to manage named.conf or zone master files that contains internationalized domain name is to keep them in your local codeset so that they can be edited with your favorite editor, and generate a version in the IDN encoding using **idnconv**.

'make' is a convenient tool for this purpose. Suppose the local codeset version has suffix '.lc', and its ACE version has suffix '.ace'. The following Makefile enables you to generate ACE version from local codeset version by just typing 'make'.

```
.SUFFIXES: .lc .ace
.lc.ace:
      idnconv -in $(LOCALCODE) -out $(IDNCODE) \
          $(IDNCONVOPT) $< > $@

LOCALCODE = EUC-JP
IDNCODE = Punycode
```

```
IDNCONVOPT =

DESTFILES = db.zone1.ace db.zone2.ace

all: $(DESTFILES)
```

SEE ALSO

idn.conf(5), iconv(3)

BUGS

The automatic input-code selection depends on your system, and sometimes it cannot guess or guess wrong. It is better to explicitly specify it using −in option.

NAME

ifnames − Extract CPP conditionals from a set of files

SYNOPSIS

ifnames [*OPTION*] ... [*FILE*] ...

DESCRIPTION

Scan all of the C source FILES (or the standard input, if none are given) and write to the standard output a sorted list of all the identifiers that appear in those files in ‘#if’, ‘#elif’, ‘#ifdef’, or ‘#ifndef’ directives. Print each identifier on a line, followed by a space-separated list of the files in which that identifier occurs.

−h, **−−help**
: print this help, then exit

−V, **−−version**
: print version number, then exit

AUTHOR

Written by David J. MacKenzie and Paul Eggert.

REPORTING BUGS

Report bugs to <bug-autoconf@gnu.org>.

SEE ALSO

autoconf(1), **automake**(1), **autoreconf**(1), **autoupdate**(1), **autoheader**(1), **autoscan**(1), **config.guess**(1), **config.sub**(1), **ifnames**(1), **libtool**(1).

The full documentation for **ifnames** is maintained as a Texinfo manual. If the **info** and **ifnames** programs are properly installed at your site, the command

info ifnames

should give you access to the complete manual.

NAME

indent — indent and format C program source

SYNOPSIS

indent [*input-file* [*output-file*]] [**-bacc** | **-nbacc**] [**-bad** | **-nbad**]
[**-bap** | **-nbap**] [**-bbb** | **-nbbb**] [**-bc** | **-nbc**] [**-bl**] [**-br**] [**-c***n*] [**-cd***n*]
[**-cdb** | **-ncdb**] [**-ce** | **-nce**] [**-ci***n*] [**-cli***n*] [**-d***n*] [**-di***n*] [**-fc1** | **-nfc1**]
[**-i***n*] [**-ip** | **-nip**] [**-l***n*] [**-lc***n*] [**-lp** | **-nlp**] [**-npro**] [**-pcs** | **-npcs**]
[**-psl** | **-npsl**] [**-sc** | **-nsc**] [**-sob** | **-nsob**] [**-st**] [**-troff**] [**-v** | **-nv**]

DESCRIPTION

indent is a C program formatter. It reformats the C program in the *input-file* according to the switches. The switches which can be specified are described below. They may appear before or after the file names.

NOTE: If you only specify an *input-file*, the formatting is done 'in-place', that is, the formatted file is written back into *input-file* and a backup copy of *input-file* is written in the current directory. If *input-file* is named '/blah/blah/file', the backup file is named file.BAK.

If *output-file* is specified, **indent** checks to make sure it is different from *input-file*.

The options listed below control the formatting style imposed by **indent**.

-bacc, **-nbacc**
: If **-bacc** is specified, a blank line is forced around every conditional compilation block. For example, in front of every #ifdef and after every #endif. Other blank lines surrounding such blocks will be swallowed. Default: **-nbacc**.

-bad, **-nbad**
: If **-bad** is specified, a blank line is forced after every block of declarations. Default: **-nbad**.

-bap, **-nbap**
: If **-bap** is specified, a blank line is forced after every procedure body. Default: **-nbap**.

-bbb, **-nbbb**
: If **-bbb** is specified, a blank line is forced before every block comment. Default: **-nbbb**.

-bc, **-nbc**
: If **-bc** is specified, then a newline is forced after each comma in a declaration. **-nbc** turns off this option. Default: **-bc**.

-br, **-bl**
: Specifying **-bl** lines up compound statements like this:

```
if (...)
{
  code
}
```

Specifying **-br** (the default) makes them look like this:

```
if (...) {
  code
}
```

-bs, **-nbs**
: If **-bs** is specified, a blank is forced after **sizeof**. Default: **-nbs**.

-c*n*
: The column in which comments on code start. Default: **-c33**.

-cd*n*
: The column in which comments on declarations start. The default is for these comments to start in the same column as those on code.

-cdb, **-ncdb** Enables (disables) the placement of comment delimiters on blank lines. With this option enabled, comments look like this:

```
/*
 * this is a comment
 */
```

Rather than like this:

```
/* this is a comment */
```

This only affects block comments, not comments to the right of code. Default: **-cdb**.

-ce, **-nce** Enables (disables) forcing ‘else’s to cuddle up to the immediately preceding ‘}’. Default: **-ce**.

-ci*n* Sets the continuation indent to be *n*. Continuation lines will be indented that far from the beginning of the first line of the statement. Parenthesized expressions have extra indentation added to indicate the nesting, unless **-lp** is in effect. **-ci** defaults to the same value as **-i**.

-cli*n* Causes case labels to be indented *n* tab stops to the right of the containing **switch** statement. **-cli0.5** causes case labels to be indented half a tab stop. Default: **-cli0**.

-d*n* Controls the placement of comments which are not to the right of code. For example, **-d1** means that such comments are placed one indentation level to the left of code. Specifying the default **-d0** lines up these comments with the code. See the section on comment indentation below.

-di*n* Specifies the indentation, in character positions, from a declaration keyword to the following identifier. Default: **-di16**.

-dj, **-ndj** **-dj** left justifies declarations. **-ndj** indents declarations the same as code. Default: **-ndj**.

-ei, **-nei** Enables (disables) special **else-if** processing. If it’s enabled, an **if** following an **else** will have the same indentation as the preceding **if** statement. Default: **-ei**.

-eei, **-neei** Enables (disables) extra indentation on continuation lines of the expression part of **if** and **while** statements. These continuation lines will be indented one extra level. Default: **-neei**.

-fc1, **-nfc1** Enables (disables) the formatting of comments that start in column 1. Often, comments whose leading ‘/’ is in column 1 have been carefully hand formatted by the programmer. In such cases, **-nfc1** should be used. Default: **-fc1**.

-i*n* The number of spaces for one indentation level. Default: **-i8**.

-ip, **-nip** Enables (disables) the indentation of parameter declarations from the left margin. Default: **-ip**.

-l*n* Maximum length of an output line. Default: **-l78**.

-lp, **-nlp** Lines up code surrounded by parenthesis in continuation lines. If a line has a left paren which is not closed on that line, then continuation lines will be lined up to start at the character position just after the left paren. For example, here is how a piece of continued code looks with **-nlp** in effect:

```
p1 = first_procedure(second_procedure(p2, p3),
  third_procedure(p4,p5));
```

With **-lp** in effect (the default) the code looks somewhat clearer:

```
p1 = first_procedure(second_procedure(p2, p3),
                     third_procedure(p4,p5));
```

Inserting two more newlines we get:

```
p1 = first_procedure(second_procedure(p2,
                                      p3),
                     third_procedure(p4
                                     p5));
```

-npro Causes the profile files, '`./.indent.pro`' and '`~/.indent.pro`', to be ignored.

-pcs, **-npcs** If true (**-pcs**) all procedure calls will have a space inserted between the name and the '('. Default: **-npcs**.

-psl, **-npsl** If true (**-psl**) the names of procedures being defined are placed in column 1 − their types, if any, will be left on the previous lines. Default: **-psl**.

-sc, **-nsc** Enables (disables) the placement of asterisks ('*'s) at the left edge of all comments. Default: **-sc**.

-sob, **-nsob** If **-sob** is specified, indent will swallow optional blank lines. You can use this to get rid of blank lines after declarations. Default: **-nsob**.

-st Causes **indent** to take its input from stdin, and put its output to stdout.

-T*typename* Adds *typename* to the list of type keywords. Names accumulate: **-T** can be specified more than once. You need to specify all the typenames that appear in your program that are defined by **typedef** − nothing will be harmed if you miss a few, but the program won't be formatted as nicely as it should. This sounds like a painful thing to have to do, but it's really a symptom of a problem in C: **typedef** causes a syntactic change in the language and **indent** can't find all instances of **typedef**.

-troff Causes **indent** to format the program for processing by `troff`(1). It will produce a fancy listing in much the same spirit as `vgrind`(1). If the output file is not specified, the default is standard output, rather than formatting in place.

-v, **-nv** **-v** turns on 'verbose' mode; **-nv** turns it off. When in verbose mode, **indent** reports when it splits one line of input into two or more lines of output, and gives some size statistics at completion. Default: **-nv**.

You may set up your own 'profile' of defaults to **indent** by creating a file called `.indent.pro` in your login directory and/or the current directory and including whatever switches you like. A '.indent.pro' in the current directory takes precedence over the one in your login directory. If **indent** is run and a profile file exists, then it is read to set up the program's defaults. Switches on the command line, though, always override profile switches. The switches should be separated by spaces, tabs or newlines.

Comments

'*Box*' *comments*. **indent** assumes that any comment with a dash or star immediately after the start of comment (that is, '/*−' or '/**') is a comment surrounded by a box of stars. Each line of such a comment is left unchanged, except that its indentation may be adjusted to account for the change in indentation of the first line of the comment.

Straight text. All other comments are treated as straight text. **indent** fits as many words (separated by blanks, tabs, or newlines) on a line as possible. Blank lines break paragraphs.

Comment indentation

If a comment is on a line with code it is started in the 'comment column', which is set by the **-c***n* command line parameter. Otherwise, the comment is started at *n* indentation levels less than where code is currently being placed, where *n* is specified by the **-d***n* command line parameter. If the code on a line extends past the comment column, the comment starts further to the right, and the right margin may be automatically extended in extreme cases.

Preprocessor lines

In general, **indent** leaves preprocessor lines alone. The only reformatting that it will do is to straighten up trailing comments. It leaves embedded comments alone. Conditional compilation (**#ifdef...#endif**) is recognized and **indent** attempts to correctly compensate for the syntactic peculiarities introduced.

C syntax

indent understands a substantial amount about the syntax of C, but it has a 'forgiving' parser. It attempts to cope with the usual sorts of incomplete and misformed syntax. In particular, the use of macros like:

```
#define forever for(;;)
```

is handled properly.

ENVIRONMENT

indent uses the HOME environment variable.

FILES

`./.indent.pro` profile file
`~/.indent.pro` profile file

HISTORY

The **indent** command appeared in 4.2BSD.

BUGS

indent has even more switches than ls(1).

A common mistake that often causes grief is typing:

```
indent *.c
```

to the shell in an attempt to indent all the C programs in a directory. This is probably a bug, not a feature.

NAME

info − read Info documents

SYNOPSIS

info [*OPTION*]... [*MENU-ITEM*...]

DESCRIPTION

Read documentation in Info format.

OPTIONS

−−apropos=*STRING*
: look up STRING in all indices of all manuals.

−d, **−−directory**=*DIR*
: add DIR to INFOPATH.

−−dribble=*FILENAME*
: remember user keystrokes in FILENAME.

−f, **−−file**=*FILENAME*
: specify Info file to visit.

−h, **−−help**
: display this help and exit.

−−index−search=*STRING*
: go to node pointed by index entry STRING.

−n, **−−node**=*NODENAME*
: specify nodes in first visited Info file.

−o, **−−output**=*FILENAME*
: output selected nodes to FILENAME.

−R, **−−raw−escapes**
: output "raw" ANSI escapes (default).

−−no−raw−escapes
: output escapes as literal text.

−−restore=*FILENAME*
: read initial keystrokes from FILENAME.

−O, **−−show−options**, **−−usage**
: go to command-line options node.

−−subnodes
: recursively output menu items.

−w, **−−where**, **−−location**
: print physical location of Info file.

−−vi−keys
: use vi-like and less-like key bindings.

−−version
: display version information and exit.

The first non-option argument, if present, is the menu entry to start from; it is searched for in all ‘dir’ files along INFOPATH. If it is not present, info merges all ‘dir’ files and shows the result. Any remaining arguments are treated as the names of menu items relative to the initial node visited.

EXAMPLES

info show top-level dir menu

info emacs
start at emacs node from top-level dir

info emacs buffers
start at buffers node within emacs manual

info −−**show−options** emacs
start at node with emacs' command line options

info **−f** ./foo.info
show file ./foo.info, not searching dir

REPORTING BUGS

Email bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org. Texinfo home page: http://www.gnu.org/software/texinfo/

COPYRIGHT

NAME

infokey − compile customizations for Info

SYNOPSIS

infokey [*OPTION*]... [*INPUT-FILE*]

DESCRIPTION

Compile infokey source file to infokey file. Reads INPUT-FILE (default $HOME/.infokey) and writes compiled key file to (by default) $HOME/.info.

OPTIONS

−−output FILE
: output to FILE instead of $HOME/.info

−−help display this help and exit.

−−version
: display version information and exit.

REPORTING BUGS

Email bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org. Texinfo home page: http://www.gnu.org/software/texinfo/

COPYRIGHT

SEE ALSO

The full documentation for **infokey** is maintained as a Texinfo manual. If the **info** and **infokey** programs are properly installed at your site, the command

info infokey

should give you access to the complete manual.

NAME

innetgr — check netgroup members

SYNOPSIS

innetgr [**-v**] [**-d** *domain*] [**-h** *host*] [**-u** *user*] *netgroup*

DESCRIPTION

innetgr checks if the specified domain, host or user is a member of the given netgroup. This program does not produce any output unless the verbose (**-v**) flag is given.

EXIT STATUS

0 Successful completion.

1 Name not found in group.

2 Invalid usage.

SEE ALSO

netgroup(1), innetgr(3), netgroup(5)

HISTORY

A **innetgr** utility appeared in NetBSD 1.4.

NAME

install-info − update info/dir entries

SYNOPSIS

install-info [*OPTION*]... [*INFO-FILE* [*DIR-FILE*]]

DESCRIPTION

Install or delete dir entries from INFO-FILE in the Info directory file DIR-FILE.

OPTIONS

−−delete
delete existing entries for INFO-FILE from DIR-FILE; don't insert any new entries.

−−dir−file=*NAME*
specify file name of Info directory file. This is equivalent to using the DIR-FILE argument.

−−entry=*TEXT*
insert TEXT as an Info directory entry. TEXT should have the form of an Info menu item line plus zero or more extra lines starting with whitespace. If you specify more than one entry, they are all added. If you don't specify any entries, they are determined from information in the Info file itself.

−−help display this help and exit.

−−info−file=*FILE*
specify Info file to install in the directory. This is equivalent to using the INFO-FILE argument.

−−info−dir=*DIR*
same as **−−dir−file**=*DIR*/dir.

−−item=*TEXT*
same as **−−entry** TEXT. An Info directory entry is actually a menu item.

−−quiet
suppress warnings.

−−remove
same as **−−delete**.

−−section=*SEC*
put this file's entries in section SEC of the directory. If you specify more than one section, all the entries are added in each of the sections. If you don't specify any sections, they are determined from information in the Info file itself.

−−version
display version information and exit.

REPORTING BUGS

Email bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org. Texinfo home page: http://www.gnu.org/software/texinfo/

COPYRIGHT

SEE ALSO

The full documentation for **install-info** is maintained as a Texinfo manual. If the **info** and **install-info** programs are properly installed at your site, the command

info install-info

should give you access to the complete manual.

NAME

install — install binaries

SYNOPSIS

install [**-Ubcprs**] [**-B** *suffix*] [**-D** *destdir*] [**-f** *flags*] [**-M** *metalog*] [**-T** *tags*]
[**-a** *command*] [**-m** *mode*] [**-N** *dbdir*] [**-o** *owner*] [**-g** *group*]
[**-l** *linkflags*] [**-h** *hash*] [**-S** *stripflag*] *file1 file2*
install [**-Ubcprs**] [**-B** *suffix*] [**-D** *destdir*] [**-f** *flags*] [**-M** *metalog*] [**-T** *tags*]
[**-a** *command*] [**-m** *mode*] [**-N** *dbdir*] [**-o** *owner*] [**-g** *group*]
[**-l** *linkflags*] [**-h** *hash*] [**-S** *stripflag*] *file1 ... fileN directory*
install -d [**-Up**] [**-D** *destdir*] [**-M** *metalog*] [**-T** *tags*] [**-a** *command*] [**-m** *mode*]
[**-N** *dbdir*] [**-o** *owner*] [**-g** *group*] *directory ...*

DESCRIPTION

The file(s) are copied (or linked if the **-l** option is specified) to the target file or directory. If the destination is a directory, then the *file* is copied into *directory* with its original filename. If the target file already exists, it is either renamed to *file.old* if the **-b** option is given or overwritten if permissions allow; an alternate backup suffix may be specified via the **-B** option's argument.

-a *command*
: Run *command* on the target after installation and stripping (**-s**), but before ownership, permissions or timestamps are set and before renaming (**-r**) occurs. *command* is invoked via the sh(1) shell, allowing a single **-a** argument be to specified to **install** which the shell can then tokenize.

-b
: Backup any existing files before overwriting them by renaming them to *file.old*. *See* **-B** for specifying a different backup suffix.

-B *suffix*
: Use *suffix* as the backup suffix if **-b** is given. If *suffix* contains a '%' sign, a numbered backup will be performed, and the %-pattern will be expanded using sprintf(3), given an integer counter as the backup number. The counter used starts from 0, and the first available name resulting from the expansion is used.

-c
: Copy the file. This is the default behavior; the flag is maintained for backwards compatibility only.

-d
: Create directories. Missing parent directories are created as required.

-D *destdir*
: Specify the DESTDIR (top of the file hierarchy) that the items are installed in to. If **-M** *metalog* is in use, a leading string of "*destdir*" will be removed from the file names logged to the *metalog*. This option does not affect where the actual files are installed.

-f *flags*
: Specify the target's file flags. (See chflags(1) for a list of possible flags and their meanings.)

-g *group*
: Specify a group.

-h *hash*
: When copying, calculate the digest of the files with *hash* to store in the **-M** *metalog*. Supported digests:

  none No hash. This is the default.

  md5 The MD5 cryptographic message digest.

rmd160 The RMD-160 cryptographic message digest.

sha1 The SHA-1 cryptographic message digest.

-l *linkflags*
Instead of copying the file make a link to the source. The type of the link is determined by the *linkflags* argument. Valid *linkflags* are: *a* (absolute), *r* (relative), *h* (hard), *s* (symbolic), *m* (mixed). Absolute and relative have effect only for symbolic links. Mixed links are hard links for files on the same filesystem, symbolic otherwise.

-M *metalog*
Write the metadata associated with each item installed to *metalog* in an mtree(8) "full path" specification line. The metadata includes: the file name and file type, and depending upon other options, the owner, group, file flags, modification time, and tags.

-m *mode*
Specify an alternative mode. The default mode is set to rwxr-xr-x (0755). The specified mode may be either an octal or symbolic value; see chmod(1) for a description of possible mode values.

-N *dbdir*
Use the user database text file master.passwd and group database text file group from *dbdir*, rather than using the results from the system's getpwnam(3) and getgrnam(3) (and related) library calls.

-o *owner*
Specify an owner.

-p Preserve the source files access and modification times.

-r Install to a temporary file and then rename the file to its final destination name. This can be used for precious files, to avoid truncation of the original when error conditions (filesystem full etc.) occur.

-s **install** exec's the command strip(1) to strip binaries so that install can be portable over a large number of systems and binary types. If the environment variable STRIP is set, it is used as the strip(1) program.

-S *stripflags*
install passes *stripflags* as option arguments to strip(1). When **-S** is used, strip(1) is invoked via the sh(1) shell, allowing a single **-S** argument be to specified to **install** which the shell can then tokenize. Normally, **install** invokes strip(1) directly. This flag implies **-s**.

-T *tags*
Specify the mtree(8) tags to write out for the file when using **-M** *metalog*.

-U Indicate that install is running unprivileged, and that it should not try to change the owner, the group, or the file flags of the destination. The information that would have been updated can be stored in a log file with **-M** *metalog*.

By default, **install** preserves all file flags, with the exception of the ''nodump'' flag.

The **install** utility attempts to prevent copying a file onto itself.

Installing /dev/null creates an empty file.

EXIT STATUS

The **install** utility exits 0 on success, and >0 if an error occurs.

ENVIRONMENT

`STRIP` The program used to strip installed binaries when the **`-s`** option is used. If unspecified, `/usr/bin/strip` is used.

SEE ALSO

`chflags`(1), `chgrp`(1), `chmod`(1), `cp`(1), `mv`(1), `strip`(1), `chown`(8), `mtree`(8)

HISTORY

The **`install`** utility appeared in 4.2BSD.

NAME

ipcrm — remove the specified message queues, semaphore sets, and shared memory segments

SYNOPSIS

ipcrm [**-M** *shmkey*] [**-m** *shmid*] [**-Q** *msgkey*] [**-q** *msqid*] [**-S** *semkey*] [**-s** *semid*]
...

DESCRIPTION

ipcrm removes the specified message queues, semaphores, and shared memory segments. These System V IPC objects can be specified by their creation ID or any associated key.

The following options are used to specify which IPC objects will be removed. Any number and combination of these options can be used:

-M *shmkey*
: Mark the shared memory segment associated with key *shmkey* for removal. This marked segment will be destroyed after the last detach.

-m *shmid*
: Mark the shared memory segment associated with ID *shmid* for removal. This marked segment will be destroyed after the last detach.

-Q *msgkey*
: Remove the message queue associated with key *msgkey* from the system.

-q *msqid*
: Remove the message queue associated with the ID *msqid* from the system.

-S *semkey*
: Remove the semaphore set associated with key *semkey* from the system.

-s *semid*
: Removes the semaphore set associated with ID *semid* from the system.

If the *id* or *key* argument is “all” then all entries of the appropriate type are removed.

The identifiers and keys associated with these System V IPC objects can be determined by using ipcs(1).

SEE ALSO

ipcs(1), shmat(2), shmctl(2), shmdt(2), shmget(2)

NAME

ipcs — report System V interprocess communication facilities status

SYNOPSIS

ipcs [**-abcmopqstMQST**] [**-C** *system*] [**-N** *core*]

DESCRIPTION

The **ipcs** program provides information on System V interprocess communication (IPC) facilities on the system.

The options are as follows:

-a Show the maximum amount of information possible when displaying active semaphores, message queues, and shared memory segments. (This is shorthand for specifying the **-b**, **-c**, **-o**, **-p**, and **-t** options.)

-b Show the maximum allowed sizes for active semaphores, message queues, and shared memory segments. The “maximum allowed size” is the maximum number of bytes in a message on a message queue, the size of a shared memory segment, or the number of semaphores in a set of semaphores.

-c Show the creator’s name and group for active semaphores, message queues, and shared memory segments.

-m Display information about active shared memory segments.

-o Show outstanding usage for active message queues, and shared memory segments. The “outstanding usage” is the number of messages in a message queue, or the number of processes attached to a shared memory segment.

-p Show the process ID information for active semaphores, message queues, and shared memory segments. The “process ID information” is the last process to send a message to or receive a message from a message queue, the process that created a semaphore, or the last process to attach or detach a shared memory segment.

-q Display information about active message queues.

-s Display information about active semaphores.

-t Show access times for active semaphores, message queues, and shared memory segments. The access times is the time of the last control operation on an IPC object, the last send or receive of a message, the last attach or detach of a shared memory segment, or the last operation on a semaphore.

-C *system*
Extract the name list from the specified system instead of the default “`/netbsd`”.

-M Display system information about shared memory.

-N *core*
Extract values associated with the name list from the specified core instead of the default “`/dev/kmem`”. and semaphores.

-Q Display system information about messages queues.

-S Display system information about semaphores.

-T Display system information about shared memory, message queues and semaphores. (This is shorthand for specifying the **-M**, **-Q**, and **-S** options.)

If none of the **-M**, **-m**, **-Q**, **-q**, **-S**, **-s**, or **-T** options are specified, information about all active IPC facilities is listed.

RESTRICTIONS

System data structures may change while **ipcs** is running; the output of **ipcs** is not guaranteed to be consistent.

FILES

`/dev/kmem`	default kernel memory
`/netbsd`	default system name list

SEE ALSO

`ipcrm`(1), `shmat`(2), `shmctl`(2), `shmdt`(2), `shmget`(2)

AUTHORS

Thorsten Lockert ⟨tholo@sigmasoft.com⟩

BUGS

This manual page is woefully incomplete, because it does not at all attempt to explain the information printed by **ipcs**.

NAME

ipftest − test packet filter rules with arbitrary input.

SYNOPSIS

ipftest [**−6bCdDoRvx**] [**−F** input-format] [**−i** <filename>] [**−I** interface] [**−l** <filename>] [**−N** <filename>] [**−P** <filename>] [**−r** <filename>] [**−S** <ip_address>] [**−T** <optionlist>]

DESCRIPTION

ipftest is provided for the purpose of being able to test a set of filter rules without having to put them in place, in operation and proceed to test their effectiveness. The hope is that this minimises disruptions in providing a secure IP environment.

ipftest will parse any standard ruleset for use with **ipf**, **ipnat** and/or **ippool** and apply input, returning output as to the result. However, **ipftest** will return one of three values for packets passed through the filter: pass, block or nomatch. This is intended to give the operator a better idea of what is happening with packets passing through their filter ruleset.

At least one of **−N**, **-P** or **−r** must be specified.

OPTIONS

−6 Use IPv6.

−b Cause the output to be a brief summary (one-word) of the result of passing the packet through the filter; either "pass", "block" or "nomatch". This is used in the regression testing.

−C Force the checksums to be (re)calculated for all packets being input into **ipftest**. This may be necessary if pcap files from tcpdump are being fed in where there are partial checksums present due to hardware offloading.

−d Turn on filter rule debugging. Currently, this only shows you what caused the rule to not match in the IP header checking (addresses/netmasks, etc).

−D Dump internal tables before exiting. This excludes log messages.

−F This option is used to select which input format the input file is in. The following formats are available: etherfind, hex, pcap, snoop, tcpdump,text.

etherfind

The input file is to be text output from etherfind. The text formats which are currently supported are those which result from the following etherfind option combinations:

```
etherfind -n
etherfind -n -t
```

hex The input file is to be hex digits, representing the binary makeup of the packet. No length correction is made, if an incorrect length is put in the IP header. A packet may be broken up over several lines of hex digits, a blank line indicating the end of the packet. It is possible to specify both the interface name and direction of the packet (for filtering purposes) at the start of the line using this format: [direction,interface] To define a packet going in on le0, we would use **[in,le0]** - the []'s are required and part of the input syntax.

pcap The input file specified by **−i** is a binary file produced using libpcap (i.e., tcpdump version 3). Packets are read from this file as being input (for rule purposes). An interface maybe specified using **−I**.

snoop The input file is to be in "snoop" format (see RFC 1761). Packets are read from this file and used as input from any interface. This is perhaps the most useful input type, currently.

tcpdump

The input file is to be text output from tcpdump. The text formats which are currently supported are those which result from the following tcpdump option combinations:

```
tcpdump -n
tcpdump -nq
```

```
tcpdump -nqt
tcpdump -nqtt
tcpdump -nqte
```

text The input file is in **ipftest** text input format. This is the default if no **−F** argument is specified. The format used is as follows:

```
"in"|"out" "on" if ["tcp"|"udp"|"icmp"]
        srchost[,srcport] dsthost[,destport] [FSRPAU]
```

This allows for a packet going "in" or "out" of an interface (if) to be generated, being one of the three main protocols (optionally), and if either TCP or UDP, a port parameter is also expected. If TCP is selected, it is possible to (optionally) supply TCP flags at the end. Some examples are:

```
# a UDP packet coming in on le0
in on le0 udp 10.1.1.1,2210 10.2.1.5,23
# an IP packet coming in on le0 from localhost - hmm :)
in on le0 localhost 10.4.12.1
# a TCP packet going out of le0 with the SYN flag set.
out on le0 tcp 10.4.12.1,2245 10.1.1.1,23 S
```

−i <filename>
Specify the filename from which to take input. Default is stdin.

−I <interface>
Set the interface name (used in rule matching) to be the name supplied. This is useful where it is not otherwise possible to associate a packet with an interface. Normal "text packets" can override this setting.

−l <filename>
Dump log messages generated during testing to the specified file.

−N <filename>
Specify the filename from which to read NAT rules in **ipnat**(5) format.

−o Save output packets that would have been written to each interface in a file /tmp/*interface_name* in raw format.

−P <filename>
Read IP pool configuration information in **ippool**(5) format from the specified file.

−r <filename>
Specify the filename from which to read filter rules in **ipf**(5) format.

−R Don’t attempt to convert IP addresses to hostnames.

−S <ip_address>
The IP address specifived with this option is used by ipftest to determine whether a packet should be treated as "input" or "output". If the source address in an IP packet matches then it is considered to be inbound. If it does not match then it is considered to be outbound. This is primarily for use with tcpdump (pcap) files where there is no in/out information saved with each packet.

−T <optionlist>
This option simulates the run-time changing of IPFilter kernel variables available with the **−T** option of **ipf**. The optionlist parameter is a comma separated list of tuning commands. A tuning command is either "list" (retrieve a list of all variables in the kernel, their maximum, minimum and current value), a single variable name (retrieve its current value) and a variable name with a following assignment to set a new value. See **ipf**(8) for examples.

−v Verbose mode. This provides more information about which parts of rule matching the input packet passes and fails.

−x Print a hex dump of each packet before printing the decoded contents.

SEE ALSO

ipf(5), ipf(8), tcpdump(8),

BUGS

Not all of the input formats are sufficiently capable of introducing a wide enough variety of packets for them to be all useful in testing.

NAME

ipresend − resend IP packets out to network

SYNOPSIS

ipresend [**−EHPRSTX**] [**−d** <device>] [**−g** <*gateway*>] [**−m** <*MTU*>] [**−r** <*filename*>]

DESCRIPTION

ipresend was designed to allow packets to be resent, once captured, back out onto the network for use in testing. *ipresend* supports a number of different file formats as input, including saved snoop/tcpdump binary data.

OPTIONS

−d <interface>

Set the interface name to be the name supplied. This is useful with the **−P, −S, −T** and **−E** options, where it is not otherwise possible to associate a packet with an interface. Normal "text packets" can override this setting.

−g <gateway>

Specify the hostname of the gateway through which to route packets. This is required whenever the destination host isn't directly attached to the same network as the host from which you're sending.

−m <MTU>

Specify the MTU to be used when sending out packets. This option allows you to set a fake MTU, allowing the simulation of network interfaces with small MTU's without setting them so.

−r <filename>

Specify the filename from which to take input. Default is stdin.

−E The input file is to be text output from etherfind. The text formats which are currently supported are those which result from the following etherfind option combinations:

etherfind -n
etherfind -n -t

−H The input file is to be hex digits, representing the binary makeup of the packet. No length correction is made, if an incorrect length is put in the IP header.

−P The input file specified by **−i** is a binary file produced using libpcap (i.e., tcpdump version 3). Packets are read from this file as being input (for rule purposes).

−R When sending packets out, send them out "raw" (the way they came in). The only real significance here is that it will expect the link layer (i.e. ethernet) headers to be prepended to the IP packet being output.

−S The input file is to be in "snoop" format (see RFC 1761). Packets are read from this file and used as input from any interface. This is perhaps the most useful input type, currently.

−T The input file is to be text output from tcpdump. The text formats which are currently supported are those which result from the following tcpdump option combinations:

tcpdump -n
tcpdump -nq
tcpdump -nqt
tcpdump -nqtt
tcpdump -nqte

−X The input file is composed of text descriptions of IP packets.

SEE ALSO

ipftest(1), ipsend(1), iptest(1), bpf(4), ipsend(5), tcpdump(8)

DIAGNOSTICS

Needs to be run as root.

BUGS

Not all of the input formats are sufficiently capable of introducing a wide enough variety of packets for them to be all useful in testing. If you find any, please send email to me at darrenr@pobox.com

NAME

ipsend − sends IP packets

SYNOPSIS

ipsend [**−dITUv**] [**−i** <interface>] [**−f** <*offset*>] [**−g** <*gateway*>] [**−m** <*MTU*>] [**−o** <*option*>] [**−P** <protocol>] [**−s** <*source*>] [**−t** <*dest. port*>] [**−w** <*window*>] <destination> [TCP-flags]

DESCRIPTION

ipsend can be compiled in two ways. The first is used to send one-off packets to a destination host, using command line options to specify various attributes present in the headers. The *destination* must be given as the last command line option, except for when TCP flags are specified as a combination of A, S, F, U, P and R, last.

The other way it may be compiled, with DOSOCKET defined, is to allow an attempt at making a TCP connection using a with ipsend resending the SYN packet as per the command line options.

OPTIONS

−d enable debugging mode.

−f <offset>
The *-f* allows the IP offset field in the IP header to be set to an arbitrary value, which can be specified in decimal or hexadecimal.

−g <gateway>
Specify the hostname of the gateway through which to route packets. This is required whenever the destination host isn't directly attached to the same network as the host from which you're sending.

−i <interface>
Set the interface name to be the name supplied.

−m <MTU>
Specify the MTU to be used when sending out packets. This option allows you to set a fake MTU, allowing the simulation of network interfaces with small MTU's without setting them so.

−o <option>
Specify options to be included at the end of the IP header. An EOL option is automatically appended and need not be given. If an option would also have data associated with it (source as an IP# for a lsrr option), then this will not be initialised.

−s <source>
Set the source address in the packet to that provided - maybe either a hostname or IP#.

−t <dest.**port>**
Set the destination port for TCP/UDP packets.

−w <window>
Set the window size for TCP packets.

−I Set the protocol to ICMP.

−P <protocol>
Set the protocol to the value given. If the parameter is a name, the name is looked up in the */etc/protocols* file.

−T Set the protocol to TCP.

−U Set the protocol to UDP.

−v enable verbose mode.

SEE ALSO

ipresend(1), iptest(1), bpf(4), protocols(5), ipsend(5)

DIAGNOSTICS

Needs to be run as root.

BUGS

If you find any, please send email to me at darrenr@pobox.com

NAME

iptest − automatically generate a packets to test IP functionality

SYNOPSIS

iptest [**−1234567**] [**−d** <device>] [**−g** <gateway>] [**−m** <*MTU*>] [**−p** <*pointtest*>] [**−s** <*source*>] <destination>

DESCRIPTION

iptest ...

OPTIONS

−1 Run IP test group #1. This group of tests generates packets with the IP header fields set to invalid values given other packet characteristics. The point tests are: 1 (ip_hl < ip_len), 2 (ip_hl > ip_len), 3 (ip_v < 4), 4 (ip_v > 4), 5 (ip_len < packetsize, long packets), 6 (ip_len > packet size, short packets), 7 (Zero length fragments), 8 (packet > 64k after reassembly), 9 (IP offset with MSB set), 10 (ttl variations).

−2 Run IP test group #2. This group of tests generates packets with the IP options constructed with invalid values given other packet characteristics. The point tests are: 1 (option length > packet length), 2 (option length = 0).

−3 Run IP test group #3. This group of tests generates packets with the ICMP header fields set to non-standard values. The point tests are: 1 (ICMP types 0-31 & 255), 2 (type 3 & code 0 - 31), 3 (type 4 & code 0, 127, 128, 255), 4 (type 5 & code 0, 127, 128, 255), 5 (types 8-10,13-18 with codes 0, 127, 128 and 255), 6 (type 12 & code 0, 127, 128, 129, 255) and 7 (type 3 & codes 9-10, 13-14 and 17-18 - shortened packets).

−4 Run IP test group #4. This group of tests generates packets with the UDP header fields set to non-standard values. The point tests are: 1 (UDP length > packet size), 2 (UDP length < packetsize), 3 (sport = 0, 1, 32767, 32768, 65535), 4 (dport = 0, 1, 32767, 32768, 65535) and 5 (sizeof(struct ip) <= MTU <= sizeof(struct udphdr) + sizeof(struct ip)).

−5 Run IP test group #5. This group of tests generates packets with the TCP header fields set to non-standard values. The point tests are: 1 (TCP flags variations, all combinations), 2 (seq = 0, 0x7fffffff, 0x8000000, 0xa0000000, 0xffffffff), 3 (ack = 0, 0x7fffffff, 0x8000000, 0xa0000000, 0xffffffff), 4 (SYN packet with window of 0, 32768, 65535), 5 (set urgent pointer to 1, 0x7fff, 0x8000, 0xffff), 6 (data offset), 7 (sport = 0, 1, 32767, 32768, 65535) and 8 (dport = 0, 1, 32767, 32768, 65535).

−6 Run IP test group #6. This test generates a large number of fragments in an attempt to exhaust the network buffers used for holding packets for later reassembly. WARNING: this may crash or cause serious performance degradation to the target host.

−7 Run IP test group #7. This test generates 1024 random IP packets with only the IP version, checksum, length and IP offset field correct.

−d <interface>
Set the interface name to be the name supplied.

−g <gateway>
Specify the hostname of the gateway through which to route packets. This is required whenever the destination host isn't directly attached to the same network as the host from which you're sending.

−m <MTU>
Specify the MTU to be used when sending out packets. This option allows you to set a fake MTU, allowing the simulation of network interfaces with small MTU's without setting them so.

−p <test>
Run a...

SEE ALSO

ipresend(1), ipsend(1), bpf(4), ipsend(5)

DIAGNOSTICS

Only one of the numeric test options may be given when *iptest* is run.

Needs to be run as root.

BUGS

If you find any, please send email to me at darrenr@pobox.com

NAME

join — relational database operator

SYNOPSIS

join [**-a** *file_number* | **-v** *file_number*] [**-e** *string*] [**-j** *file_number field*] [**-o** *list*] [**-t** *char*] [**-1** *field*] [**-2** *field*] *file1 file2*

DESCRIPTION

The join utility performs an ‘‘equality join’’ on the specified files and writes the result to the standard output. The ‘‘join field’’ is the field in each file by which the files are compared. The first field in each line is used by default. There is one line in the output for each pair of lines in *file1* and *file2* which have identical join fields. Each output line consists of the join field, the remaining fields from *file1* and then the remaining fields from *file2*.

The default field separators are tab and space characters. In this case, multiple tabs and spaces count as a single field separator, and leading tabs and spaces are ignored. The default output field separator is a single space character.

Many of the options use file and field numbers. Both file numbers and field numbers are 1 based, i.e. the first file on the command line is file number 1 and the first field is field number 1. The following options are available:

-a *file_number*
: In addition to the default output, produce a line for each unpairable line in file *file_number*. (The argument to **-a** must not be preceded by a space; see the **COMPATIBILITY** section.)

-e *string*
: Replace empty output fields with *string*.

-o *list*
: The **-o** option specifies the fields that will be output from each file for each line with matching join fields. Each element of *list* has the form file_number.field, where *file_number* is a file number and *field* is a field number. The elements of list must be either comma (‘‘,’’) or whitespace separated. (The latter requires quoting to protect it from the shell, or, a simpler approach is to use multiple **-o** options.)

-t *char*
: Use character *char* as a field delimiter for both input and output. Every occurrence of *char* in a line is significant.

-v *file_number*
: Do not display the default output, but display a line for each unpairable line in file *file_number*. The options **-v** *1* and **-v** *2* may be specified at the same time.

-1 *field*
: Join on the *field*’th field of file 1.

-2 *field*
: Join on the *field*’th field of file 2.

When the default field delimiter characters are used, the files to be joined should be ordered in the collating sequence of sort(1), using the **-b** option, on the fields on which they are to be joined, otherwise **join** may not report all field matches. When the field delimiter characters are specified by the **-t** option, the collating sequence should be the same as sort(1) without the **-b** option.

If one of the arguments *file1* or *file2* is ‘‘-’’, the standard input is used.

The **join** utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

For compatibility with historic versions of **join**, the following options are available:

-a In addition to the default output, produce a line for each unpairable line in both file 1 and file 2. (To distinguish between this and **-a** *file_number*, **join** currently requires that the latter not include any white space.)

-j1 *field*
Join on the *field*'th field of file 1.

-j2 *field*
Join on the *field*'th field of file 2.

-j *field* Join on the *field*'th field of both file 1 and file 2.

-o *list ...*
Historical implementations of **join** permitted multiple arguments to the **-o** option. These arguments were of the form ''file_number.field_number'' as described for the current **-o** option. This has obvious difficulties in the presence of files named ''1.2''.

These options are available only so historic shell scripts don't require modification and should not be used.

SEE ALSO

awk(1), comm(1), paste(1), sort(1), uniq(1)

STANDARDS

The **join** command is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

NAME

jot — print sequential or random data

SYNOPSIS

jot [**-cnr**] [**-b** *word*] [**-p** *precision*] [**-s** *string*] [**-w** *word*] [*reps* [*begin* [*end* [*s*]]]]

DESCRIPTION

The **jot** utility is used to print out increasing, decreasing, random, or redundant data (usually numbers) one per line.

The following options are available:

-b *word*
: Just print *word* repetitively.

-c
: This is an abbreviation for **-w %c**.

-n
: Do not print the final newline normally appended to the output.

-p *precision*
: Print only as many digits or characters of the data as indicated by the integer *precision*. In the absence of **-p**, the precision is the greater of the precisions of *begin* and *end*. The **-p** option is overridden by whatever appears in a `printf`(3) conversion following **-w**.

-r
: Generate random data instead of sequential data, the default.

-s *string*
: Print data separated by *string*. Normally, newlines separate data.

-w *word*
: Print *word* with the generated data appended to it. Octal, hexadecimal, exponential, ASCII, zero padded, and right-adjusted representations are possible by using the appropriate `printf`(3) conversion specification inside *word*, in which case the data are inserted rather than appended.

The last four arguments indicate, respectively, the number of data, the lower bound, the upper bound, and the step size or, for random data, the seed. While at least one of them must appear, any of the other three may be omitted, and will be considered as such if given as "-". Any three of these arguments determines the fourth. If four are specified and the given and computed values of *reps* conflict, the lower value is used. If fewer than three are specified, defaults are assigned left to right, except for *s*, which assumes its default unless both *begin* and *end* are given.

Defaults for the four arguments are, respectively, 100, 1, 100, and 1, except that when random data are requested, *s* defaults to a seed depending upon the time of day. *reps* is expected to be an unsigned integer, and if given as zero is taken to be infinite. *begin* and *end* may be given as real numbers or as characters representing the corresponding value in ASCII. The last argument must be a real number.

Random numbers are obtained through `random`(3). The name **jot** derives in part from **iota**, a function in APL.

EXAMPLES

The command:

```
jot 21 -1 1.00
```

prints 21 evenly spaced numbers increasing from −1 to 1.

The command:

```
jot -c 128 0
```

prints the ASCII character set.

The command:

```
jot -w xa%c 26 a
```

prints the strings “xaa” through “xaz”.

The command:

```
jot -r -c 160 a z | rs -g 0 8
```

prints 20 random 8-letter strings.

The command:

```
jot -b y 0
```

is equivalent to `yes`(1).

The command:

```
jot -w %ds/old/new/ 30 2 - 5
```

prints thirty `ed`(1) substitution commands applying to lines 2, 7, 12, etc.

The command:

```
jot 0 9 - -.5
```

prints the stuttering sequence 9, 8, 8, 7, etc.

The command:

```
jot -b x 512 > block
```

creates a file containing exactly 1024 bytes.

The command:

```
expand -`jot -s, - 10 132 4`
```

sets tabs four spaces apart starting from column 10 and ending in column 132.

The command:

```
grep `jot -s "" -b . 80`
```

prints all lines 80 characters or longer.

SEE ALSO

`ed`(1), `expand`(1), `rs`(1), `yes`(1), `printf`(3), `random`(3)

NAME

kdestroy — remove one credental or destroy the current ticket file

SYNOPSIS

kdestroy [**-c** *cachefile*] [**--credential=***principal*] [**--cache=***cachefile*]
[**--no-unlog**] [**--no-delete-v4**] [**--version**] [**--help**]

DESCRIPTION

kdestroy remove one or the current set of tickets.

Supported options:

-credential=*principal*
: remove *principal* from the credential cache if it exists.

-c *cachefile*

-cache=*cachefile*
: The cache file to remove.

--no-unlog
: Do not remove AFS tokens.

--no-delete-v4
: Do not remove v4 tickets.

SEE ALSO

kinit(1), klist(1)

NAME

kdump — display kernel trace data

SYNOPSIS

kdump [**-dlNnRT**] [**-e** *emulation*] [**-f** *file*] [**-m** *maxdata*] [**-p** *pid*] [**-t** *trstr*]
[**-x** | **-X** *size*] [*file*]

DESCRIPTION

kdump displays the kernel trace files produced with ktrace(1) in human readable format. The file ktrace.out in the current directory is displayed, unless either the **-f** option is used, or a file name is supplied as the last argument.

The options are as follows:

-d Display all numbers in decimal.

-e *emulation*
If an emulation of a process is unknown, interpret system call maps assuming the named emulation instead of default "netbsd".

-f *file* Display the specified file instead of ktrace.out.

-l Loop reading the trace file, once the end-of-file is reached, waiting for more data.

-m *maxdata*
Display at most *maxdata* bytes when decoding I/O.

-N Suppress system call number-to-name translation.

-n Suppress ad hoc translations. Normally **kdump** tries to decode many system calls into a more human readable format. For example, ioctl(2) values are replaced with the macro name and *errno* values are replaced with the strerror(3) string. Suppressing this feature yields a more consistent output format and is easily amenable to further processing.

-p *pid* Only display records from the trace file that are for the indicated pid.

-R Display relative timestamps (time since previous entry).

-T Display absolute timestamps for each entry (seconds since epoch).

-t *trstr* Restrict display to the specified set of kernel trace points. The default is to display everything in the file. See the **-t** option of ktrace(1).

-x Display GIO data in hex and ascii instead of vis(3) format.

-X *size* Same as **-x** but display hex values by groups of *size* bytes. Supported values are 1, 2, 4, 8, and 16.

SEE ALSO

ktrace(1)

HISTORY

The **kdump** command appears in 4.4BSD.

NAME

kf — securely forward tickets

SYNOPSIS

kf [**-p** *port* | **--port**=*port*] [**-l** *login* | **--login**=*login*] [**-c** *ccache* | **--ccache**=*ccache*] [**-F** | **--forwardable**] [**-G** | **--no-forwardable**] [**-h** | **--help**] [**--version**] *host ...*

DESCRIPTION

The **kf** program forwards tickets to a remote host through an authenticated and encrypted stream. Options supported are:

-p *port*, **--port**=*port*
: port to connect to

-l *login*, **--login**=*login*
: remote login name

-c *ccache*, **--ccache**=*ccache*
: remote cred cache

-F, **--forwardable**
: forward forwardable credentials

-G, **--no-forwardable**
: do not forward forwardable credentials

-h, **--help**

--version

kf is useful when you do not want to enter your password on a remote host but want to have your tickets one for example AFS.

In order for **kf** to work you will need to acquire your initial ticket with forwardable flag, i.e. **kinit --forwardable**.

telnet is able to forward tickets by itself.

SEE ALSO

kinit(1), telnet(1), kfd(8)

NAME

kgetcred — get a ticket for a particular service

SYNOPSIS

kgetcred [**--canonicalize**][**-c -cache** | **--cache=***cache*][**-e** *enctype* | **--enctype=***enctype*][**--no-transit-check**][**--version**][**--help**] *service*

DESCRIPTION

kgetcred obtains a ticket for a service. Usually tickets for services are obtained automatically when needed but sometimes for some odd reason you want to obtain a particular ticket or of a special type.

Supported options:

--canonicalize
: requests that the KDC canonicalize the principal.

-c *cache*, **--cache=***cache*
: the credential cache to use.

-e *enctype*, **--enctype=***enctype*
: encryption type to use.

--no-transit-check
: requests that the KDC doesn't do trasnit checking.

--version

--help

SEE ALSO

kinit(1), klist(1)

NAME

kill — terminate or signal a process

SYNOPSIS

kill [**-s** *signal_name*] *pid* ...
kill -l [*exit_status*]
kill -*signal_name* *pid* ...
kill -*signal_number* *pid* ...

DESCRIPTION

The **kill** utility sends a signal to the process(es) specified by the pid operand(s).

Only the super-user may send signals to other users' processes.

The options are as follows:

-s *signal_name*
: A symbolic signal name specifying the signal to be sent instead of the default TERM.

-l [*exit_status*]
: Display the name of the signal corresponding to *exit_status*. *exit_status* may be the exit status of a command killed by a signal (see the special sh(1) parameter '?') or a signal number.

  If no operand is given, display the names of all the signals.

-*signal_name*
: A symbolic signal name specifying the signal to be sent instead of the default TERM.

-*signal_number*
: A non-negative decimal integer, specifying the signal to be sent instead of the default TERM.

The following pids have special meanings:

-1 If superuser, broadcast the signal to all processes; otherwise broadcast to all processes belonging to the user.

0 Broadcast the signal to all processes in the current process group belonging to the user.

Some of the more commonly used signals:

1 HUP (hang up)
2 INT (interrupt)
3 QUIT (quit)
6 ABRT (abort)
9 KILL (non-catchable, non-ignorable kill)
14 ALRM (alarm clock)
15 TERM (software termination signal)

kill is a built-in to csh(1); it allows job specifiers of the form ''%...'' as arguments so process id's are not as often used as **kill** arguments. See csh(1) for details.

SEE ALSO

csh(1), pgrep(1), pkill(1), ps(1), kill(2), sigaction(2), signal(7)

STANDARDS

The **kill** function is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

HISTORY

A **kill** command appeared in Version 6 AT&T UNIX.

NAME

kimpersonate — impersonate a user when there exist a srvtab, keyfile or KeyFile

SYNOPSIS

kimpersonate [**-s** *string* | **--server=***string*] [**-c** *string* | **--client=***string*] [**-k** *string* | **--keytab=***string*] [**-5** | **--krb5**] [**-e** *integer* | **--expire-time=***integer*] [**-a** *string* | **--client-address=***string*] [**-t** *string* | **--enc-type=***string*] [**-f** *string* | **--ticket-flags=***string*] [**--verbose**] [**--version**] [**--help**]

DESCRIPTION

The **kimpersonate** program creates a "fake" ticket using the service-key of the service. The service key can be read from a Kerberos 5 keytab, AFS KeyFile or (if compiled with support for Kerberos 4) a Kerberos 4 srvtab. Supported options:

-s *string*, **--server=***string*
: name of server principal

-c *string*, **--client=***string*
: name of client principal

-k *string*, **--keytab=***string*
: name of keytab file

-5, **--krb5**
: create a Kerberos 5 ticket

-e *integer*, **--expire-time=***integer*
: lifetime of ticket in seconds

-a *string*, **--client-address=***string*
: address of client

-t *string*, **--enc-type=***string*
: encryption type

-f *string*, **--ticket-flags=***string*
: ticket flags for krb5 ticket

--verbose
: Verbose output

--version
: Print version

--help

FILES

Uses `/etc/krb5.keytab`, `/etc/srvtab` and `/usr/afs/etc/KeyFile` when avalible and the the **-k** is used with appropriate prefix.

EXAMPLES

kimpersonate can be used in **samba** root preexec option or for debugging. **kimpersonate** -s host/hummel.e.kth.se@E.KTH.SE -c lha@E.KTH.SE -5 will create a Kerberos 5 ticket for lha@E.KTH.SE for the host hummel.e.kth.se if there exists a keytab entry for it in `/etc/krb5.keytab`.

SEE ALSO

`kinit`(1), `klist`(1)

AUTHORS

Love Hornquist Astrand <lha@kth.se>

NAME

kinit **kauth** — acquire initial tickets

SYNOPSIS

kinit [**-4** | **--524init**][**-9** | **--524convert**][**--afslog**][**-c** *cachename* | **--cache=***cachename*][**-f** | **--forwardable**][**-t** *keytabname* | **--keytab=***keytabname*][**-l** *time* | **--lifetime=***time*][**-p** | **--proxiable**] [**-R** | **--renew**][**--renewable**][**-r** *time* | **--renewable-life=***time*][**-S** *principal* | **--server=***principal*][**-s** *time* | **--start-time=***time*] [**-k** | **--use-keytab**][**-v** | **--validate**][**-e** *enctypes* | **--enctypes=***enctypes*][**-a** *addresses* | **--extra-addresses=***addresses*] [**--password-file=***filename*][**--fcache-version=***version-number*] [**-A** | **--no-addresses**][**--anonymous**][**--version**][**--help**] [*principal* [*command*]]

DESCRIPTION

kinit is used to authenticate to the Kerberos server as *principal*, or if none is given, a system generated default (typically your login name at the default realm), and acquire a ticket granting ticket that can later be used to obtain tickets for other services.

If you have compiled **kinit** with Kerberos 4 support and you have a Kerberos 4 server, **kinit** will detect this and get you Kerberos 4 tickets.

Supported options:

-c *cachename* **--cache=***cachename*
: The credentials cache to put the acquired ticket in, if other than default.

-f, **--forwardable**
: Get ticket that can be forwarded to another host.

-t *keytabname*, **--keytab=***keytabname*
: Don't ask for a password, but instead get the key from the specified keytab.

-l *time*, **--lifetime=***time*
: Specifies the lifetime of the ticket. The argument can either be in seconds, or a more human readable string like '1h'.

-p, **--proxiable**
: Request tickets with the proxiable flag set.

-R, **--renew**
: Try to renew ticket. The ticket must have the 'renewable' flag set, and must not be expired.

--renewable
: The same as **--renewable-life**, with an infinite time.

-r *time*, **--renewable-life=***time*
: The max renewable ticket life.

-S *principal*, **--server=***principal*
: Get a ticket for a service other than krbtgt/LOCAL.REALM.

-s *time*, **--start-time=***time*
: Obtain a ticket that starts to be valid *time* (which can really be a generic time specification, like '1h') seconds into the future.

-k, **--use-keytab**
The same as **--keytab**, but with the default keytab name (normally *FILE:/etc/krb5.keytab*).

-v, **--validate**
Try to validate an invalid ticket.

-e, **--enctypes=***enctypes*
Request tickets with this particular enctype.

--password-file=*filename*
read the password from the first line of *filename*. If the *filename* is *STDIN*, the password will be read from the standard input.

--fcache-version=*version-number*
Create a credentials cache of version *version-number*.

-a, **--extra-addresses=***enctypes*
Adds a set of addresses that will, in addition to the systems local addresses, be put in the ticket. This can be useful if all addresses a client can use can't be automatically figured out. One such example is if the client is behind a firewall. Also settable via `libdefaults/extra_addresses` in `krb5.conf`(5).

-A, **--no-addresses**
Request a ticket with no addresses.

--anonymous
Request an anonymous ticket (which means that the ticket will be issued to an anonymous principal, typically "anonymous@REALM").

The following options are only available if **kinit** has been compiled with support for Kerberos 4.

-4, **--524init**
Try to convert the obtained Kerberos 5 krbtgt to a version 4 compatible ticket. It will store this ticket in the default Kerberos 4 ticket file.

-9, **--524convert**
only convert ticket to version 4

--afslog
Gets AFS tickets, converts them to version 4 format, and stores them in the kernel. Only useful if you have AFS.

The *forwardable*, *proxiable*, *ticket_life*, and *renewable_life* options can be set to a default value from the `appdefaults` section in krb5.conf, see `krb5_appdefault`(3).

If a *command* is given, **kinit** will set up new credentials caches, and AFS PAG, and then run the given command. When it finishes the credentials will be removed.

ENVIRONMENT

`KRB5CCNAME`
Specifies the default credentials cache.

`KRB5_CONFIG`
The file name of `krb5.conf`, the default being `/etc/krb5.conf`.

`KRBTKFILE`
Specifies the Kerberos 4 ticket file to store version 4 tickets in.

SEE ALSO

`kdestroy`(1), `klist`(1), `krb5_appdefault`(3), `krb5.conf`(5)

NAME

klist — list Kerberos credentials

SYNOPSIS

klist [**-c** *cache* | **--cache=***cache*] [**-s** | **-t** | **--test**] [**-T** | **--tokens**] [**-5** | **--v5**] [**-v** | **--verbose**] [**-l** | **--list-caches**] [**-f**] [**--version**] [**--help**]

DESCRIPTION

klist reads and displays the current tickets in the credential cache (also known as the ticket file).

Options supported:

-c *cache*, **--cache=***cache*
: credential cache to list

-s, **-t**, **--test**
: Test for there being an active and valid TGT for the local realm of the user in the credential cache.

-T, **--tokens**
: display AFS tokens

-5, **--v5**
: display v5 cred cache (this is the default)

-f
: Include ticket flags in short form, each character stands for a specific flag, as follows:

F	forwardable
f	forwarded
P	proxiable
p	proxied
D	postdate-able
d	postdated
R	renewable
I	initial
i	invalid
A	pre-authenticated
H	hardware authenticated

This information is also output with the **--verbose** option, but in a more verbose way.

-v, **--verbose**
: Verbose output. Include all possible information:

Server
: the principal the ticket is for

Ticket etype
: the encryption type used in the ticket, followed by the key version of the ticket, if it is available

Session key
: the encryption type of the session key, if it’s different from the encryption type of the ticket

Auth time
: the time the authentication exchange took place

Start time
: the time that this ticket is valid from (only printed if it's different from the auth time)

End time
: when the ticket expires, if it has already expired this is also noted

Renew till
: the maximum possible end time of any ticket derived from this one

Ticket flags
: the flags set on the ticket

Addresses
: the set of addresses from which this ticket is valid

-l, **--list-caches**
: List the credential caches for the current users, not all cache types supports listing multiple caches.

SEE ALSO

`kdestroy`(1), `kinit`(1)

NAME

kpasswd — Kerberos 5 password changing program

SYNOPSIS

kpasswd [**--admin-principal=***principal*][**-c** *cache* | **--cache=***cache*]
[*principal ...*]

DESCRIPTION

kpasswd is the client for changing passwords.

If administrator principal is given that principal is used to change the password.

Multiple passwords for different users can be changed at the same time, then the administrator principal will be used. If the administrator isn't specified on the command prompt, the principal of the default credential cache will be used.

If a credential cache is given, the **--admin-principal** flag is ignored and use the default name of the credential cache is used instead.

DIAGNOSTICS

If the password quality check fails or some other error occurs, an explanation is printed.

SEE ALSO

kpasswdd(8)

NAME

kpasswd — modify a user's Kerberos 5 password

SYNOPSIS

kpasswd [*principal*]

DESCRIPTION

kpasswd changes the user's Kerberos 5 password.

The **kpasswd** command is deprecated. See passwd(1) for more information.

NAME

krb5-config — give information on how to link code against Heimdal libraries

SYNOPSIS

krb5-config [**--prefix**[=*dir*]] [**--exec-prefix**[=*dir*]] [**--libs**] [**--cflags**] [*libraries*]

DESCRIPTION

krb5-config tells the application programmer what special flags to use to compile and link programs against the libraries installed by Heimdal.

Options supported:

--prefix[=*dir*]
: Print the prefix if no *dir* is specified, otherwise set prefix to *dir*.

--exec-prefix[=*dir*]
: Print the exec-prefix if no *dir* is specified, otherwise set exec-prefix to *dir*.

--libs
: Output the set of libraries that should be linked against.

--cflags
: Output the set of flags to give to the C compiler when using the Heimdal libraries.

By default **krb5-config** will output the set of flags and libraries to be used by a normal program using the krb5 API. The user can also supply a library to be used, the supported ones are:

krb5 (the default)

gssapi use the krb5 gssapi mechanism

kadm-client
: use the client-side kadmin libraries

kadm-server
: use the server-side kadmin libraries

SEE ALSO

cc(1)

HISTORY

krb5-config appeared in Heimdal 0.3d.

NAME

ktrace, **ktruss** — enable kernel process tracing

SYNOPSIS

ktrace [**-aCcdins**] [**-f** *trfile*] [**-g** *pgrp*] [**-p** *pid*] [**-t** *trstr*]
ktrace [**-adis**] [**-f** *trfile*] [**-t** *trstr*] *command*
ktruss [**-aCcdilnRT**] [**-e** *emulation*] [**-f** *infile*] [**-g** *pgrp*] [**-m** *maxdata*]
[**-o** *outfile*] [**-p** *pid*] [**-t** *trstr*]
ktruss [**-adinRT**] [**-e** *emulation*] [**-m** *maxdata*] [**-o** *outfile*] [**-t** *trstr*]
[**-v** *vers*] command

DESCRIPTION

ktrace enables kernel trace logging for the specified processes. Kernel trace data is logged to the file `ktrace.out`. The kernel operations that are traced include system calls, namei translations, signal processing, and I/O.

Once tracing is enabled on a process, trace data will be logged until either the process exits or the trace point is cleared. A traced process can generate enormous amounts of log data quickly; It is strongly suggested that users memorize how to disable tracing before attempting to trace a process. The following command is sufficient to disable tracing on all user owned processes, and, if executed by root, all processes:

```
$ ktrace -C
```

The trace file is not human readable; use `kdump`(1) to decode it.

ktruss is functionally the same as **ktrace** except that trace output is printed on standard output or to the file specified with the **-o** option. **ktruss** is useful to see the kernel operations interleaved with the program output.

The options are as follows:

-a Append to the trace file instead of truncating it.

-C Disable tracing on all user owned processes, and, if executed by root, all processes in the system.

-c Clear the trace points associated with the specified file or processes.

-d Descendants; perform the operation for all current children of the designated processes.

-f *trfile*
Log trace records to *trfile* instead of `ktrace.out`.

-f *infile*
Read the trace records from *infile* and print them in a human readable format to standard out.

-g *pgid*
Enable (disable) tracing on all processes in the process group (only one **-g** flag is permitted).

-i Inherit; pass the trace flags to all future children of the designated processes.

-l Poll the trace file for new data and print it to standard out. Only for use together with the **-f** option.

-m *maxdata*
Print at most *maxdata* bytes of data. This is used for pointer type arguments, e.g., strings. The data will be escaped in C-style unless **-x** is specified when it will be output in hex and ascii.

-n Stop tracing if attempts to write to the trace file would block. This option always affects **ktruss** and only affects **ktrace** when writing to `stdout`. If this flag is not set, then the traced program will block until it can write more data to the trace file descriptor.

-o *outfile*
Log trace records to *outfile*. Without this option **ktruss** will print its output in a human readable format to standard out.

-p *pid*
Enable (disable) tracing on the indicated process id (only one **-p** flag is permitted).

-s Write to the trace file with synchronized I/O.

-R Display relative time stamps to output.

-T Same as the **-R** option, but use absolute timestamps instead.

-t *trstr*
The string argument represents the kernel trace points, one per letter. The following table equates the letters with the tracepoints:

A trace all tracepoints
a trace exec arguments
c trace system calls
e trace emulation changes
i trace I/O
l trace Mach out of line data when running Mach binaries with COMPAT_MACH (currently limited to i386 and powerpc ports).
m trace Mach messages when running Mach binaries with COMPAT_MACH (currently limited to i386 and powerpc ports).
n trace namei translations
S trace MIB access (sysctl)
s trace signal processing
u trace user data
v trace exec environment
w trace context switches
+ trace the default set of trace points (c, e, i, l, m, n, s, u)
- do not trace following trace points

-e *emulation*
If an emulation of a process is unknown, interpret system call maps assuming the named emulation instead of default "netbsd".

command
Execute *command* with the specified trace flags.

-v *version*
Determines the *version* of the file generated. Version 0 is the compatible ktrace format, and version 1 is the new format with lwp IDs and nanosecond (instead of microsecond) timestamps.

The **-p**, **-g**, and *command* options are mutually exclusive. The **-R** and **-T** options are also mutually exclusive.

EXAMPLES

trace all kernel operations of process id 34

```
$ ktrace -p 34
```

```
# trace all kernel operations of processes in process group 15 and
# pass the trace flags to all current and future children
$ ktrace -idg 15
```

disable all tracing of process 65
```
$ ktrace -cp 65
```

disable tracing signals on process 70 and all current children
```
$ ktrace -t s -cdp 70
```

enable tracing of I/O on process 67
```
$ ktrace -ti -p 67
```

run the command "w", tracing only system calls
```
$ ktrace -tc w
```

disable all tracing to the file "tracedata"
```
$ ktrace -c -f tracedata
```

disable tracing of all processes owned by the user
```
$ ktrace -C
```

run the command "w", displaying to standard output
```
$ ktruss w
```

trace process 42 and log the records to "ktruss.out"
```
$ ktruss -p 42 -o ktruss.out
```

poll ktruss.out for available records and print them
```
$ ktruss -lf ktruss.out
```

SEE ALSO

`kdump`(1), `ktrace`(2)

HISTORY

The **`ktrace`** command appears in 4.4BSD.

NAME

kx — securely forward X conections

SYNOPSIS

kx [**-l** *username*] [**-k**] [**-d**] [**-t**] [**-p** *port*] [**-P**] *host*

DESCRIPTION

The **kx** program forwards an X connection from a remote client to a local screen through an authenticated and encrypted stream. Options supported by **kx**:

-l Log in on the remote the host as user *username*.

-k Do not enable keep-alives on the TCP connections.

-d Do not fork. This is mainly useful for debugging.

-t Listen not only on a UNIX-domain socket but on a TCP socket as well.

-p Use the port *port*.

-P Force passive mode.

This program is used by **rxtelnet** and **rxterm** and you should not need to run it directly.

It connects to a **kxd** on the host *host* and then will relay the traffic from the remote X clients to the local server. When started, it prints the display and Xauthority-file to be used on host *host* and then goes to the background, waiting for connections from the remote **kxd**.

SEE ALSO

rxtelnet(1), rxterm(1), kxd(8)

NAME

lam — laminate files

SYNOPSIS

lam [**-f** *min.max*] [**-p** *min.max*] [**-s** *sepstring*] [**-t** *c*] *file ...*

DESCRIPTION

lam copies the named files side by side onto the standard output. The *n*-th input lines from the input *files* are considered fragments of the single long *n*-th output line into which they are assembled. The name "-" means the standard input, and may be repeated.

Normally, each option affects only the *file* after it. If the option letter is capitalized it affects all subsequent files until it appears again uncapitalized. The options are described below.

-f *min.max* Print line fragments according to the format string *min.max*, where *min* is the minimum field width and *max* the maximum field width. If *min* begins with a zero, zeros will be added to make up the field width, and if it begins with a '-', the fragment will be left-adjusted within the field.

-p *min.max* Like **-f**, but pad this file's field when end-of-file is reached and other files are still active.

-s *sepstring*
Print *sepstring* before printing line fragments from the next file. This option may appear after the last file.

-t *c* The input line terminator is *c* instead of a newline. The newline normally appended to each output line is omitted.

To print files simultaneously for easy viewing use pr(1).

EXAMPLES

The command

```
lam file1 file2 file3 file4
```

joins 4 files together along each line. To merge the lines from four different files use

```
lam file1 -S "\
" file2 file3 file4
```

Every 2 lines of a file may be joined on one line with

```
lam - - < file
```

and a form letter with substitutions keyed by '@' can be done with

```
lam -t @ letter changes
```

SEE ALSO

join(1), pr(1), printf(3)

NAME

last — indicate last logins of users and ttys

SYNOPSIS

last [**-***n*] [**-nTx**] [**-f** *file*] [**-H** *hostsize*] [**-h** *host*] [**-L** *linesize*] [**-N** *namesize*] [**-t** *tty*] [user ...]

DESCRIPTION

last will list the sessions of specified *users*, *ttys*, and *hosts*, in reverse time order. Each line of output contains the user name, the tty from which the session was conducted, any hostname, the start and stop times for the session, and the duration of the session. If the session is still continuing or was cut short by a crash or shutdown, **last** will so indicate.

-*n* Limits the report to *n* lines.

-f *file* **last** reads the file *file* instead of the default, `/var/log/wtmpx` or `/var/log/wtmp`. If the file ends with 'x', it is treated as a `utmpx`(5) format file, else it is treated as a `utmp`(5) format file. If the file is "-", standard input is used.

-H *hostsize* Use the provided hostsize as the width to format the host name field.

-h *host* *Host* names may be names or internet numbers.

-L *linesize* Use the provided linesize as the width to format the tty field.

-N *namesize* Use the provided namesize as the width to format the login name field.

-n Print host addresses numerically. This option works only on `wtmpx`(5) entries, and prints nothing on `wtmp`(5) entries.

-T Display better time information, including the year and seconds.

-t *tty* Specify the *tty*. Tty names may be given fully or abbreviated, for example, "`last -t 03`" is equivalent to "`last -t tty03`".

-x Assume that the file given is `wtmpx`(5) format, even if the filename does not end with an 'x'.

If multiple arguments are given, the information which applies to any of the arguments is printed, e.g., "`last root -t console`" would list all of "`root`'s" sessions as well as all sessions on the console terminal. If no users, hostnames, or terminals are specified, **last** prints a record of all logins and logouts.

The pseudo-user *reboot* logs in at reboots of the system, thus "`last reboot`" will give an indication of mean time between reboot.

If **last** is interrupted, it indicates to what date the search has progressed. If interrupted with a quit signal **last** indicates how far the search has progressed and then continues.

FILES

`/var/log/wtmp` login data base
`/var/log/wtmpx` login data base

SEE ALSO

`lastcomm`(1), `utmp`(5), `utmpx`(5), `ac`(8), `lastlogin`(8)

HISTORY

last appeared in 3.0 BSD.

NAME

lastcomm — show last commands executed in reverse order

SYNOPSIS

lastcomm [**-f** *file*] [*command ...*] [*user ...*] [*terminal ...*]

DESCRIPTION

lastcomm gives information on previously executed commands. With no arguments, **lastcomm** prints information about all the commands recorded during the current accounting file's lifetime.

Option:

-f *file* Read from *file* rather than the default accounting file.

If called with arguments, only accounting entries with a matching *command* name, *user* name, or *terminal* name are printed. So, for example:

```
lastcomm a.out root ttyd0
```

would produce a listing of all the executions of commands named `a.out` by user *root* on the terminal *ttyd0*.

For each process entry, the following are printed.

- The name of the user who ran the process.
- Flags, as accumulated by the accounting facilities in the system.
- The command name under which the process was called.
- The amount of cpu time used by the process (in seconds).
- The time the process started.
- The elapsed time of the process.

The flags are encoded as follows: "S" indicates the command was executed by the super-user, "F" indicates the command ran after a fork, but without a following `exec`(3), "C" indicates the command was run in PDP-11 compatibility mode (VAX only), "D" indicates the command terminated with the generation of a `core` file, and "X" indicates the command was terminated with a signal.

The "S" and "C" flags are no longer recorded by the system, but will be reported by **lastcomm** when reading from an accounting file generated by an older version of the system.

FILES

`/var/account/acct` Default accounting file.

SEE ALSO

`last`(1), `sigaction`(2), `acct`(5), `core`(5)

HISTORY

The **lastcomm** command appeared in 3.0 BSD.

NAME

ld − Using LD, the GNU linker

SYNOPSIS

ld [**options**] *objfile* ...

DESCRIPTION

ld combines a number of object and archive files, relocates their data and ties up symbol references. Usually the last step in compiling a program is to run **ld**.

ld accepts Linker Command Language files written in a superset of AT&T's Link Editor Command Language syntax, to provide explicit and total control over the linking process.

This man page does not describe the command language; see the **ld** entry in `info`, or the manual ld: the GNU linker, for full details on the command language and on other aspects of the GNU linker.

This version of **ld** uses the general purpose BFD libraries to operate on object files. This allows **ld** to read, combine, and write object files in many different formats−−−for example, COFF or `a.out`. Different formats may be linked together to produce any available kind of object file.

Aside from its flexibility, the GNU linker is more helpful than other linkers in providing diagnostic information. Many linkers abandon execution immediately upon encountering an error; whenever possible, **ld** continues executing, allowing you to identify other errors (or, in some cases, to get an output file in spite of the error).

The GNU linker **ld** is meant to cover a broad range of situations, and to be as compatible as possible with other linkers. As a result, you have many choices to control its behavior.

OPTIONS

The linker supports a plethora of command-line options, but in actual practice few of them are used in any particular context. For instance, a frequent use of **ld** is to link standard Unix object files on a standard, supported Unix system. On such a system, to link a file `hello.o`:

```
ld −o <output> /lib/crt0.o hello.o −lc
```

This tells **ld** to produce a file called *output* as the result of linking the file `/lib/crt0.o` with `hello.o` and the library `libc.a`, which will come from the standard search directories. (See the discussion of the **−l** option below.)

Some of the command-line options to **ld** may be specified at any point in the command line. However, options which refer to files, such as **−l** or **−T**, cause the file to be read at the point at which the option appears in the command line, relative to the object files and other file options. Repeating non-file options with a different argument will either have no further effect, or override prior occurrences (those further to the left on the command line) of that option. Options which may be meaningfully specified more than once are noted in the descriptions below.

Non-option arguments are object files or archives which are to be linked together. They may follow, precede, or be mixed in with command-line options, except that an object file argument may not be placed between an option and its argument.

Usually the linker is invoked with at least one object file, but you can specify other forms of binary input files using **−l**, **−R**, and the script command language. If *no* binary input files at all are specified, the linker does not produce any output, and issues the message **No input files**.

If the linker cannot recognize the format of an object file, it will assume that it is a linker script. A script specified in this way augments the main linker script used for the link (either the default linker script or the one specified by using **−T**). This feature permits the linker to link against a file which appears to be an object or an archive, but actually merely defines some symbol values, or uses `INPUT` or `GROUP` to load other objects. Note that specifying a script in this way merely augments the main linker script; use the **−T** option to replace the default linker script entirely.

For options whose names are a single letter, option arguments must either follow the option letter without intervening whitespace, or be given as separate arguments immediately following the option that requires

them.

For options whose names are multiple letters, either one dash or two can precede the option name; for example, **−trace−symbol** and **−−trace−symbol** are equivalent. Note−−−there is one exception to this rule. Multiple letter options that start with a lower case 'o' can only be preceeded by two dashes. This is to reduce confusion with the **−o** option. So for example **−omagic** sets the output file name to **magic** whereas **−−omagic** sets the NMAGIC flag on the output.

Arguments to multiple-letter options must either be separated from the option name by an equals sign, or be given as separate arguments immediately following the option that requires them. For example, **−−trace−symbol foo** and **−−trace−symbol=foo** are equivalent. Unique abbreviations of the names of multiple-letter options are accepted.

Note−−−if the linker is being invoked indirectly, via a compiler driver (e.g. **gcc**) then all the linker command line options should be prefixed by **−Wl,** (or whatever is appropriate for the particular compiler driver) like this:

```
gcc −Wl,−−startgroup foo.o bar.o −Wl,−−endgroup
```

This is important, because otherwise the compiler driver program may silently drop the linker options, resulting in a bad link.

Here is a table of the generic command line switches accepted by the GNU linker:

−a*keyword*

This option is supported for HP/UX compatibility. The *keyword* argument must be one of the strings **archive**, **shared**, or **default**. **−aarchive** is functionally equivalent to **−Bstatic**, and the other two keywords are functionally equivalent to **−Bdynamic**. This option may be used any number of times.

−A*architecture*
−−architecture=*architecture*

In the current release of **ld**, this option is useful only for the Intel 960 family of architectures. In that **ld** configuration, the *architecture* argument identifies the particular architecture in the 960 family, enabling some safeguards and modifying the archive-library search path.

Future releases of **ld** may support similar functionality for other architecture families.

−b *input-format*
−−format=*input-format*

ld may be configured to support more than one kind of object file. If your **ld** is configured this way, you can use the **−b** option to specify the binary format for input object files that follow this option on the command line. Even when **ld** is configured to support alternative object formats, you don't usually need to specify this, as **ld** should be configured to expect as a default input format the most usual format on each machine. *input-format* is a text string, the name of a particular format supported by the BFD libraries. (You can list the available binary formats with **objdump −i**.)

You may want to use this option if you are linking files with an unusual binary format. You can also use **−b** to switch formats explicitly (when linking object files of different formats), by including **−b** *input-format* before each group of object files in a particular format.

The default format is taken from the environment variable `GNUTARGET`.

You can also define the input format from a script, using the command `TARGET`;

−c *MRI-commandfile*
−−mri−script=*MRI-commandfile*

For compatibility with linkers produced by MRI, **ld** accepts script files written in an alternate, restricted command language, described in the MRI Compatible Script Files section of GNU ld documentation. Introduce MRI script files with the option **−c**; use the **−T** option to run linker scripts written in the general-purpose **ld** scripting language. If *MRI-cmdfile* does not exist, **ld** looks for it in the directories specified by any **−L** options.

−d
−dc
−dp
These three options are equivalent; multiple forms are supported for compatibility with other linkers. They assign space to common symbols even if a relocatable output file is specified (with **−r**). The script command `FORCE_COMMON_ALLOCATION` has the same effect.

−e *entry*
−−entry=*entry*
Use *entry* as the explicit symbol for beginning execution of your program, rather than the default entry point. If there is no symbol named *entry*, the linker will try to parse *entry* as a number, and use that as the entry address (the number will be interpreted in base 10; you may use a leading **0x** for base 16, or a leading **0** for base 8).

−−exclude−libs *lib,lib,...*
Specifies a list of archive libraries from which symbols should not be automatically exported. The library names may be delimited by commas or colons. Specifying `−−exclude−libs ALL` excludes symbols in all archive libraries from automatic export. This option is available only for the i386 PE targeted port of the linker and for ELF targeted ports. For i386 PE, symbols explicitly listed in a .def file are still exported, regardless of this option. For ELF targeted ports, symbols affected by this option will be treated as hidden.

−E
−−export−dynamic
When creating a dynamically linked executable, add all symbols to the dynamic symbol table. The dynamic symbol table is the set of symbols which are visible from dynamic objects at run time.

If you do not use this option, the dynamic symbol table will normally contain only those symbols which are referenced by some dynamic object mentioned in the link.

If you use `dlopen` to load a dynamic object which needs to refer back to the symbols defined by the program, rather than some other dynamic object, then you will probably need to use this option when linking the program itself.

You can also use the version script to control what symbols should be added to the dynamic symbol table if the output format supports it. See the description of **−−version−script** in `@ref{`VERSION}.

−EB
Link big-endian objects. This affects the default output format.

−EL
Link little-endian objects. This affects the default output format.

−f
−−auxiliary *name*
When creating an ELF shared object, set the internal DT_AUXILIARY field to the specified name. This tells the dynamic linker that the symbol table of the shared object should be used as an auxiliary filter on the symbol table of the shared object *name*.

If you later link a program against this filter object, then, when you run the program, the dynamic linker will see the DT_AUXILIARY field. If the dynamic linker resolves any symbols from the filter object, it will first check whether there is a definition in the shared object *name*. If there is one, it will be used instead of the definition in the filter object. The shared object *name* need not exist. Thus the shared object *name* may be used to provide an alternative implementation of certain functions, perhaps for debugging or for machine specific performance.

This option may be specified more than once. The DT_AUXILIARY entries will be created in the order in which they appear on the command line.

−F *name*
−−filter *name*
When creating an ELF shared object, set the internal DT_FILTER field to the specified name. This tells the dynamic linker that the symbol table of the shared object which is being created should be used as a filter on the symbol table of the shared object *name*.

If you later link a program against this filter object, then, when you run the program, the dynamic linker will see the DT_FILTER field. The dynamic linker will resolve symbols according to the symbol table of the filter object as usual, but it will actually link to the definitions found in the shared object *name*. Thus the filter object can be used to select a subset of the symbols provided by the object *name*.

Some older linkers used the **−F** option throughout a compilation toolchain for specifying object-file format for both input and output object files. The GNU linker uses other mechanisms for this purpose: the **−b**, **−−format**, **−−oformat** options, the `TARGET` command in linker scripts, and the `GNUTARGET` environment variable. The GNU linker will ignore the **−F** option when not creating an ELF shared object.

−fini *name*
When creating an ELF executable or shared object, call NAME when the executable or shared object is unloaded, by setting DT_FINI to the address of the function. By default, the linker uses `_fini` as the function to call.

−g Ignored. Provided for compatibility with other tools.

−G*value*
−−gpsize=*value*
Set the maximum size of objects to be optimized using the GP register to *size*. This is only meaningful for object file formats such as MIPS ECOFF which supports putting large and small objects into different sections. This is ignored for other object file formats.

−h*name*
−soname=*name*
When creating an ELF shared object, set the internal DT_SONAME field to the specified name. When an executable is linked with a shared object which has a DT_SONAME field, then when the executable is run the dynamic linker will attempt to load the shared object specified by the DT_SONAME field rather than the using the file name given to the linker.

−i Perform an incremental link (same as option **−r**).

−init *name*
When creating an ELF executable or shared object, call NAME when the executable or shared object is loaded, by setting DT_INIT to the address of the function. By default, the linker uses `_init` as the function to call.

−l*archive*
−−library=*archive*
Add archive file *archive* to the list of files to link. This option may be used any number of times. **ld** will search its path-list for occurrences of `lib`*`archive`*`.a` for every *archive* specified.

On systems which support shared libraries, **ld** may also search for libraries with extensions other than `.a`. Specifically, on ELF and SunOS systems, **ld** will search a directory for a library with an extension of `.so` before searching for one with an extension of `.a`. By convention, a `.so` extension indicates a shared library.

The linker will search an archive only once, at the location where it is specified on the command line. If the archive defines a symbol which was undefined in some object which appeared before the archive on the command line, the linker will include the appropriate file(s) from the archive. However, an undefined symbol in an object appearing later on the command line will not cause the linker to search the archive again.

See the **−(** option for a way to force the linker to search archives multiple times.

You may list the same archive multiple times on the command line.

This type of archive searching is standard for Unix linkers. However, if you are using **ld** on AIX, note that it is different from the behaviour of the AIX linker.

−L*searchdir*
−−library−path=*searchdir*
Add path *searchdir* to the list of paths that **ld** will search for archive libraries and **ld** control scripts. You may use this option any number of times. The directories are searched in the order in which they are specified on the command line. Directories specified on the command line are searched before the default directories. All **−L** options apply to all **−l** options, regardless of the order in which the options appear.

If *searchdir* begins with `=`, then the `=` will be replaced by the *sysroot prefix*, a path specified when the linker is configured.

The default set of paths searched (without being specified with **−L**) depends on which emulation mode **ld** is using, and in some cases also on how it was configured.

The paths can also be specified in a link script with the `SEARCH_DIR` command. Directories specified this way are searched at the point in which the linker script appears in the command line.

−m*emulation*
Emulate the *emulation* linker. You can list the available emulations with the **−−verbose** or **−V** options.

If the **−m** option is not used, the emulation is taken from the `LDEMULATION` environment variable, if that is defined.

Otherwise, the default emulation depends upon how the linker was configured.

−M
−−print−map
Print a link map to the standard output. A link map provides information about the link, including the following:

* Where object files and symbols are mapped into memory.
* How common symbols are allocated.
* All archive members included in the link, with a mention of the symbol which caused the archive member to be brought in.

−n
−−nmagic
Turn off page alignment of sections, and mark the output as `NMAGIC` if possible.

−N
−−omagic
Set the text and data sections to be readable and writable. Also, do not page-align the data segment, and disable linking against shared libraries. If the output format supports Unix style magic numbers, mark the output as `OMAGIC`. Note: Although a writable text section is allowed for PE-COFF targets, it does not conform to the format specification published by Microsoft.

−−no−omagic
This option negates most of the effects of the **−N** option. It sets the text section to be read−only, and forces the data segment to be page−aligned. Note − this option does not enable linking against shared libraries. Use **−Bdynamic** for this.

−o *output*
−−output=*output*
Use *output* as the name for the program produced by **ld**; if this option is not specified, the name *a.out* is used by default. The script command `OUTPUT` can also specify the output file name.

−O *level*
If *level* is a numeric values greater than zero **ld** optimizes the output. This might take significantly longer and therefore probably should only be enabled for the final binary.

−q
−−emit−relocs
Leave relocation sections and contents in fully linked exececutables. Post link analysis and optimization tools may need this information in order to perform correct modifications of executables. This results in larger executables.

This option is currently only supported on ELF platforms.

−r
−−relocatable
Generate relocatable output−−−i.e., generate an output file that can in turn serve as input to **ld**. This is often called *partial linking*. As a side effect, in environments that support standard Unix magic numbers, this option also sets the output file's magic number to `OMAGIC`. If this option is not specified, an absolute file is produced. When linking C++ programs, this option *will not* resolve references to constructors; to do that, use **−Ur**.

When an input file does not have the same format as the output file, partial linking is only supported if that input file does not contain any relocations. Different output formats can have further restrictions; for example some `a.out`−based formats do not support partial linking with input files in other formats at all.

This option does the same thing as **−i**.

−R *filename*
−−just−symbols=*filename*
Read symbol names and their addresses from *filename*, but do not relocate it or include it in the output. This allows your output file to refer symbolically to absolute locations of memory defined in other programs. You may use this option more than once.

For compatibility with other ELF linkers, if the **−R** option is followed by a directory name, rather than a file name, it is treated as the **−rpath** option.

−s
−−strip−all
Omit all symbol information from the output file.

−S
−−strip−debug
Omit debugger symbol information (but not all symbols) from the output file.

−t
−−trace
Print the names of the input files as **ld** processes them.

−T *scriptfile*
−−script=*scriptfile*
Use *scriptfile* as the linker script. This script replaces **ld**'s default linker script (rather than adding to it), so *commandfile* must specify everything necessary to describe the output file. If *scriptfile* does not exist in the current directory, `ld` looks for it in the directories specified by any preceding **−L** options. Multiple **−T** options accumulate.

−u *symbol*
−−undefined=*symbol*
Force *symbol* to be entered in the output file as an undefined symbol. Doing this may, for example, trigger linking of additional modules from standard libraries. **−u** may be repeated with different option arguments to enter additional undefined symbols. This option is equivalent to the `EXTERN` linker script command.

−Ur

For anything other than C++ programs, this option is equivalent to **−r**: it generates relocatable output−−−i.e., an output file that can in turn serve as input to **ld**. When linking C++ programs, **−Ur** *does* resolve references to constructors, unlike **−r**. It does not work to use **−Ur** on files that were themselves linked with **−Ur**; once the constructor table has been built, it cannot be added to. Use **−Ur** only for the last partial link, and **−r** for the others.

−−unique[=*SECTION***]**

Creates a separate output section for every input section matching *SECTION*, or if the optional wildcard *SECTION* argument is missing, for every orphan input section. An orphan section is one not specifically mentioned in a linker script. You may use this option multiple times on the command line; It prevents the normal merging of input sections with the same name, overriding output section assignments in a linker script.

−v
−−version
−V Display the version number for **ld**. The **−V** option also lists the supported emulations.

−x
−−discard−all

Delete all local symbols.

−X
−−discard−locals

Delete all temporary local symbols. For most targets, this is all local symbols whose names begin with **L**.

−y *symbol*
−−trace−symbol=*symbol*

Print the name of each linked file in which *symbol* appears. This option may be given any number of times. On many systems it is necessary to prepend an underscore.

This option is useful when you have an undefined symbol in your link but don't know where the reference is coming from.

−Y *path*

Add *path* to the default library search path. This option exists for Solaris compatibility.

−z *keyword*

The recognized keywords are:

combreloc

Combines multiple reloc sections and sorts them to make dynamic symbol lookup caching possible.

defs

Disallows undefined symbols in object files. Undefined symbols in shared libraries are still allowed.

initfirst

This option is only meaningful when building a shared object. It marks the object so that its runtime initialization will occur before the runtime initialization of any other objects brought into the process at the same time. Similarly the runtime finalization of the object will occur after the runtime finalization of any other objects.

interpose

Marks the object that its symbol table interposes before all symbols but the primary executable.

loadfltr

Marks the object that its filters be processed immediately at runtime.

muldefs
Allows multiple definitions.

nocombreloc
Disables multiple reloc sections combining.

nocopyreloc
Disables production of copy relocs.

nodefaultlib
Marks the object that the search for dependencies of this object will ignore any default library search paths.

nodelete
Marks the object shouldn't be unloaded at runtime.

nodlopen
Marks the object not available to `dlopen`.

nodump
Marks the object can not be dumped by `dldump`.

now
When generating an executable or shared library, mark it to tell the dynamic linker to resolve all symbols when the program is started, or when the shared library is linked to using dlopen, instead of deferring function call resolution to the point when the function is first called.

origin
Marks the object may contain `$ORIGIN`.

Other keywords are ignored for Solaris compatibility.

−(*archives* **−)**
−−start−group *archives* **−−end−group**
The *archives* should be a list of archive files. They may be either explicit file names, or **−l** options.

The specified archives are searched repeatedly until no new undefined references are created. Normally, an archive is searched only once in the order that it is specified on the command line. If a symbol in that archive is needed to resolve an undefined symbol referred to by an object in an archive that appears later on the command line, the linker would not be able to resolve that reference. By grouping the archives, they all be searched repeatedly until all possible references are resolved.

Using this option has a significant performance cost. It is best to use it only when there are unavoidable circular references between two or more archives.

−−accept−unknown−input−arch
−−no−accept−unknown−input−arch
Tells the linker to accept input files whose architecture cannot be recognised. The assumption is that the user knows what they are doing and deliberately wants to link in these unknown input files. This was the default behaviour of the linker, before release 2.14. The default behaviour from release 2.14 onwards is to reject such input files, and so the **−−accept−unknown−input−arch** option has been added to restore the old behaviour.

−−as−needed
−−no−as−needed
This option affects ELF DT_NEEDED tags for dynamic libraries mentioned on the command line after the **−−as−needed** option. Normally, the linker will add a DT_NEEDED tag for each dynamic library mentioned on the command line, regardless of whether the library is actually needed. **−−as−needed** causes DT_NEEDED tags to only be emitted for libraries that satisfy some symbol reference from regular objects which is undefined at the point that the library was linked. **−−no−as−needed** restores the default behaviour.

−−add−needed
−−no−add−needed
This option affects the treatment of dynamic libraries from ELF DT_NEEDED tags in dynamic libraries mentioned on the command line after the **−−no−add−needed** option. Normally, the linker will add a DT_NEEDED tag for each dynamic library from DT_NEEDED tags. **−−no−add−needed** causes DT_NEEDED tags will never be emitted for those libraries from DT_NEEDED tags. **−−add−needed** restores the default behaviour.

−assert *keyword*
This option is ignored for SunOS compatibility.

−Bdynamic
−dy
−call_shared
Link against dynamic libraries. This is only meaningful on platforms for which shared libraries are supported. This option is normally the default on such platforms. The different variants of this option are for compatibility with various systems. You may use this option multiple times on the command line: it affects library searching for **−l** options which follow it.

−Bgroup
Set the `DF_1_GROUP` flag in the `DT_FLAGS_1` entry in the dynamic section. This causes the run-time linker to handle lookups in this object and its dependencies to be performed only inside the group. **−−unresolved−symbols=report−all** is implied. This option is only meaningful on ELF platforms which support shared libraries.

−Bstatic
−dn
−non_shared
−static
Do not link against shared libraries. This is only meaningful on platforms for which shared libraries are supported. The different variants of this option are for compatibility with various systems. You may use this option multiple times on the command line: it affects library searching for **−l** options which follow it. This option also implies **−−unresolved−symbols=report−all**.

−Bsymbolic
When creating a shared library, bind references to global symbols to the definition within the shared library, if any. Normally, it is possible for a program linked against a shared library to override the definition within the shared library. This option is only meaningful on ELF platforms which support shared libraries.

−−check−sections
−−no−check−sections
Asks the linker *not* to check section addresses after they have been assigned to see if there any overlaps. Normally the linker will perform this check, and if it finds any overlaps it will produce suitable error messages. The linker does know about, and does make allowances for sections in overlays. The default behaviour can be restored by using the command line switch **−−check−sections**.

−−cref
Output a cross reference table. If a linker map file is being generated, the cross reference table is printed to the map file. Otherwise, it is printed on the standard output.

The format of the table is intentionally simple, so that it may be easily processed by a script if necessary. The symbols are printed out, sorted by name. For each symbol, a list of file names is given. If the symbol is defined, the first file listed is the location of the definition. The remaining files contain references to the symbol.

−−no−define−common
This option inhibits the assignment of addresses to common symbols. The script command `INHIBIT_COMMON_ALLOCATION` has the same effect.

The **--no-define-common** option allows decoupling the decision to assign addresses to Common symbols from the choice of the output file type; otherwise a non-Relocatable output type forces assigning addresses to Common symbols. Using **--no-define-common** allows Common symbols that are referenced from a shared library to be assigned addresses only in the main program. This eliminates the unused duplicate space in the shared library, and also prevents any possible confusion over resolving to the wrong duplicate when there are many dynamic modules with specialized search paths for runtime symbol resolution.

--defsym *symbol=expression*
Create a global symbol in the output file, containing the absolute address given by *expression*. You may use this option as many times as necessary to define multiple symbols in the command line. A limited form of arithmetic is supported for the *expression* in this context: you may give a hexadecimal constant or the name of an existing symbol, or use + and - to add or subtract hexadecimal constants or symbols. If you need more elaborate expressions, consider using the linker command language from a script. *Note:* there should be no white space between *symbol*, the equals sign ("="), and *expression*.

--demangle[=*style*]
--no-demangle
These options control whether to demangle symbol names in error messages and other output. When the linker is told to demangle, it tries to present symbol names in a readable fashion: it strips leading underscores if they are used by the object file format, and converts C++ mangled symbol names into user readable names. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler. The linker will demangle by default unless the environment variable **COLLECT_NO_DEMANGLE** is set. These options may be used to override the default.

--dynamic-linker *file*
Set the name of the dynamic linker. This is only meaningful when generating dynamically linked ELF executables. The default dynamic linker is normally correct; don't use this unless you know what you are doing.

--fatal-warnings
Treat all warnings as errors.

--force-exe-suffix
Make sure that an output file has a .exe suffix.

If a successfully built fully linked output file does not have a `.exe` or `.dll` suffix, this option forces the linker to copy the output file to one of the same name with a `.exe` suffix. This option is useful when using unmodified Unix makefiles on a Microsoft Windows host, since some versions of Windows won't run an image unless it ends in a `.exe` suffix.

--no-gc-sections
--gc-sections
Enable garbage collection of unused input sections. It is ignored on targets that do not support this option. This option is not compatible with **-r**. The default behaviour (of not performing this garbage collection) can be restored by specifying **--no-gc-sections** on the command line.

--help
Print a summary of the command-line options on the standard output and exit.

--target-help
Print a summary of all target specific options on the standard output and exit.

-Map *mapfile*
Print a link map to the file *mapfile*. See the description of the **-M** option, above.

--no-keep-memory
ld normally optimizes for speed over memory usage by caching the symbol tables of input files in memory. This option tells **ld** to instead optimize for memory usage, by rereading the symbol tables as necessary. This may be required if **ld** runs out of memory space while linking a large executable.

−−no−undefined
−z defs
Report unresolved symbol references from regular object files. This is done even if the linker is creating a non-symbolic shared library. The switch **−−[no−]allow−shlib−undefined** controls the behaviour for reporting unresolved references found in shared libraries being linked in.

−−allow−multiple−definition
−z muldefs
Normally when a symbol is defined multiple times, the linker will report a fatal error. These options allow multiple definitions and the first definition will be used.

−−allow−shlib−undefined
−−no−allow−shlib−undefined
Allows (the default) or disallows undefined symbols in shared libraries. This switch is similar to **−−no−undefined** except that it determines the behaviour when the undefined symbols are in a shared library rather than a regular object file. It does not affect how undefined symbols in regular object files are handled.

The reason that **−−allow−shlib−undefined** is the default is that the shared library being specified at link time may not be the same as the one that is available at load time, so the symbols might actually be resolvable at load time. Plus there are some systems, (eg BeOS) where undefined symbols in shared libraries is normal. (The kernel patches them at load time to select which function is most appropriate for the current architecture. This is used for example to dynamically select an appropriate memset function). Apparently it is also normal for HPPA shared libraries to have undefined symbols.

−−no−undefined−version
Normally when a symbol has an undefined version, the linker will ignore it. This option disallows symbols with undefined version and a fatal error will be issued instead.

−−default−symver
Create and use a default symbol version (the soname) for unversioned exported symbols.

−−default−imported−symver
Create and use a default symbol version (the soname) for unversioned imported symbols.

−−no−warn−mismatch
Normally **ld** will give an error if you try to link together input files that are mismatched for some reason, perhaps because they have been compiled for different processors or for different endiannesses. This option tells **ld** that it should silently permit such possible errors. This option should only be used with care, in cases when you have taken some special action that ensures that the linker errors are inappropriate.

−−no−whole−archive
Turn off the effect of the **−−whole−archive** option for subsequent archive files.

−−noinhibit−exec
Retain the executable output file whenever it is still usable. Normally, the linker will not produce an output file if it encounters errors during the link process; it exits without writing an output file when it issues any error whatsoever.

−nostdlib
Only search library directories explicitly specified on the command line. Library directories specified in linker scripts (including linker scripts specified on the command line) are ignored.

−−oformat *output-format*
ld may be configured to support more than one kind of object file. If your **ld** is configured this way, you can use the **−−oformat** option to specify the binary format for the output object file. Even when **ld** is configured to support alternative object formats, you don't usually need to specify this, as **ld** should be configured to produce as a default output format the most usual format on each machine. *output-format* is a text string, the name of a particular format supported by the BFD libraries. (You can list the available binary formats with **objdump −i**.) The script command `OUTPUT_FORMAT` can also

specify the output format, but this option overrides it.

−pie
−−pic−executable
Create a position independent executable. This is currently only supported on ELF platforms. Position independent executables are similar to shared libraries in that they are relocated by the dynamic linker to the virtual address the OS chooses for them (which can vary between invocations). Like normal dynamically linked executables they can be executed and symbols defined in the executable cannot be overridden by shared libraries.

−qmagic
This option is ignored for Linux compatibility.

−Qy
This option is ignored for SVR4 compatibility.

−−relax
An option with machine dependent effects. This option is only supported on a few targets.

On some platforms, the **−−relax** option performs global optimizations that become possible when the linker resolves addressing in the program, such as relaxing address modes and synthesizing new instructions in the output object file.

On some platforms these link time global optimizations may make symbolic debugging of the resulting executable impossible. This is known to be the case for the Matsushita MN10200 and MN10300 family of processors.

On platforms where this is not supported, **−−relax** is accepted, but ignored.

−−retain−symbols−file *filename*
Retain *only* the symbols listed in the file *filename*, discarding all others. *filename* is simply a flat file, with one symbol name per line. This option is especially useful in environments (such as VxWorks) where a large global symbol table is accumulated gradually, to conserve run-time memory.

−−retain−symbols−file does *not* discard undefined symbols, or symbols needed for relocations.

You may only specify **−−retain−symbols−file** once in the command line. It overrides **−s** and **−S**.

−rpath *dir*
Add a directory to the runtime library search path. This is used when linking an ELF executable with shared objects. All **−rpath** arguments are concatenated and passed to the runtime linker, which uses them to locate shared objects at runtime. The **−rpath** option is also used when locating shared objects which are needed by shared objects explicitly included in the link; see the description of the **−rpath−link** option. If **−rpath** is not used when linking an ELF executable, the contents of the environment variable `LD_RUN_PATH` will be used if it is defined.

The **−rpath** option may also be used on SunOS. By default, on SunOS, the linker will form a runtime search patch out of all the **−L** options it is given. If a **−rpath** option is used, the runtime search path will be formed exclusively using the **−rpath** options, ignoring the **−L** options. This can be useful when using gcc, which adds many **−L** options which may be on NFS mounted filesystems.

For compatibility with other ELF linkers, if the **−R** option is followed by a directory name, rather than a file name, it is treated as the **−rpath** option.

−rpath−link *DIR*
When using ELF or SunOS, one shared library may require another. This happens when an `ld −shared` link includes a shared library as one of the input files.

When the linker encounters such a dependency when doing a non−shared, non-relocatable link, it will automatically try to locate the required shared library and include it in the link, if it is not included explicitly. In such a case, the **−rpath−link** option specifies the first set of directories to search. The **−rpath−link** option may specify a sequence of directory names either by specifying a list of names separated by colons, or by appearing multiple times.

This option should be used with caution as it overrides the search path that may have been hard compiled into a shared library. In such a case it is possible to use unintentionally a different search path than the runtime linker would do.

The linker uses the following search paths to locate required shared libraries.

1. Any directories specified by **−rpath−link** options.
2. Any directories specified by **−rpath** options. The difference between **−rpath** and **−rpath−link** is that directories specified by **−rpath** options are included in the executable and used at runtime, whereas the **−rpath−link** option is only effective at link time. It is for the native linker only.
3. On an ELF system, if the **−rpath** and `rpath−link` options were not used, search the contents of the environment variable `LD_RUN_PATH`. It is for the native linker only.
4. On SunOS, if the **−rpath** option was not used, search any directories specified using **−L** options.
5. For a native linker, the contents of the environment variable `LD_LIBRARY_PATH`.
6. For a native ELF linker, the directories in `DT_RUNPATH` or `DT_RPATH` of a shared library are searched for shared libraries needed by it. The `DT_RPATH` entries are ignored if `DT_RUNPATH` entries exist.
7. The default directories, normally */lib* and */usr/lib*.
8. For a native linker on an ELF system, if the file */etc/ld.so.conf* exists, the list of directories found in that file.

If the required shared library is not found, the linker will issue a warning and continue with the link.

−shared
−Bshareable
Create a shared library. This is currently only supported on ELF, XCOFF and SunOS platforms. On SunOS, the linker will automatically create a shared library if the **−e** option is not used and there are undefined symbols in the link.

−−sort−common
This option tells **ld** to sort the common symbols by size when it places them in the appropriate output sections. First come all the one byte symbols, then all the two byte, then all the four byte, and then everything else. This is to prevent gaps between symbols due to alignment constraints.

−−sort−section name
This option will apply `SORT_BY_NAME` to all wildcard section patterns in the linker script.

−−sort−section alignment
This option will apply `SORT_BY_ALIGNMENT` to all wildcard section patterns in the linker script.

−−split−by−file [*size***]**
Similar to **−−split−by−reloc** but creates a new output section for each input file when *size* is reached. *size* defaults to a size of 1 if not given.

−−split−by−reloc [*count***]**
Tries to creates extra sections in the output file so that no single output section in the file contains more than *count* relocations. This is useful when generating huge relocatable files for downloading into certain real time kernels with the COFF object file format; since COFF cannot represent more than 65535 relocations in a single section. Note that this will fail to work with object file formats which do not support arbitrary sections. The linker will not split up individual input sections for redistribution, so if a single input section contains more than *count* relocations one output section will contain that many relocations. *count* defaults to a value of 32768.

−−stats
Compute and display statistics about the operation of the linker, such as execution time and memory usage.

−−sysroot=*directory*
Use *directory* as the location of the sysroot, overriding the configure-time default. This option is only supported by linkers that were configured using **−−with−sysroot**.

−−traditional−format
For some targets, the output of **ld** is different in some ways from the output of some existing linker. This switch requests **ld** to use the traditional format instead.

For example, on SunOS, **ld** combines duplicate entries in the symbol string table. This can reduce the size of an output file with full debugging information by over 30 percent. Unfortunately, the SunOS `dbx` program can not read the resulting program (`gdb` has no trouble). The **−−traditional−format** switch tells **ld** to not combine duplicate entries.

−−section−start *sectionname=org*
Locate a section in the output file at the absolute address given by *org*. You may use this option as many times as necessary to locate multiple sections in the command line. *org* must be a single hexadecimal integer; for compatibility with other linkers, you may omit the leading **0x** usually associated with hexadecimal values. *Note:* there should be no white space between *sectionname*, the equals sign (‘‘=’’), and *org*.

−Tbss *org*
−Tdata *org*
−Ttext *org*
Same as −−section−start, with `.bss`, `.data` or `.text` as the *sectionname*.

−−unresolved−symbols=*method*
Determine how to handle unresolved symbols. There are four possible values for **method**:

ignore-all
Do not report any unresolved symbols.

report-all
Report all unresolved symbols. This is the default.

ignore-in-object-files
Report unresolved symbols that are contained in shared libraries, but ignore them if they come from regular object files.

ignore-in-shared-libs
Report unresolved symbols that come from regular object files, but ignore them if they come from shared libraries. This can be useful when creating a dynamic binary and it is known that all the shared libraries that it should be referencing are included on the linker’s command line.

The behaviour for shared libraries on their own can also be controlled by the **−−[no−]allow−shlib−undefined** option.

Normally the linker will generate an error message for each reported unresolved symbol but the option **−−warn−unresolved−symbols** can change this to a warning.

−−dll−verbose
−−verbose
Display the version number for **ld** and list the linker emulations supported. Display which input files can and cannot be opened. Display the linker script being used by the linker.

−−version−script=*version-scriptfile*
Specify the name of a version script to the linker. This is typically used when creating shared libraries to specify additional information about the version hierarchy for the library being created. This option is only meaningful on ELF platforms which support shared libraries.

−−warn−common
Warn when a common symbol is combined with another common symbol or with a symbol definition. Unix linkers allow this somewhat sloppy practise, but linkers on some other operating systems do not. This option allows you to find potential problems from combining global symbols. Unfortunately,

some C libraries use this practise, so you may get some warnings about symbols in the libraries as well as in your programs.

There are three kinds of global symbols, illustrated here by C examples:

int i = 1;
A definition, which goes in the initialized data section of the output file.

extern int i;
An undefined reference, which does not allocate space. There must be either a definition or a common symbol for the variable somewhere.

int i;
A common symbol. If there are only (one or more) common symbols for a variable, it goes in the uninitialized data area of the output file. The linker merges multiple common symbols for the same variable into a single symbol. If they are of different sizes, it picks the largest size. The linker turns a common symbol into a declaration, if there is a definition of the same variable.

The **--warn-common** option can produce five kinds of warnings. Each warning consists of a pair of lines: the first describes the symbol just encountered, and the second describes the previous symbol encountered with the same name. One or both of the two symbols will be a common symbol.

1. Turning a common symbol into a reference, because there is already a definition for the symbol.

```
<file>(<section>): warning: common of `<symbol>'
   overridden by definition
<file>(<section>): warning: defined here
```

2. Turning a common symbol into a reference, because a later definition for the symbol is encountered. This is the same as the previous case, except that the symbols are encountered in a different order.

```
<file>(<section>): warning: definition of `<symbol>'
   overriding common
<file>(<section>): warning: common is here
```

3. Merging a common symbol with a previous same-sized common symbol.

```
<file>(<section>): warning: multiple common
   of `<symbol>'
<file>(<section>): warning: previous common is here
```

4. Merging a common symbol with a previous larger common symbol.

```
<file>(<section>): warning: common of `<symbol>'
   overridden by larger common
<file>(<section>): warning: larger common is here
```

5. Merging a common symbol with a previous smaller common symbol. This is the same as the previous case, except that the symbols are encountered in a different order.

```
<file>(<section>): warning: common of `<symbol>'
   overriding smaller common
<file>(<section>): warning: smaller common is here
```

--warn-constructors
Warn if any global constructors are used. This is only useful for a few object file formats. For formats like COFF or ELF, the linker can not detect the use of global constructors.

--warn-multiple-gp
Warn if multiple global pointer values are required in the output file. This is only meaningful for certain processors, such as the Alpha. Specifically, some processors put large-valued constants in a special section. A special register (the global pointer) points into the middle of this section, so that constants can be loaded efficiently via a base-register relative addressing mode. Since the offset in base-

register relative mode is fixed and relatively small (e.g., 16 bits), this limits the maximum size of the constant pool. Thus, in large programs, it is often necessary to use multiple global pointer values in order to be able to address all possible constants. This option causes a warning to be issued whenever this case occurs.

--warn-once
Only warn once for each undefined symbol, rather than once per module which refers to it.

--warn-section-align
Warn if the address of an output section is changed because of alignment. Typically, the alignment will be set by an input section. The address will only be changed if it not explicitly specified; that is, if the `SECTIONS` command does not specify a start address for the section.

--warn-shared-textrel
Warn if the linker adds a DT_TEXTREL to a shared object.

--warn-unresolved-symbols
If the linker is going to report an unresolved symbol (see the option **--unresolved-symbols**) it will normally generate an error. This option makes it generate a warning instead.

--error-unresolved-symbols
This restores the linker's default behaviour of generating errors when it is reporting unresolved symbols.

--whole-archive
For each archive mentioned on the command line after the **--whole-archive** option, include every object file in the archive in the link, rather than searching the archive for the required object files. This is normally used to turn an archive file into a shared library, forcing every object to be included in the resulting shared library. This option may be used more than once.

Two notes when using this option from gcc: First, gcc doesn't know about this option, so you have to use **-Wl,-whole-archive**. Second, don't forget to use **-Wl,-no-whole-archive** after your list of archives, because gcc will add its own list of archives to your link and you may not want this flag to affect those as well.

--wrap *symbol*
Use a wrapper function for *symbol*. Any undefined reference to *symbol* will be resolved to `__wrap_`*symbol*. Any undefined reference to `__real_`*symbol* will be resolved to *symbol*.

This can be used to provide a wrapper for a system function. The wrapper function should be called `__wrap_`*symbol*. If it wishes to call the system function, it should call `__real_`*symbol*.

Here is a trivial example:

```
void *
__wrap_malloc (size_t c)
{
  printf ("malloc called with %zu\n", c);
  return __real_malloc (c);
}
```

If you link other code with this file using **--wrap malloc**, then all calls to `malloc` will call the function `__wrap_malloc` instead. The call to `__real_malloc` in `__wrap_malloc` will call the real `malloc` function.

You may wish to provide a `__real_malloc` function as well, so that links without the **--wrap** option will succeed. If you do this, you should not put the definition of `__real_malloc` in the same file as `__wrap_malloc`; if you do, the assembler may resolve the call before the linker has a chance to wrap it to `malloc`.

--enable-new-dtags
--disable-new-dtags
This linker can create the new dynamic tags in ELF. But the older ELF systems may not understand them. If you specify **--enable-new-dtags**, the dynamic tags will be created as needed. If you specify **--disable-new-dtags**, no new dynamic tags will be created. By default, the new dynamic tags are not created. Note that those options are only available for ELF systems.

--hash-size=*number*
Set the default size of the linker's hash tables to a prime number close to *number*. Increasing this value can reduce the length of time it takes the linker to perform its tasks, at the expense of increasing the linker's memory requirements. Similarly reducing this value can reduce the memory requirements at the expense of speed.

--reduce-memory-overheads
This option reduces memory requirements at ld runtime, at the expense of linking speed. This was introduced to to select the old O(n^2) algorithm for link map file generation, rather than the new O(n) algorithm which uses about 40% more memory for symbol storage.

Another affect of the switch is to set the default hash table size to 1021, which again saves memory at the cost of lengthening the linker's run time. This is not done however if the **--hash-size** switch has been used.

The **--reduce-memory-overheads** switch may be also be used to enable other tradeoffs in future versions of the linker.

The i386 PE linker supports the **-shared** option, which causes the output to be a dynamically linked library (DLL) instead of a normal executable. You should name the output `*.dll` when you use this option. In addition, the linker fully supports the standard `*.def` files, which may be specified on the linker command line like an object file (in fact, it should precede archives it exports symbols from, to ensure that they get linked in, just like a normal object file).

In addition to the options common to all targets, the i386 PE linker support additional command line options that are specific to the i386 PE target. Options that take values may be separated from their values by either a space or an equals sign.

--add-stdcall-alias
If given, symbols with a stdcall suffix (*@nn*) will be exported as-is and also with the suffix stripped. [This option is specific to the i386 PE targeted port of the linker]

--base-file *file*
Use *file* as the name of a file in which to save the base addresses of all the relocations needed for generating DLLs with *dlltool*. [This is an i386 PE specific option]

--dll
Create a DLL instead of a regular executable. You may also use **-shared** or specify a `LIBRARY` in a given `.def` file. [This option is specific to the i386 PE targeted port of the linker]

--enable-stdcall-fixup
--disable-stdcall-fixup
If the link finds a symbol that it cannot resolve, it will attempt to do ''fuzzy linking'' by looking for another defined symbol that differs only in the format of the symbol name (cdecl vs stdcall) and will resolve that symbol by linking to the match. For example, the undefined symbol `_foo` might be linked to the function `_foo@12`, or the undefined symbol `_bar@16` might be linked to the function `_bar`. When the linker does this, it prints a warning, since it normally should have failed to link, but sometimes import libraries generated from third-party dlls may need this feature to be usable. If you specify **--enable-stdcall-fixup**, this feature is fully enabled and warnings are not printed. If you specify **--disable-stdcall-fixup**, this feature is disabled and such mismatches are considered to be errors. [This option is specific to the i386 PE targeted port of the linker]

--export-all-symbols
If given, all global symbols in the objects used to build a DLL will be exported by the DLL. Note that this is the default if there otherwise wouldn't be any exported symbols. When symbols are explicitly exported via DEF files or implicitly exported via function attributes, the default is to not export anything else unless this option is given. Note that the symbols `DllMain@12`, `DllEntryPoint@0`, `DllMainCRTStartup@12`, and `impure_ptr` will not be automatically exported. Also, symbols imported from other DLLs will not be re-exported, nor will symbols specifying the DLL's internal layout such as those beginning with `_head_` or ending with `_iname`. In addition, no symbols from `libgcc`, `libstd++`, `libmingw32`, or `crtX.o` will be exported. Symbols whose names begin with `__rtti_` or `__builtin_` will not be exported, to help with C++ DLLs. Finally, there is an extensive list of cygwin-private symbols that are not exported (obviously, this applies on when building DLLs for cygwin targets). These cygwin-excludes are: `_cygwin_dll_entry@12`, `_cygwin_crt0_common@8`, `_cygwin_noncygwin_dll_entry@12`, `_fmode`, `_impure_ptr`, `cygwin_attach_dll`, `cygwin_premain0`, `cygwin_premain1`, `cygwin_premain2`, `cygwin_premain3`, and `environ`. [This option is specific to the i386 PE targeted port of the linker]

--exclude-symbols *symbol,symbol,...*
Specifies a list of symbols which should not be automatically exported. The symbol names may be delimited by commas or colons. [This option is specific to the i386 PE targeted port of the linker]

--file-alignment
Specify the file alignment. Sections in the file will always begin at file offsets which are multiples of this number. This defaults to 512. [This option is specific to the i386 PE targeted port of the linker]

--heap *reserve*
--heap *reserve,commit*
Specify the amount of memory to reserve (and optionally commit) to be used as heap for this program. The default is 1Mb reserved, 4K committed. [This option is specific to the i386 PE targeted port of the linker]

--image-base *value*
Use *value* as the base address of your program or dll. This is the lowest memory location that will be used when your program or dll is loaded. To reduce the need to relocate and improve performance of your dlls, each should have a unique base address and not overlap any other dlls. The default is 0x400000 for executables, and 0x10000000 for dlls. [This option is specific to the i386 PE targeted port of the linker]

--kill-at
If given, the stdcall suffixes (*@nn*) will be stripped from symbols before they are exported. [This option is specific to the i386 PE targeted port of the linker]

--large-address-aware
If given, the appropriate bit in the ''Charateristics'' field of the COFF header is set to indicate that this executable supports virtual addresses greater than 2 gigabytes. This should be used in conjuction with the /3GB or /USERVA=*value* megabytes switch in the ''[operating systems]'' section of the BOOT.INI. Otherwise, this bit has no effect. [This option is specific to PE targeted ports of the linker]

--major-image-version *value*
Sets the major number of the ''image version''. Defaults to 1. [This option is specific to the i386 PE targeted port of the linker]

--major-os-version *value*
Sets the major number of the ''os version''. Defaults to 4. [This option is specific to the i386 PE targeted port of the linker]

--major-subsystem-version *value*
Sets the major number of the ''subsystem version''. Defaults to 4. [This option is specific to the i386 PE targeted port of the linker]

--minor-image-version *value*
Sets the minor number of the ''image version''. Defaults to 0. [This option is specific to the i386 PE targeted port of the linker]

--minor-os-version *value*
Sets the minor number of the ''os version''. Defaults to 0. [This option is specific to the i386 PE targeted port of the linker]

--minor-subsystem-version *value*
Sets the minor number of the ''subsystem version''. Defaults to 0. [This option is specific to the i386 PE targeted port of the linker]

--output-def *file*
The linker will create the file *file* which will contain a DEF file corresponding to the DLL the linker is generating. This DEF file (which should be called `*.def`) may be used to create an import library with `dlltool` or may be used as a reference to automatically or implicitly exported symbols. [This option is specific to the i386 PE targeted port of the linker]

--out-implib *file*
The linker will create the file *file* which will contain an import lib corresponding to the DLL the linker is generating. This import lib (which should be called `*.dll.a` or `*.a` may be used to link clients against the generated DLL; this behaviour makes it possible to skip a separate `dlltool` import library creation step. [This option is specific to the i386 PE targeted port of the linker]

--enable-auto-image-base
Automatically choose the image base for DLLs, unless one is specified using the `--image-base` argument. By using a hash generated from the dllname to create unique image bases for each DLL, in-memory collisions and relocations which can delay program execution are avoided. [This option is specific to the i386 PE targeted port of the linker]

--disable-auto-image-base
Do not automatically generate a unique image base. If there is no user-specified image base (`--image-base`) then use the platform default. [This option is specific to the i386 PE targeted port of the linker]

--dll-search-prefix *string*
When linking dynamically to a dll without an import library, search for `<string><basename>.dll` in preference to `lib<basename>.dll`. This behaviour allows easy distinction between DLLs built for the various ''subplatforms'': native, cygwin, uwin, pw, etc. For instance, cygwin DLLs typically use `--dll-search-prefix=cyg`. [This option is specific to the i386 PE targeted port of the linker]

--enable-auto-import
Do sophisticated linking of `_symbol` to `__imp__symbol` for DATA imports from DLLs, and create the necessary thunking symbols when building the import libraries with those DATA exports. Note: Use of the 'auto-import' extension will cause the text section of the image file to be made writable. This does not conform to the PE-COFF format specification published by Microsoft.

Using 'auto-import' generally will 'just work' — but sometimes you may see this message:

"variable '<var>' can't be auto-imported. Please read the documentation for ld's `--enable-auto-import` for details."

This message occurs when some (sub)expression accesses an address ultimately given by the sum of two constants (Win32 import tables only allow one). Instances where this may occur include accesses to member fields of struct variables imported from a DLL, as well as using a constant index into an array variable imported from a DLL. Any multiword variable (arrays, structs, long long, etc) may trigger this error condition. However, regardless of the exact data type of the offending exported variable, ld will always detect it, issue the warning, and exit.

There are several ways to address this difficulty, regardless of the data type of the exported variable:

One way is to use --enable-runtime-pseudo-reloc switch. This leaves the task of adjusting references in your client code for runtime environment, so this method works only when runtime environment supports this feature.

A second solution is to force one of the 'constants' to be a variable — that is, unknown and un-optimizable at compile time. For arrays, there are two possibilities: a) make the indexee (the array's address) a variable, or b) make the 'constant' index a variable. Thus:

```
extern type extern_array[];
extern_array[1] -->
   { volatile type *t=extern_array; t[1] }
```

or

```
extern type extern_array[];
extern_array[1] -->
   { volatile int t=1; extern_array[t] }
```

For structs (and most other multiword data types) the only option is to make the struct itself (or the long long, or the ...) variable:

```
extern struct s extern_struct;
extern_struct.field -->
   { volatile struct s *t=&extern_struct; t->field }
```

or

```
extern long long extern_ll;
extern_ll -->
  { volatile long long * local_ll=&extern_ll; *local_ll }
```

A third method of dealing with this difficulty is to abandon 'auto-import' for the offending symbol and mark it with `__declspec(dllimport)`. However, in practise that requires using compile-time #defines to indicate whether you are building a DLL, building client code that will link to the DLL, or merely building/linking to a static library. In making the choice between the various methods of resolving the 'direct address with constant offset' problem, you should consider typical real-world usage:

Original:

```
--foo.h
extern int arr[];
--foo.c
#include "foo.h"
void main(int argc, char **argv){
  printf("%d\n",arr[1]);
}
```

Solution 1:

```
--foo.h
extern int arr[];
--foo.c
#include "foo.h"
void main(int argc, char **argv){
  /* This workaround is for win32 and cygwin; do not "optimize" */
  volatile int *parr = arr;
  printf("%d\n",parr[1]);
}
```

Solution 2:

```
--foo.h
/* Note: auto-export is assumed (no __declspec(dllexport)) */
#if (defined(_WIN32) || defined(__CYGWIN__)) && \
  !(defined(FOO_BUILD_DLL) || defined(FOO_STATIC))
#define FOO_IMPORT __declspec(dllimport)
#else
#define FOO_IMPORT
#endif
extern FOO_IMPORT int arr[];
--foo.c
#include "foo.h"
void main(int argc, char **argv){
  printf("%d\n",arr[1]);
}
```

A fourth way to avoid this problem is to re-code your library to use a functional interface rather than a data interface for the offending variables (e.g. *set_foo()* and *get_foo()* accessor functions). [This option is specific to the i386 PE targeted port of the linker]

--disable-auto-import
Do not attempt to do sophisticated linking of `_symbol` to `__imp__symbol` for DATA imports from DLLs. [This option is specific to the i386 PE targeted port of the linker]

--enable-runtime-pseudo-reloc
If your code contains expressions described in --enable-auto-import section, that is, DATA imports from DLL with non-zero offset, this switch will create a vector of 'runtime pseudo relocations' which can be used by runtime environment to adjust references to such data in your client code. [This option is specific to the i386 PE targeted port of the linker]

--disable-runtime-pseudo-reloc
Do not create pseudo relocations for non-zero offset DATA imports from DLLs. This is the default. [This option is specific to the i386 PE targeted port of the linker]

--enable-extra-pe-debug
Show additional debug info related to auto-import symbol thunking. [This option is specific to the i386 PE targeted port of the linker]

--section-alignment
Sets the section alignment. Sections in memory will always begin at addresses which are a multiple of this number. Defaults to 0x1000. [This option is specific to the i386 PE targeted port of the linker]

--stack *reserve*
--stack *reserve*,*commit*
Specify the amount of memory to reserve (and optionally commit) to be used as stack for this program. The default is 2Mb reserved, 4K committed. [This option is specific to the i386 PE targeted port of the linker]

--subsystem *which*
--subsystem *which*:*major*
--subsystem *which*:*major*.*minor*
Specifies the subsystem under which your program will execute. The legal values for *which* are `native`, `windows`, `console`, `posix`, and `xbox`. You may optionally set the subsystem version also. Numeric values are also accepted for *which*. [This option is specific to the i386 PE targeted port of the linker]

The 68HC11 and 68HC12 linkers support specific options to control the memory bank switching mapping and trampoline code generation.

--no-trampoline
This option disables the generation of trampoline. By default a trampoline is generated for each far function which is called using a `jsr` instruction (this happens when a pointer to a far function is taken).

--bank-window *name*
This option indicates to the linker the name of the memory region in the **MEMORY** specification that describes the memory bank window. The definition of such region is then used by the linker to compute paging and addresses within the memory window.

ENVIRONMENT

You can change the behaviour of **ld** with the environment variables `GNUTARGET`, `LDEMULATION` and `COLLECT_NO_DEMANGLE`.

`GNUTARGET` determines the input-file object format if you don't use **-b** (or its synonym **--format**). Its value should be one of the BFD names for an input format. If there is no `GNUTARGET` in the environment, **ld** uses the natural format of the target. If `GNUTARGET` is set to `default` then BFD attempts to discover the input format by examining binary input files; this method often succeeds, but there are potential ambiguities, since there is no method of ensuring that the magic number used to specify object-file formats is unique. However, the configuration procedure for BFD on each system places the conventional format for that system first in the search-list, so ambiguities are resolved in favor of convention.

`LDEMULATION` determines the default emulation if you don't use the **-m** option. The emulation can affect various aspects of linker behaviour, particularly the default linker script. You can list the available emulations with the **--verbose** or **-V** options. If the **-m** option is not used, and the `LDEMULATION` environment variable is not defined, the default emulation depends upon how the linker was configured.

Normally, the linker will default to demangling symbols. However, if `COLLECT_NO_DEMANGLE` is set in the environment, then it will default to not demangling symbols. This environment variable is used in a similar fashion by the `gcc` linker wrapper program. The default may be overridden by the **--demangle** and **--no-demangle** options.

SEE ALSO

ar (1), *nm* (1), *objcopy* (1), *objdump* (1), *readelf* (1) and the Info entries for *binutils* and *ld*.

COPYRIGHT

NAME

ld.so — run-time link-editor

DESCRIPTION

ld.so is a self-contained, position independent program image providing run-time support for loading and link-editing shared objects into a process' address space. It uses the data structures (see link(5)) contained within dynamically linked programs to determine which shared libraries are needed and loads them at a convenient virtual address using the mmap(2) system call.

After all shared libraries have been successfully loaded, **ld.so** proceeds to resolve external references from both the main program and all objects loaded. A mechanism is provided for initialization routines to be called, on a per-object basis, giving a shared object an opportunity to perform any extra set-up, before execution of the program proper begins. **ld.so** looks for a symbol named *.init* in each object's symbol table. If present, this symbol is assumed to represent a C-function declared as *void* **.init**(*void*), which is then called. Similarly, a *void* **.fini**(*void*) function is called just before an object is unloaded from the process address space as a result of calling dlclose(3). Note that while an object's *.init* is always called, whether the object is loaded automatically at program startup or programmatically by using dlopen(3), the *.fini* function is called only on 'last dlclose(3)'.

This mechanism is exploited by the system-supplied C++ constructor initialization code located in /usr/lib/c++rt.o. This file should be included in the list of object-code files passed to ld(1) when building a shared C++ library.

ld.so is itself a shared object that is initially loaded by the startup module *crt0*. Since a.out(5) formats do not provide easy access to the file header from within a running process, *crt0* uses the special symbol *_DYNAMIC* to determine whether a program is in fact dynamically linked or not. Whenever the linker ld(1) has relocated this symbol to a location other than 0, *crt0* assumes the services of **ld.so** are needed (see link(5) for details). *crt0* passes control to **rtld**'s entry point before the program's **main**() routine is called. Thus, **ld.so** can complete the link-editing process before the dynamic program calls upon services of any dynamic library.

To quickly locate the required shared objects in the filesystem, **ld.so** may use a "hints" file, prepared by the ldconfig(8) utility, in which the full path specification of the shared objects can be looked up by hashing on the 3-tuple ⟨library-name, major-version-number, minor-version-number⟩.

ld.so recognizes a number of environment variables that can be used to modify its behavior as follows:

LD_LIBRARY_PATH
: A colon separated list of directories, overriding the default search path for shared libraries.

LD_PRELOAD
: A colon separated list of shared object filenames to be loaded *after* the main program but *before* its shared object dependencies.

LD_WARN_NON_PURE_CODE
: When set, issue a warning whenever a link-editing operation requires modification of the text segment of some loaded object. This is usually indicative of an incorrectly built library.

LD_SUPPRESS_WARNINGS
: When set, no warning messages of any kind are issued. Normally, a warning is given if satisfactorily versioned library could not be found.

LD_TRACE_LOADED_OBJECTS
: When set, causes **ld.so** to exit after loading the shared objects and printing a summary which includes the absolute pathnames of all objects, to standard output.

LD_TRACE_LOADED_OBJECTS_FMT1

`LD_TRACE_LOADED_OBJECTS_FMT2`
When set, these variables are interpreted as format strings a la `printf`(3) to customize the trace output and are used by `ldd`(1)'s **`-f`** option and allows `ldd`(1) to be operated as a filter more conveniently. The following conversions can be used:

%a The main program's name (also known as "__progname").

%A The value of the environment variable `LD_TRACE_LOADED_OBJECTS_PROGNAME`

%o The library name.

%m The library's major version number.

%n The library's minor version number.

%p The full pathname as determined by **`rtld`**'s library search rules.

%x The library's load address.

Additionally, **\n** and **\t** are recognized and have their usual meaning.

`LD_NO_INTERN_SEARCH` When set, **`ld.so`** does not process any internal search paths that were recorded in the executable.

`LD_NOSTD_PATH` When set, do not include a set of built-in standard directory paths for searching. This might be useful when running on a system with a completely non-standard filesystem layout.

FILES

`/var/run/ld.so.hints` library location hints built by `ldconfig`(8)

SEE ALSO

`ld`(1), `ld.elf_so`(1), `ld.so`(1), `link`(5), `ldconfig`(8)

HISTORY

The shared library model employed first appeared in SunOS 4.0.

SECURITY CONSIDERATIONS

The environment variables `LD_LIBRARY_PATH` and `LD_PRELOAD` are not honored when executing in a set-user-ID or set-group-ID environment. This action is taken to prevent malicious substitution of shared object dependencies or interposition of symbols.

NAME

ld.elf_so — run-time link-editor (linker)

DESCRIPTION

ld.elf_so is a self-contained, position independent program image providing run-time support for loading and link-editing shared objects into a process' address space. It uses information stored in data structures within the binary (see elf(5)) and environment variables to determine which shared objects are needed. These shared objects are loaded at a convenient virtual address using the mmap(2) system call. After all shared objects have been successfully loaded, **ld.elf_so** proceeds to resolve external references from both the main program and all objects loaded. Once all required references are resolved control is passed to the program via its entry point.

Startup

On the execution of a dynamically linked binary the kernel will load the program and its run-time linker as specified in the PT_INTERP section in the program header. At this point, instead of passing control directly to the program, the kernel passes control to the specified linker. An auxiliary vector of information is passed that includes the address of the program header, the size of each entry in the header, and the number of entries. The entry point of the program and the base address of where **ld.elf_so** is loaded is also supplied.

Careful use of code allows **ld.elf_so** to relocate itself before proceeding. Specifically the use of global variables and large switch statements is not allowed. The later can cause the output of a jump table that can use the equivalent of a global variable.

Finding objects

Each elf(5) object file may contain information in its dynamic (PT_DYNAMIC) section about which shared objects it requires (often referred to as dependencies). These dependencies are specified in the optional DT_NEEDED entry within the dynamic section. Each DT_NEEDED entry refers to a filename string of the shared object that is to be searched for.

The linker will search for libraries in three lists of paths:

1. A user defined list of paths as specified in LD_LIBRARY_PATH and ld.so.conf(5).

   The use of ld.so.conf should be avoided as the setting of a global search path can present a security risk.

2. A list of paths specified within a shared object using a DT_RPATH entry in the dynamic section. This is defined at shared object link time.

3. The list of default paths which is set to /usr/lib.

ld.elf_so will expand the following variables if present in the paths:

$HWCAP Processor hardware capabilities, for example FPU, MMX, SSE. Currently unimplemented.

$ISALIST List of instructions sets this processor can execute. Currently unimplemented.

$ORIGIN The directory of the main object.

$OSNAME The value of the kern.ostype sysctl(3).

$OSREL The value of the kern.osrelease sysctl(3).

$PLATFORM
The value of the hw.machine_arch sysctl(3).

Both ${VARIABLE} and $VARIABLE are recognized.

The filename string can be considered free form, however, it will almost always take the form lib<name>.so.<number>, where name specifies the 'library' name and number is conceptually the library's major version number.

This name and another of the form lib<name>.so are normally symbolic links to the real shared object which has a filename of the form lib<name>.so.<major>.<minor>[.<teeny>]. This naming convention allows a versioning scheme similar to `a.out`(5).

Relocation

`ld.elf_so` will perform all necessary relocations immediately except for relocations relating to the Procedure Linkage Table (PLT). The PLT is used as a indirection method for procedure calls to globally defined functions. It allows, through the use of intermediate code, the delayed binding of a call to a globally defined function to be performed at procedure call time. This 'lazy' method is the default (see LD_BIND_NOW).

Initialization

A mechanism is provided for initialization and termination routines to be called, on a per-object basis before execution of the program proper begins or after the program has completed. This gives a shared object an opportunity to perform any extra set-up or completion work.

The DT_INIT and DT_FINI entries in the dynamic section specify the addresses of the initialization and termination functions, respectively, for the shared object. **`ld.elf_so`** arranges for each initialization function to be called before control is passed to the program and for the termination functions to be called by using `atexit`(3).

This mechanism is exploited by the system-supplied constructor initialization and destructor code located in `/usr/lib/crtbeginS.o` and `/usr/lib/crtendS.o`. These files are automatically included by `cc`(1) and `c++`(1) in the list of object-code files passed to `ld`(1) when building a shared C or C++ object.

ENVIRONMENT

If the following environment variables exist they will be used by **`ld.elf_so`**.

`LD_LIBRARY_PATH` A colon separated list of directories, overriding the default search path for shared libraries.

`LD_PRELOAD` A colon or space separated list of shared object filenames to be loaded *after* the main program but *before* its shared object dependencies. Space is allowed as a separator for backwards compatibility only. Support may be removed in a future release and should not be relied upon.

`LD_BIND_NOW` If defined immediate binding of Procedure Link Table (PLT) entries is performed instead of the default lazy method.

`LD_DEBUG` If defined a variety of debug information will be written to the standard error of an dynamically linked executed when it is run. This variable is only recognized if **`ld.elf_so`** was compiled with debugging support (**-DDEBUG**).

FILES

`/etc/ld.so.conf` library location hints supplied by the system administrator.

SEE ALSO

`ld`(1), `ld.aout_so`(1), `dlfcn`(3), `elf`(5)

HISTORY

The ELF shared library model employed first appeared in Sys V R4.

The path expansion variables first appeared in Solaris 10, and in NetBSD 5.0.

SECURITY CONSIDERATIONS

The environment variables `LD_LIBRARY_PATH` and `LD_PRELOAD` are not honored when executing in a set-user-ID or set-group-ID environment. This action is taken to prevent malicious substitution of shared object dependencies or interposition of symbols.

NAME

ld.so — run-time link-editor

DESCRIPTION

ld.so is a self-contained, position independent program image providing run-time support for loading and link-editing shared objects into a process' address space.

For the particular details, refer to ld.aout_so(1) or ld.elf_so(1), depending on the system's object file format.

SEE ALSO

ld(1), ld.aout_so(1), ld.elf_so(1), a.out(5), elf(5), link(5), ldconfig(8)

NAME

ldapcompare − LDAP compare tool

SYNOPSIS

ldapcompare [**−n**] [**−v**] [**−z**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] *DN* < **attr:value** | **attr::b64value** >

DESCRIPTION

ldapcompare is a shell-accessible interface to the **ldap_compare_ext**(3) library call.

ldapcompare opens a connection to an LDAP server, binds, and performs a compare using specified parameters. The *DN* should be a distinguished name in the directory. *Attr* should be a known attribute. If followed by one colon, the assertion *value* should be provided as a string. If followed by two colons, the base64 encoding of the value is provided. The result code of the compare is provided as the exit code and, unless ran with -z, the program prints TRUE, FALSE, or UNDEFINED on standard output.

OPTIONS

−n Show what would be done, but don't actually perform the compare. Useful for debugging in conjunction with -v.

−v Run in verbose mode, with many diagnostics written to standard output.

−z Run in quiet mode, no output is written. You must check the return status. Useful in shell scripts.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapcompare** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication. Note that *complete* means that any leading or trailing whitespaces, including newlines, will be considered part of the password and, unlike other software, they will not be stripped. As a consequence, passwords stored in files by commands like **echo**(1) will not behave as expected, since **echo**(1) by default appends a trailing newline to the echoed string. The recommended portable way to store a cleartext password in a file for use with this option is to use **slappasswd**(8) with *{CLEARTEXT}* as hash and the option *−n*.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

EXAMPLES

```
ldapcompare "uid=babs,dc=example,dc=com" sn:Jensen
ldapcompare "uid=babs,dc=example,dc=com" sn::SmVuc2Vu
```

are all equivalent.

LIMITATIONS

Requiring the value be passed on the command line is limiting and introduces some security concerns. The command should support a mechanism to specify the location (file name or URL) to read the value from.

SEE ALSO

ldap.conf(5), **ldif**(5), **ldap**(3), **ldap_compare_ext**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldapcompare − LDAP compare tool

SYNOPSIS

ldapcompare [**−n**] [**−v**] [**−z**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] *DN* < **attr:value** | **attr::b64value** >

DESCRIPTION

ldapcompare is a shell-accessible interface to the **ldap_compare_ext**(3) library call.

ldapcompare opens a connection to an LDAP server, binds, and performs a compare using specified parameters. The *DN* should be a distinguished name in the directory. *Attr* should be a known attribute. If followed by one colon, the assertion *value* should be provided as a string. If followed by two colons, the base64 encoding of the value is provided. The result code of the compare is provided as the exit code and, unless ran with -z, the program prints TRUE, FALSE, or UNDEFINED on standard output.

OPTIONS

−n Show what would be done, but don't actually perform the compare. Useful for debugging in conjunction with -v.

−v Run in verbose mode, with many diagnostics written to standard output.

−z Run in quiet mode, no output is written. You must check the return status. Useful in shell scripts.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapcompare** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication. Note that *complete* means that any leading or trailing whitespaces, including newlines, will be considered part of the password and, unlike other software, they will not be stripped. As a consequence, passwords stored in files by commands like **echo**(1) will not behave as expected, since **echo**(1) by default appends a trailing newline to the echoed string. The recommended portable way to store a cleartext password in a file for use with this option is to use **slappasswd**(8) with *{CLEARTEXT}* as hash and the option *−n*.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

EXAMPLES

```
ldapcompare "uid=babs,dc=example,dc=com" sn:Jensen
ldapcompare "uid=babs,dc=example,dc=com" sn::SmVuc2Vu
```

are all equivalent.

LIMITATIONS

Requiring the value be passed on the command line is limiting and introduces some security concerns. The command should support a mechanism to specify the location (file name or URL) to read the value from.

SEE ALSO

ldap.conf(5), **ldif**(5), **ldap**(3), **ldap_compare_ext**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldapdelete − LDAP delete entry tool

SYNOPSIS

ldapdelete [**−n**] [**−v**] [**−c**] [**−M[M]**] [**−d** *debuglevel*] [**−f** *file*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−P** *2|3*] [**−p** *ldapport*] [**−O** security-properties**]** [**−U** *authcid*] [**−R** *realm*] [**−r**] [**−x**] [**−I**] [**−Q**] [**−X** *authzid*] [**−Y** *mech*] [**−z** *sizelimit*] [**−Z[Z]**] [*dn*]...

DESCRIPTION

ldapdelete is a shell-accessible interface to the **ldap_delete_ext**(3) library call.

ldapdelete opens a connection to an LDAP server, binds, and deletes one or more entries. If one or more *DN* arguments are provided, entries with those Distinguished Names are deleted. Each *DN* should be provided using the LDAPv3 string representation as defined in RFC 4514. If no *dn* arguments are provided, a list of DNs is read from standard input (or from *file* if the -f flag is used).

OPTIONS

−n Show what would be done, but don't actually delete entries. Useful for debugging in conjunction with -v.

−v Use verbose mode, with many diagnostics written to standard output.

−c Continuous operation mode. Errors are reported, but **ldapdelete** will continue with deletions. The default is to exit after reporting an error.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapdelete** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f *file* Read a series of DNs from *file*, one per line, performing an LDAP delete for each.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−r Do a recursive delete. If the DN specified isn't a leaf, its children, and all their children are deleted down the tree. No verification is done, so if you add this switch, ldapdelete will happily delete large portions of your tree. Use with care.

−z *sizelimit*
Use *sizelimit* when searching for children DN to delete, to circumvent any server-side size limit. Only useful in conjunction with −r.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the identity depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

EXAMPLE

The following command:

```
ldapdelete "cn=Delete Me,dc=example,dc=com"
```

will attempt to delete the entry named "cn=Delete Me,dc=example,dc=com". Of course it would probably be necessary to supply authentication credentials.

DIAGNOSTICS

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldap.conf(5), **ldapadd**(1), **ldapmodify**(1), **ldapmodrdn**(1), **ldapsearch**(1), **ldap**(3), **ldap_delete_ext**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldapdelete − LDAP delete entry tool

SYNOPSIS

ldapdelete [**−n**] [**−v**] [**−c**] [**−M[M]**] [**−d** *debuglevel*] [**−f** *file*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−P** *2|3*] [**−p** *ldapport*] [**−O** security-properties**]** [**−U** *authcid*] [**−R** *realm*] [**−r**] [**−x**] [**−I**] [**−Q**] [**−X** *authzid*] [**−Y** *mech*] [**−z** *sizelimit*] [**−Z[Z]**] [*dn*]...

DESCRIPTION

ldapdelete is a shell-accessible interface to the **ldap_delete_ext**(3) library call.

ldapdelete opens a connection to an LDAP server, binds, and deletes one or more entries. If one or more *DN* arguments are provided, entries with those Distinguished Names are deleted. Each *DN* should be provided using the LDAPv3 string representation as defined in RFC 4514. If no *dn* arguments are provided, a list of DNs is read from standard input (or from *file* if the -f flag is used).

OPTIONS

−n Show what would be done, but don't actually delete entries. Useful for debugging in conjunction with -v.

−v Use verbose mode, with many diagnostics written to standard output.

−c Continuous operation mode. Errors are reported, but **ldapdelete** will continue with deletions. The default is to exit after reporting an error.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapdelete** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f *file* Read a series of DNs from *file*, one per line, performing an LDAP delete for each.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−r Do a recursive delete. If the DN specified isn't a leaf, its children, and all their children are deleted down the tree. No verification is done, so if you add this switch, ldapdelete will happily delete large portions of your tree. Use with care.

−z *sizelimit*
Use *sizelimit* when searching for children DN to delete, to circumvent any server-side size limit. Only useful in conjunction with −r.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the identity depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

EXAMPLE

The following command:

```
ldapdelete "cn=Delete Me,dc=example,dc=com"
```

will attempt to delete the entry named "cn=Delete Me,dc=example,dc=com". Of course it would probably be necessary to supply authentication credentials.

DIAGNOSTICS

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldap.conf(5), **ldapadd**(1), **ldapmodify**(1), **ldapmodrdn**(1), **ldapsearch**(1), **ldap**(3), **ldap_delete_ext**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldapmodify, ldapadd − LDAP modify entry and LDAP add entry tools

SYNOPSIS

ldapmodify [**−a**] [**−c**] [**−S** *file*] [**−n**] [**−v**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [**−f** *file*]

ldapadd [**−c**] [**−S** *file*] [**−n**] [**−v**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [**−f** *file*]

DESCRIPTION

ldapmodify is a shell-accessible interface to the **ldap_add_ext**(3), **ldap_modify_ext**(3), **ldap_delete_ext**(3) and **ldap_rename**(3). library calls. **ldapadd** is implemented as a hard link to the ldapmodify tool. When invoked as **ldapadd** the -a (add new entry) flag is turned on automatically.

ldapmodify opens a connection to an LDAP server, binds, and modifies or adds entries. The entry information is read from standard input or from *file* through the use of the -f option.

OPTIONS

−a Add new entries. The default for **ldapmodify** is to modify existing entries. If invoked as **ldapadd**, this flag is always set.

−c Continuous operation mode. Errors are reported, but **ldapmodify** will continue with modifications. The default is to exit after reporting an error.

−S *file* Add or change records which where skipped due to an error are written to *file* and the error message returned by the server is added as a comment. Most useful in conjunction with -c.

−n Show what would be done, but don't actually modify entries. Useful for debugging in conjunction with -v.

−v Use verbose mode, with many diagnostics written to standard output.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapmodify** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f *file* Read the entry modification information from *file* instead of from standard input.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

INPUT FORMAT

The contents of *file* (or standard input if no −f flag is given on the command line) must conform to the format defined in **ldif**(5) (LDIF as defined in RFC 2849).

EXAMPLES

Assuming that the file **/tmp/entrymods** exists and has the contents:

```
dn: cn=Modify Me,dc=example,dc=com
changetype: modify
replace: mail
mail: modme@example.com
-
add: title
title: Grand Poobah
-
add: jpegPhoto
jpegPhoto:< file:///tmp/modme.jpeg
-
delete: description
-
```

the command:

```
ldapmodify -f /tmp/entrymods
```

will replace the contents of the "Modify Me" entry's *mail* attribute with the value "modme@example.com", add a *title* of "Grand Poobah", and the contents of the file "/tmp/modme.jpeg" as a *jpegPhoto*, and completely remove the *description* attribute.

Assuming that the file **/tmp/newentry** exists and has the contents:

```
dn: cn=Barbara Jensen,dc=example,dc=com
objectClass: person
cn: Barbara Jensen
```

```
cn: Babs Jensen
sn: Jensen
title: the world's most famous mythical manager
mail: bjensen@example.com
uid: bjensen
```

the command:

```
ldapadd -f /tmp/newentry
```

will add a new entry for Babs Jensen, using the values from the file **/tmp/newentry.**

Assuming that the file **/tmp/entrymods** exists and has the contents:

```
dn: cn=Barbara Jensen,dc=example,dc=com
changetype: delete
```

the command:

```
ldapmodify -f /tmp/entrymods
```

will remove Babs Jensen's entry.

DIAGNOSTICS

Exit status is zero if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldapadd(1), **ldapdelete**(1), **ldapmodrdn**(1), **ldapsearch**(1), **ldap.conf**(5), **ldap**(3), **ldap_add_ext**(3), **ldap_delete_ext**(3), **ldap_modify_ext**(3), **ldap_modrdn_ext**(3), **ldif**(5), **slapd.replog**(5)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldapmodify, ldapadd − LDAP modify entry and LDAP add entry tools

SYNOPSIS

ldapmodify [**−a**] [**−c**] [**−S** *file*] [**−n**] [**−v**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [**−f** *file*]

ldapadd [**−c**] [**−S** *file*] [**−n**] [**−v**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [**−f** *file*]

DESCRIPTION

ldapmodify is a shell-accessible interface to the **ldap_add_ext**(3), **ldap_modify_ext**(3), **ldap_delete_ext**(3) and **ldap_rename**(3). library calls. **ldapadd** is implemented as a hard link to the ldapmodify tool. When invoked as **ldapadd** the -a (add new entry) flag is turned on automatically.

ldapmodify opens a connection to an LDAP server, binds, and modifies or adds entries. The entry information is read from standard input or from *file* through the use of the -f option.

OPTIONS

−a Add new entries. The default for **ldapmodify** is to modify existing entries. If invoked as **ldapadd**, this flag is always set.

−c Continuous operation mode. Errors are reported, but **ldapmodify** will continue with modifications. The default is to exit after reporting an error.

−S *file* Add or change records which where skipped due to an error are written to *file* and the error message returned by the server is added as a comment. Most useful in conjunction with -c.

−n Show what would be done, but don't actually modify entries. Useful for debugging in conjunction with -v.

−v Use verbose mode, with many diagnostics written to standard output.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapmodify** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f *file* Read the entry modification information from *file* instead of from standard input.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

INPUT FORMAT

The contents of *file* (or standard input if no −f flag is given on the command line) must conform to the format defined in **ldif**(5) (LDIF as defined in RFC 2849).

EXAMPLES

Assuming that the file **/tmp/entrymods** exists and has the contents:

```
dn: cn=Modify Me,dc=example,dc=com
changetype: modify
replace: mail
mail: modme@example.com
-
add: title
title: Grand Poobah
-
add: jpegPhoto
jpegPhoto:< file:///tmp/modme.jpeg
-
delete: description
-
```

the command:

```
ldapmodify -f /tmp/entrymods
```

will replace the contents of the "Modify Me" entry's *mail* attribute with the value "modme@example.com", add a *title* of "Grand Poobah", and the contents of the file "/tmp/modme.jpeg" as a *jpegPhoto*, and completely remove the *description* attribute.

Assuming that the file **/tmp/newentry** exists and has the contents:

```
dn: cn=Barbara Jensen,dc=example,dc=com
objectClass: person
cn: Barbara Jensen
```

cn: Babs Jensen
sn: Jensen
title: the world's most famous mythical manager
mail: bjensen@example.com
uid: bjensen

the command:

ldapadd -f /tmp/newentry

will add a new entry for Babs Jensen, using the values from the file **/tmp/newentry.**

Assuming that the file **/tmp/entrymods** exists and has the contents:

dn: cn=Barbara Jensen,dc=example,dc=com
changetype: delete

the command:

ldapmodify -f /tmp/entrymods

will remove Babs Jensen's entry.

DIAGNOSTICS

Exit status is zero if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldapadd(1), **ldapdelete**(1), **ldapmodrdn**(1), **ldapsearch**(1), **ldap.conf**(5), **ldap**(3), **ldap_add_ext**(3), **ldap_delete_ext**(3), **ldap_modify_ext**(3), **ldap_modrdn_ext**(3), **ldif**(5), **slapd.replog**(5)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldapmodrdn − LDAP rename entry tool

SYNOPSIS

ldapmodrdn [**−r**] [**−s** *newsup*] [**−n**] [**−v**] [**−c**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [**−f** *file*] [*dn rdn*]

DESCRIPTION

ldapmodrdn is a shell-accessible interface to the **ldap_rename**(3) library call.

ldapmodrdn opens a connection to an LDAP server, binds, and modifies the RDN of entries. The entry information is read from standard input, from *file* through the use of the *−f* option, or from the command-line pair *dn* and *rdn*.

OPTIONS

−r Remove old RDN values from the entry. Default is to keep old values.

−s *newsup*
Specify a new superior entry. (I.e., move the target entry and make it a child of the new superior.) This option is not supported in LDAPv2.

−n Show what would be done, but don't actually change entries. Useful for debugging in conjunction with -v.

−v Use verbose mode, with many diagnostics written to standard output.

−c Continuous operation mode. Errors are reported, but ldapmodrdn will continue with modifications. The default is to exit after reporting an error.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d debuglevel
Set the LDAP debugging level to *debuglevel*. **ldapmodrdn** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f file Read the entry modification information from *file* instead of from standard input or the command-line.

−x Use simple authentication instead of SASL.

−D binddn
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w passwd
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

INPUT FORMAT

If the command-line arguments *dn* and *rdn* are given, *rdn* will replace the RDN of the entry specified by the DN, *dn*.

Otherwise, the contents of *file* (or standard input if no *−f* flag is given) should consist of one or more entries.

Distinguished Name (DN)
Relative Distinguished Name (RDN)

One or more blank lines may be used to separate each DN/RDN pair.

EXAMPLE

Assuming that the file **/tmp/entrymods** exists and has the contents:

```
cn=Modify Me,dc=example,dc=com
cn=The New Me
```

the command:

```
ldapmodrdn -r -f /tmp/entrymods
```

will change the RDN of the "Modify Me" entry from "Modify Me" to "The New Me" and the old cn, "Modify Me" will be removed.

DIAGNOSTICS

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldapadd(1), **ldapdelete**(1), **ldapmodify**(1), **ldapsearch**(1), **ldap.conf**(5), **ldap**(3), **ldap_rename**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldapmodrdn − LDAP rename entry tool

SYNOPSIS

ldapmodrdn [**−r**] [**−s** *newsup*] [**−n**] [**−v**] [**−c**] [**−M[M]**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−P** *2|3*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [**−f** *file*] [*dn rdn*]

DESCRIPTION

ldapmodrdn is a shell-accessible interface to the **ldap_rename**(3) library call.

ldapmodrdn opens a connection to an LDAP server, binds, and modifies the RDN of entries. The entry information is read from standard input, from *file* through the use of the *−f* option, or from the command-line pair *dn* and *rdn*.

OPTIONS

−r Remove old RDN values from the entry. Default is to keep old values.

−s *newsup*
Specify a new superior entry. (I.e., move the target entry and make it a child of the new superior.) This option is not supported in LDAPv2.

−n Show what would be done, but don't actually change entries. Useful for debugging in conjunction with -v.

−v Use verbose mode, with many diagnostics written to standard output.

−c Continuous operation mode. Errors are reported, but ldapmodrdn will continue with modifications. The default is to exit after reporting an error.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−d debuglevel
Set the LDAP debugging level to *debuglevel*. **ldapmodrdn** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f file Read the entry modification information from *file* instead of from standard input or the command-line.

−x Use simple authentication instead of SASL.

−D binddn
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w passwd
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

INPUT FORMAT

If the command-line arguments *dn* and *rdn* are given, *rdn* will replace the RDN of the entry specified by the DN, *dn*.

Otherwise, the contents of *file* (or standard input if no *−f* flag is given) should consist of one or more entries.

Distinguished Name (DN)
Relative Distinguished Name (RDN)

One or more blank lines may be used to separate each DN/RDN pair.

EXAMPLE

Assuming that the file **/tmp/entrymods** exists and has the contents:

```
cn=Modify Me,dc=example,dc=com
cn=The New Me
```

the command:

```
ldapmodrdn -r -f /tmp/entrymods
```

will change the RDN of the "Modify Me" entry from "Modify Me" to "The New Me" and the old cn, "Modify Me" will be removed.

DIAGNOSTICS

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldapadd(1), **ldapdelete**(1), **ldapmodify**(1), **ldapsearch**(1), **ldap.conf**(5), **ldap**(3), **ldap_rename**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldappasswd − change the password of an LDAP entry

SYNOPSIS

ldappasswd [**−A**] [**−a** *oldPasswd*] [**−t** *oldpasswdfile*] [**−D** *binddn*] [**−d** *debuglevel*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−n**] [**−p** *ldapport*] [**−S**] [**−s** *newPasswd*] [**−T** *newpasswdfile*] [**−v**] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [*user*]

DESCRIPTION

ldappasswd is a tool to set the password of an LDAP user. **ldappasswd** uses the LDAPv3 Password Modify (RFC 3062) extended operation.

ldappasswd sets the password of associated with the user [or an optionally specified *user*]. If the new password is not specified on the command line and the user doesn't enable prompting, the server will be asked to generate a password for the user.

ldappasswd is neither designed nor intended to be a replacement for **passwd**(1) and should not be installed as such.

OPTIONS

−A Prompt for old password. This is used instead of specifying the password on the command line.

−a *oldPasswd*
Set the old password to *oldPasswd*.

−t *oldPasswdFile*
Set the old password to the contents of *oldPasswdFile*.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldappasswd** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−n Do not set password. (Can be useful when used in conjunction with **−v** or **−d**)

−S Prompt for new password. This is used instead of specifying the password on the command line.

−s *newPasswd*
Set the new password to *newPasswd*.

−T *newPasswdFile*
Set the new password to the contents of *newPasswdFile*.

−v Increase the verbosity of output. Can be specified multiple times.

−W Prompt for bind password. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password to bind with.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished***name>** or **u:**<*username*>**.**

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful

SEE ALSO

ldap_sasl_bind(3), **ldap_extended_operation**(3), **ldap_start_tls_s**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldappasswd − change the password of an LDAP entry

SYNOPSIS

ldappasswd [**−A**] [**−a** *oldPasswd*] [**−t** *oldpasswdfile*] [**−D** *binddn*] [**−d** *debuglevel*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−n**] [**−p** *ldapport*] [**−S**] [**−s** *newPasswd*] [**−T** *newpasswdfile*] [**−v**] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] [*user*]

DESCRIPTION

ldappasswd is a tool to set the password of an LDAP user. **ldappasswd** uses the LDAPv3 Password Modify (RFC 3062) extended operation.

ldappasswd sets the password of associated with the user [or an optionally specified *user*]. If the new password is not specified on the command line and the user doesn't enable prompting, the server will be asked to generate a password for the user.

ldappasswd is neither designed nor intended to be a replacement for **passwd**(1) and should not be installed as such.

OPTIONS

−A Prompt for old password. This is used instead of specifying the password on the command line.

−a *oldPasswd*
Set the old password to *oldPasswd*.

−t *oldPasswdFile*
Set the old password to the contents of *oldPasswdFile*.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldappasswd** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−n Do not set password. (Can be useful when used in conjunction with **−v** or **−d**)

−S Prompt for new password. This is used instead of specifying the password on the command line.

−s *newPasswd*
Set the new password to *newPasswd*.

−T *newPasswdFile*
Set the new password to the contents of *newPasswdFile*.

−v Increase the verbosity of output. Can be specified multiple times.

−W Prompt for bind password. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password to bind with.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished***name>** or **u:**<*username*>**.**

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful

SEE ALSO

ldap_sasl_bind(3), **ldap_extended_operation**(3), **ldap_start_tls_s**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldapsearch − LDAP search tool

SYNOPSIS

ldapsearch [**−n**] [**−c**] [**−u**] [**−v**] [**−t[t]**] [**−T** *path*] [**−F** *prefix*] [**−A**] [**−L[L[L]]**] [**−M[M]**] [**−S** *attribute*] [**−d** *debuglevel*] [**−f** *file*] [**−x**] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−b** *searchbase*] [**−s** *base*|*one*|*sub*|*children*] [**−a** *never*|*always*|*search*|*find*] [**−P** *2*|*3*] [**−e** [!]ext[=extparam]] [**−E** [!]ext[=extparam]] [**−l** *timelimit*] [**−z** *sizelimit*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] *filter* [*attrs...*]

DESCRIPTION

ldapsearch is a shell-accessible interface to the **ldap_search_ext**(3) library call.

ldapsearch opens a connection to an LDAP server, binds, and performs a search using specified parameters. The *filter* should conform to the string representation for search filters as defined in RFC 4515. If not provided, the default filter, (objectClass=*), is used.

If **ldapsearch** finds one or more entries, the attributes specified by *attrs* are returned. If * is listed, all user attributes are returned. If + is listed, all operational attributes are returned. If no *attrs* are listed, all user attributes are returned. If only 1.1 is listed, no attributes will be returned.

OPTIONS

−n Show what would be done, but don't actually perform the search. Useful for debugging in conjunction with -v.

−c Continuous operation mode. Errors are reported, but ldapsearch will continue with searches. The default is to exit after reporting an error. Only useful in conjunction with -f.

−u Include the User Friendly Name form of the Distinguished Name (DN) in the output.

−v Run in verbose mode, with many diagnostics written to standard output.

−t[t] A single -t writes retrieved non-printable values to a set of temporary files. This is useful for dealing with values containing non-character data such as jpegPhoto or audio. A second -t writes all retrieved values to files.

−T *path*
Write temporary files to directory specified by *path* (default: /var/tmp/)

−F *prefix*
URL prefix for temporary files. Default is file://*path*/ where *path* is /var/tmp/ or specified with -T.

−A Retrieve attributes only (no values). This is useful when you just want to see if an attribute is present in an entry and are not interested in the specific values.

−L Search results are display in LDAP Data Interchange Format detailed in **ldif**(5). A single -L restricts the output to LDIFv1. A second -L disables comments. A third -L disables printing of the LDIF version. The default is to use an extended version of LDIF.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−S *attribute*
Sort the entries returned based on *attribute*. The default is not to sort entries returned. If *attribute* is a zero-length string (""), the entries are sorted by the components of their Distinguished Name. See **ldap_sort**(3) for more details. Note that **ldapsearch** normally prints out entries as it receives them. The use of the **−S** option defeats this behavior, causing all entries to be retrieved, then sorted, then printed.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapsearch** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f *file* Read a series of lines from *file*, performing one LDAP search for each line. In this case, the *filter* given on the command line is treated as a pattern where the first and only occurrence of **%s** is

replaced with a line from *file*. Any other occurrence of the the **%** character in the pattern will be regarded as an error. Where it is desired that the search filter include a **%** character, the character should be encoded as **\25** (see RFC 4515). If *file* is a single - character, then the lines are read from standard input. **ldapsearch** will exit when the first non-successful search result is returned, unless -c is used.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); a list of URI, separated by whitespace or commas is expected; only the protocol/host/port fields are allowed. As an exception, if no host/port is specified, but a DN is, the DN is used to look up the corresponding host(s) using the DNS SRV records, according to RFC 2782. The DN must be a non-empty sequence of AVAs whose attribute type is "dc" (domain component), and must be escaped according to RFC 2396.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−b *searchbase*
Use *searchbase* as the starting point for the search instead of the default.

−s *base*|*one*|*sub*|*children*
Specify the scope of the search to be one of *base*, *one*, *sub*, or *children* to specify a base object, one-level, subtree, or children search. The default is *sub*. Note: *children* scope requires LDAPv3 subordinate feature extension.

−a *never*|*always*|*search*|*find*
Specify how aliases dereferencing is done. Should be one of *never*, *always*, *search*, or *find* to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search. The default is to never dereference aliases.

−P *2*|*3* Specify the LDAP protocol version to use.

−e *[!]ext[=extparam]*

−E *[!]ext[=extparam]*

Specify general extensions with -e and search extensions with -E. ´!´ indicates criticality.

General extensions:
[!]assert=<filter> (an RFC 4515 Filter)
[!]authzid=<authzid> ("dn:<dn>" or "u:<user>")
[!]manageDSAit
[!]noop
ppolicy
[!]postread[=<attrs>] (a comma-separated attribute list)
[!]preread[=<attrs>] (a comma-separated attribute list)

abandon, cancel (SIGINT sends abandon/cancel; not really controls)

Search extensions:
[!]domainScope (domain scope)
[!]mv=<filter> (matched values filter)
[!]pr=<size>[/prompt|noprompt] (paged results/prompt)
[!]subentries[=true|false] (subentries)
[!]sync=ro[/<cookie>] (LDAP Sync refreshOnly)
rp[/<cookie>][/<slimit>] (LDAP Sync refreshAndPersist)

−l *timelimit*
wait at most *timelimit* seconds for a search to complete. A timelimit of *0* (zero) or *none* means no limit. A timelimit of *max* means the maximum integer allowable by the protocol. A server may impose a maximal timelimit which only the root user may override.

−z *sizelimit*
retrieve at most *sizelimit* entries for a search. A sizelimit of *0* (zero) or *none* means no limit. A sizelimit of *max* means the maximum integer allowable by the protocol. A server may impose a maximal sizelimit which only the root user may override.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

OUTPUT FORMAT

If one or more entries are found, each entry is written to standard output in LDAP Data Interchange Format or **ldif**(5):

```
version: 1

# bjensen, example, net
dn: uid=bjensen,dc=example,dc=net
objectClass: person
objectClass: dcObject
uid: bjensen
cn: Barbara Jensen
sn: Jensen
...
```

If the -t option is used, the URI of a temporary file is used in place of the actual value. If the -A option is

given, only the "attributename" part is written.

EXAMPLE

The following command:

```
ldapsearch -LLL "(sn=smith)" cn sn telephoneNumber
```

will perform a subtree search (using the default search base and other parameters defined in **ldap.conf**(5)) for entries with a surname (sn) of smith. The common name (cn), surname (sn) and telephoneNumber values will be retrieved and printed to standard output. The output might look something like this if two entries are found:

```
dn: uid=jts,dc=example,dc=com
cn: John Smith
cn: John T. Smith
sn: Smith
sn;lang-en: Smith
sn;lang-de: Schmidt
telephoneNumber: 1 555 123-4567

dn: uid=sss,dc=example,dc=com
cn: Steve Smith
cn: Steve S. Smith
sn: Smith
sn;lang-en: Smith
sn;lang-de: Schmidt
telephoneNumber: 1 555 765-4321
```

The command:

```
ldapsearch -LLL -u -t "(uid=xyz)" jpegPhoto audio
```

will perform a subtree search using the default search base for entries with user id of "xyz". The user friendly form of the entry's DN will be output after the line that contains the DN itself, and the jpegPhoto and audio values will be retrieved and written to temporary files. The output might look like this if one entry with one value for each of the requested attributes is found:

```
dn: uid=xyz,dc=example,dc=com
ufn: xyz, example, com
audio:< file:///tmp/ldapsearch-audio-a19924
jpegPhoto:< file:///tmp/ldapsearch-jpegPhoto-a19924
```

This command:

```
ldapsearch -LLL -s one -b "c=US" "(o=University*)" o description
```

will perform a one-level search at the c=US level for all entries whose organization name (o) begins begins with **University**. The organization name and description attribute values will be retrieved and printed to standard output, resulting in output similar to this:

```
dn: o=University of Alaska Fairbanks,c=US
o: University of Alaska Fairbanks
description: Preparing Alaska for a brave new yesterday
description: leaf node only

dn: o=University of Colorado at Boulder,c=US
o: University of Colorado at Boulder
description: No personnel information
description: Institution of education and research

dn: o=University of Colorado at Denver,c=US
o: University of Colorado at Denver
```

o: UCD
o: CU/Denver
o: CU-Denver
description: Institute for Higher Learning and Research

dn: o=University of Florida,c=US
o: University of Florida
o: UFl
description: Warper of young minds

...

DIAGNOSTICS

Exit status is zero if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldapadd(1), **ldapdelete**(1), **ldapmodify**(1), **ldapmodrdn**(1), **ldap.conf**(5), **ldif**(5), **ldap**(3), **ldap_search_ext**(3), **ldap_sort**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldapsearch − LDAP search tool

SYNOPSIS

ldapsearch [**−n**] [**−c**] [**−u**] [**−v**] [**−t[t]**] [**−T** *path*] [**−F** *prefix*] [**−A**] [**−L[L[L]]**] [**−M[M]**] [**−S** *attribute*] [**−d** *debuglevel*] [**−f** *file*] [**−x**] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−b** *searchbase*] [**−s** *base*|*one*|*sub*|*children*] [**−a** *never*|*always*|*search*|*find*] [**−P** *2*|*3*] [**−e** [!]ext[=extparam]] [**−E** [!]ext[=extparam]] [**−l** *timelimit*] [**−z** *sizelimit*] [**−O** security-properties**]** [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**] *filter* [*attrs...*]

DESCRIPTION

ldapsearch is a shell-accessible interface to the **ldap_search_ext**(3) library call.

ldapsearch opens a connection to an LDAP server, binds, and performs a search using specified parameters. The *filter* should conform to the string representation for search filters as defined in RFC 4515. If not provided, the default filter, (objectClass=*), is used.

If **ldapsearch** finds one or more entries, the attributes specified by *attrs* are returned. If * is listed, all user attributes are returned. If + is listed, all operational attributes are returned. If no *attrs* are listed, all user attributes are returned. If only 1.1 is listed, no attributes will be returned.

OPTIONS

−n Show what would be done, but don't actually perform the search. Useful for debugging in conjunction with -v.

−c Continuous operation mode. Errors are reported, but ldapsearch will continue with searches. The default is to exit after reporting an error. Only useful in conjunction with -f.

−u Include the User Friendly Name form of the Distinguished Name (DN) in the output.

−v Run in verbose mode, with many diagnostics written to standard output.

−t[t] A single -t writes retrieved non-printable values to a set of temporary files. This is useful for dealing with values containing non-character data such as jpegPhoto or audio. A second -t writes all retrieved values to files.

−T *path*
Write temporary files to directory specified by *path* (default: /var/tmp/)

−F *prefix*
URL prefix for temporary files. Default is file://*path*/ where *path* is /var/tmp/ or specified with -T.

−A Retrieve attributes only (no values). This is useful when you just want to see if an attribute is present in an entry and are not interested in the specific values.

−L Search results are display in LDAP Data Interchange Format detailed in **ldif**(5). A single -L restricts the output to LDIFv1. A second -L disables comments. A third -L disables printing of the LDIF version. The default is to use an extended version of LDIF.

−M[M]
Enable manage DSA IT control. **−MM** makes control critical.

−S *attribute*
Sort the entries returned based on *attribute*. The default is not to sort entries returned. If *attribute* is a zero-length string (""), the entries are sorted by the components of their Distinguished Name. See **ldap_sort**(3) for more details. Note that **ldapsearch** normally prints out entries as it receives them. The use of the **−S** option defeats this behavior, causing all entries to be retrieved, then sorted, then printed.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapsearch** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−f *file* Read a series of lines from *file*, performing one LDAP search for each line. In this case, the *filter* given on the command line is treated as a pattern where the first and only occurrence of **%s** is

replaced with a line from *file*. Any other occurrence of the the **%** character in the pattern will be regarded as an error. Where it is desired that the search filter include a **%** character, the character should be encoded as **\25** (see RFC 4515). If *file* is a single - character, then the lines are read from standard input. **ldapsearch** will exit when the first non-successful search result is returned, unless -c is used.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); a list of URI, separated by whitespace or commas is expected; only the protocol/host/port fields are allowed. As an exception, if no host/port is specified, but a DN is, the DN is used to look up the corresponding host(s) using the DNS SRV records, according to RFC 2782. The DN must be a non-empty sequence of AVAs whose attribute type is "dc" (domain component), and must be escaped according to RFC 2396.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−b *searchbase*
Use *searchbase* as the starting point for the search instead of the default.

−s *base*|*one*|*sub*|*children*
Specify the scope of the search to be one of *base*, *one*, *sub*, or *children* to specify a base object, one-level, subtree, or children search. The default is *sub*. Note: *children* scope requires LDAPv3 subordinate feature extension.

−a *never*|*always*|*search*|*find*
Specify how aliases dereferencing is done. Should be one of *never*, *always*, *search*, or *find* to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search. The default is to never dereference aliases.

−P *2*|*3* Specify the LDAP protocol version to use.

−e *[!]ext[=extparam]*

−E *[!]ext[=extparam]*

Specify general extensions with -e and search extensions with -E. ´!´ indicates criticality.

General extensions:
[!]assert=<filter> (an RFC 4515 Filter)
[!]authzid=<authzid> ("dn:<dn>" or "u:<user>")
[!]manageDSAit
[!]noop
ppolicy
[!]postread[=<attrs>] (a comma-separated attribute list)
[!]preread[=<attrs>] (a comma-separated attribute list)

abandon, cancel (SIGINT sends abandon/cancel; not really controls)

Search extensions:
[!]domainScope (domain scope)
[!]mv=<filter> (matched values filter)
[!]pr=<size>[/prompt|noprompt] (paged results/prompt)
[!]subentries[=true|false] (subentries)
[!]sync=ro[/<cookie>] (LDAP Sync refreshOnly)
rp[/<cookie>][/<slimit>] (LDAP Sync refreshAndPersist)

−l *timelimit*
wait at most *timelimit* seconds for a search to complete. A timelimit of *0* (zero) or *none* means no limit. A timelimit of *max* means the maximum integer allowable by the protocol. A server may impose a maximal timelimit which only the root user may override.

−z *sizelimit*
retrieve at most *sizelimit* entries for a search. A sizelimit of *0* (zero) or *none* means no limit. A sizelimit of *max* means the maximum integer allowable by the protocol. A server may impose a maximal sizelimit which only the root user may override.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

OUTPUT FORMAT

If one or more entries are found, each entry is written to standard output in LDAP Data Interchange Format or **ldif**(5):

```
version: 1

# bjensen, example, net
dn: uid=bjensen,dc=example,dc=net
objectClass: person
objectClass: dcObject
uid: bjensen
cn: Barbara Jensen
sn: Jensen
...
```

If the -t option is used, the URI of a temporary file is used in place of the actual value. If the -A option is

given, only the "attributename" part is written.

EXAMPLE

The following command:

```
ldapsearch -LLL "(sn=smith)" cn sn telephoneNumber
```

will perform a subtree search (using the default search base and other parameters defined in **ldap.conf**(5)) for entries with a surname (sn) of smith. The common name (cn), surname (sn) and telephoneNumber values will be retrieved and printed to standard output. The output might look something like this if two entries are found:

```
dn: uid=jts,dc=example,dc=com
cn: John Smith
cn: John T. Smith
sn: Smith
sn;lang-en: Smith
sn;lang-de: Schmidt
telephoneNumber: 1 555 123-4567

dn: uid=sss,dc=example,dc=com
cn: Steve Smith
cn: Steve S. Smith
sn: Smith
sn;lang-en: Smith
sn;lang-de: Schmidt
telephoneNumber: 1 555 765-4321
```

The command:

```
ldapsearch -LLL -u -t "(uid=xyz)" jpegPhoto audio
```

will perform a subtree search using the default search base for entries with user id of "xyz". The user friendly form of the entry's DN will be output after the line that contains the DN itself, and the jpegPhoto and audio values will be retrieved and written to temporary files. The output might look like this if one entry with one value for each of the requested attributes is found:

```
dn: uid=xyz,dc=example,dc=com
ufn: xyz, example, com
audio:< file:///tmp/ldapsearch-audio-a19924
jpegPhoto:< file:///tmp/ldapsearch-jpegPhoto-a19924
```

This command:

```
ldapsearch -LLL -s one -b "c=US" "(o=University*)" o description
```

will perform a one-level search at the c=US level for all entries whose organization name (o) begins begins with **University**. The organization name and description attribute values will be retrieved and printed to standard output, resulting in output similar to this:

```
dn: o=University of Alaska Fairbanks,c=US
o: University of Alaska Fairbanks
description: Preparing Alaska for a brave new yesterday
description: leaf node only

dn: o=University of Colorado at Boulder,c=US
o: University of Colorado at Boulder
description: No personnel information
description: Institution of education and research

dn: o=University of Colorado at Denver,c=US
o: University of Colorado at Denver
```

o: UCD
o: CU/Denver
o: CU-Denver
description: Institute for Higher Learning and Research

dn: o=University of Florida,c=US
o: University of Florida
o: UFl
description: Warper of young minds

...

DIAGNOSTICS

Exit status is zero if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.

SEE ALSO

ldapadd(1), **ldapdelete**(1), **ldapmodify**(1), **ldapmodrdn**(1), **ldap.conf**(5), **ldif**(5), **ldap**(3), **ldap_search_ext**(3), **ldap_sort**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldapwhoami − LDAP who am i? tool

SYNOPSIS

ldapwhoami [**−n**] [**−v**] [**−z**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−O** *security-properties*] [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**]

DESCRIPTION

ldapwhoami implements the LDAP "Who Am I?" extended operation.

ldapwhoami opens a connection to an LDAP server, binds, and performs a whoami operation.

OPTIONS

−n Show what would be done, but don't actually perform the whoami operation. Useful for debugging in conjunction with -v.

−v Run in verbose mode, with many diagnostics written to standard output.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapwhoami** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it’s not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

EXAMPLE

ldapwhoami -x -D "cn=Manager,dc=example,dc=com" -W

SEE ALSO

ldap.conf(5), **ldap**(3), **ldap_extended_operation**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

NAME

ldapwhoami − LDAP who am i? tool

SYNOPSIS

ldapwhoami [**−n**] [**−v**] [**−z**] [**−d** *debuglevel*] [**−D** *binddn*] [**−W**] [**−w** *passwd*] [**−y** *passwdfile*] [**−H** *ldapuri*] [**−h** *ldaphost*] [**−p** *ldapport*] [**−O** *security-properties*] [**−I**] [**−Q**] [**−U** *authcid*] [**−R** *realm*] [**−x**] [**−X** *authzid*] [**−Y** *mech*] [**−Z[Z]**]

DESCRIPTION

ldapwhoami implements the LDAP "Who Am I?" extended operation.

ldapwhoami opens a connection to an LDAP server, binds, and performs a whoami operation.

OPTIONS

−n Show what would be done, but don't actually perform the whoami operation. Useful for debugging in conjunction with -v.

−v Run in verbose mode, with many diagnostics written to standard output.

−d *debuglevel*
Set the LDAP debugging level to *debuglevel*. **ldapwhoami** must be compiled with LDAP_DEBUG defined for this option to have any effect.

−x Use simple authentication instead of SASL.

−D *binddn*
Use the Distinguished Name *binddn* to bind to the LDAP directory.

−W Prompt for simple authentication. This is used instead of specifying the password on the command line.

−w *passwd*
Use *passwd* as the password for simple authentication.

−y *passwdfile*
Use complete contents of *passwdfile* as the password for simple authentication.

−H *ldapuri*
Specify URI(s) referring to the ldap server(s); only the protocol/host/port fields are allowed; a list of URI, separated by whitespace or commas is expected.

−h *ldaphost*
Specify an alternate host on which the ldap server is running. Deprecated in favor of -H.

−p *ldapport*
Specify an alternate TCP port where the ldap server is listening. Deprecated in favor of -H.

−P *2|3* Specify the LDAP protocol version to use.

−O *security-properties*
Specify SASL security properties.

−I Enable SASL Interactive mode. Always prompt. Default is to prompt only as needed.

−Q Enable SASL Quiet mode. Never prompt.

−U *authcid*
Specify the authentication ID for SASL bind. The form of the ID depends on the actual SASL mechanism used.

−R *realm*
Specify the realm of authentication ID for SASL bind. The form of the realm depends on the actual SASL mechanism used.

−X *authzid*
Specify the requested authorization ID for SASL bind. *authzid* must be one of the following formats: **dn:**<*distinguished name*> or **u:**<*username*>

−Y *mech*
Specify the SASL mechanism to be used for authentication. If it's not specified, the program will choose the best mechanism the server knows.

−Z[Z] Issue StartTLS (Transport Layer Security) extended operation. If you use **−ZZ**, the command will require the operation to be successful.

EXAMPLE

ldapwhoami -x -D "cn=Manager,dc=example,dc=com" -W

SEE ALSO

ldap.conf(5), **ldap**(3), **ldap_extended_operation**(3)

AUTHOR

The OpenLDAP Project <http://www.openldap.org/>

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from University of Michigan LDAP 3.3 Release.

NAME

ldd — list dynamic object dependencies

SYNOPSIS

ldd [**-f** *format*] *program ...*

DESCRIPTION

ldd displays all shared objects that are needed to run the given program. Contrary to nm(1), the list includes "indirect" dependencies that are the result of needed shared objects which themselves depend on yet other shared objects. Zero, one or two **-f** options may be given. The argument is a format string passed to rtld(1) and allows customization of **ldd**'s output. The first format argument is used for library objects and is equivalent to the ld.aout_so(1) environment variable LD_TRACE_LOADED_OBJECTS_FMT1. It defaults to "\t-l%o.%m => %p\n" for elf and "\t-l%o.%m => %p (%x)\n" for a.out. The second format argument is used for non-library objects and it is equivalent to the ld.aout_so(1) environment variable LD_TRACE_LOADED_OBJECTS_FMT2. It defaults to "\t%o => %p\n" for elf and "\t%o (%x)\n" for a.out.

These arguments are interpreted as format strings a la printf(3) to customize the trace output and allow **ldd** to be operated as a filter more conveniently. The following conversions can be used:

%a The main program's name (also known as "__progname").

%A The value of the environment variable LD_TRACE_LOADED_OBJECTS_PROGNAME in a.out and the program name from the argument vector from elf.

%o The library name.

%m The library's major version number.

%n The library's minor version number (a.out only, ignored in elf).

%p The full pathname as determined by **rtld**'s library search rules.

%x The library's load address

Additionally, **\n** and **\t** are recognized and have their usual meaning.

SEE ALSO

ld(1), ld.aout_so(1), ld.elf_so(1), nm(1), rtld(1)

HISTORY

A **ldd** utility first appeared in SunOS 4.0, it appeared in its current form in NetBSD 0.9A.

BUGS

The a.out **ldd** actually runs the program it has been requested to analyze which in specially constructed environments can have security implications.

NAME

leave — remind you when you have to leave

SYNOPSIS

leave [[**+**]*hhmm*]

DESCRIPTION

leave waits until the specified time (within the next 12 hours), then reminds you that you have to leave by writing to the TTY that you executed **leave** on.

You are reminded 5 minutes and 1 minute before the actual time, at the time, and every minute thereafter. When you log off, **leave** exits just before it would have printed the next message.

OPTIONS

hhmm The time of day is in the form *hhmm* where *hh* is a time in hours (on a 12 or 24 hour clock), and *mm* are minutes.

However, all times are converted to a 12 hour clock, and assumed to be in the next 12 hours. An attempt to set an alarm for farther into the future will be truncated to within the next 12 hours.

+ If the time is preceded by **+**, the alarm will go off in hours and minutes from the current time.

If no argument is given, **leave** prompts with "When do you have to leave?". A reply of newline causes **leave** to exit, otherwise the reply is assumed to be a time. This form is suitable for inclusion in a `~/.login` or `~/.profile`.

SEE ALSO

`calendar`(1), `csh`(1), `sh`(1)

HISTORY

The **leave** command appeared in 3.0 BSD.

BUGS

In the modern age with `X`(7) and window multiplexing programs like `window`(1) and `screen`(1), the **leave** command's reminders and admonitions might not be seen if the user has the window where **leave** was started minimized or obscured.

This all begs for a more general user notifications system to be implemented.

NAME

less − opposite of more (a filter for browsing text files)

SYNOPSIS

less −?
less −−help
less −V
less −−version
less [−[+]aBcCdeEfFgGiIJKLmMnNqQrRsSuUVwWXˆ]
[−b *space***] [−h** *lines***] [−j** *line***] [−k** *keyfile***]**
[−{oO} *logfile***] [−p** *pattern***] [−P** *prompt***] [−t** *tag***]**
[−T *tagsfile***] [−x** *tab***,...] [−y** *lines***] [−[z]** *lines***]**
[−# *shift***] [+[+]**cmd**] [−−] [**filename**]...**
more [options]
page [options]
(See the OPTIONS section for alternate option syntax with long option names.)

DESCRIPTION

Less is a program similar to *more* (1), a filter that shows text one page at a time. In NetBSD the two utilities are implemented by the same binary but they expose slightly different behavior. Their differences are detailed below:

- *less* allows backward movement in the file as well as forward movement.
- *less* does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like *vi* (1).
- *more* ignores * and @ in search commands.
- *less* does not clear the screen or squish it to the bottom, for the first page, or when exiting.
- *more* sets the -E -m -G -f flags automatically.
- *more* uses $MORE instead of $LESS for extra options.

Less uses termcap (or terminfo on some systems), so it can run on a variety of terminals. There is even limited support for hardcopy terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are prefixed with a caret.)

Commands are based on both *more* and *vi*. Commands may be preceded by a decimal number, called N in the descriptions below. The number is used by some commands, as indicated.

COMMANDS

In the following descriptions, ˆX means control-X. ESC stands for the ESCAPE key; for example ESC-v means the two character sequence "ESCAPE", then "v".

h or H Help: display a summary of these commands. If you forget all the other commands, remember this one.

SPACE or ˆV or f or ˆF
Scroll forward N lines, default one window (see option −z below). If N is more than the screen size, only the final screenful is displayed. Warning: some systems use ˆV as a special literalization character.

z Like SPACE, but if N is specified, it becomes the new window size.

ESC-SPACE
Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the process.

RETURN or ˆN or e or ˆE or j or ˆJ
Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

d or ^D Scroll forward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

b or ^B or ESC-v
Scroll backward N lines, default one window (see option −z below). If N is more than the screen size, only the final screenful is displayed.

w Like ESC-v, but if N is specified, it becomes the new window size.

y or ^Y or ^P or k or ^K
Scroll backward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size. Warning: some systems use ^Y as a special job control character.

u or ^U Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

ESC-) or RIGHTARROW
Scroll horizontally right N characters, default half the screen width (see the −# option). If a number N is specified, it becomes the default for future RIGHTARROW and LEFTARROW commands. While the text is scrolled, it acts as though the −S option (chop lines) were in effect.

ESC-(or LEFTARROW
Scroll horizontally left N characters, default half the screen width (see the −# option). If a number N is specified, it becomes the default for future RIGHTARROW and LEFTARROW commands.

r or ^R or ^L
Repaint the screen.

R Repaint the screen, discarding any buffered input. Useful if the file is changing while it is being viewed.

F Scroll forward, and keep trying to read when the end of file is reached. Normally this command would be used when already at the end of the file. It is a way to monitor the tail of a file which is growing while it is being viewed. (The behavior is similar to the "tail −f" command.)

g or < or ESC-<
Go to line N in the file, default 1 (beginning of file). (Warning: this may be slow if N is large.)

G or > or ESC->
Go to line N in the file, default the end of the file. (Warning: this may be slow if N is large, or if N is not specified and standard input, rather than a file, is being read.)

p or % Go to a position N percent into the file. N should be between 0 and 100.

{ If a left curly bracket appears in the top line displayed on the screen, the { command will go to the matching right curly bracket. The matching right curly bracket is positioned on the bottom line of the screen. If there is more than one left curly bracket on the top line, a number N may be used to specify the N-th bracket on the line.

} If a right curly bracket appears in the bottom line displayed on the screen, the } command will go to the matching left curly bracket. The matching left curly bracket is positioned on the top line of the screen. If there is more than one right curly bracket on the top line, a number N may be used to specify the N-th bracket on the line.

(Like {, but applies to parentheses rather than curly brackets.

) Like }, but applies to parentheses rather than curly brackets.

[Like {, but applies to square brackets rather than curly brackets.

] Like }, but applies to square brackets rather than curly brackets.

ESC-^F Followed by two characters, acts like {, but uses the two characters as open and close brackets, respectively. For example, "ESC ^F < >" could be used to go forward to the > which matches the < in the top displayed line.

ESC-^B
Followed by two characters, acts like }, but uses the two characters as open and close brackets, respectively. For example, "ESC ^B < >" could be used to go backward to the < which matches the > in the bottom displayed line.

m Followed by any lowercase letter, marks the current position with that letter.

' (Single quote.) Followed by any lowercase letter, returns to the position which was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. Followed by a ^ or $, jumps to the beginning or end of the file respectively. Marks are preserved when a new file is examined, so the ' command can be used to switch between input files.

^X^X Same as single quote.

/pattern Search forward in the file for the N-th line containing the pattern. N defaults to 1. When invoked as less, the pattern is an extended regular expression. Otherwise, the pattern is a basic regular expression, as recognized by *ed*. The search starts at the second line displayed (but see the −a and −j options, which change this).

Certain characters are special if entered at the beginning of the pattern; they modify the type of search rather than become part of the pattern:

^N or ! Search for lines which do NOT match the pattern.

^E or * Search multiple files. That is, if the search reaches the END of the current file without finding a match, the search continues in the next file in the command line list. The * modifier is available when invoked as less only.

^F or @
Begin the search at the first line of the FIRST file in the command line list, regardless of what is currently displayed on the screen or the settings of the −a or −j options. The @ modifier is available when invoked as less only.

^K Highlight any text which matches the pattern on the current screen, but don't move to the first match (KEEP current position).

^R Don't interpret regular expression metacharacters; that is, do a simple textual comparison.

?pattern
Search backward in the file for the N-th line containing the pattern. The search starts at the line immediately before the top line displayed.

Certain characters are special as in the / command:

^N or ! Search for lines which do NOT match the pattern.

^E or * Search multiple files. That is, if the search reaches the beginning of the current file without finding a match, the search continues in the previous file in the command line list.

^F or @
Begin the search at the last line of the last file in the command line list, regardless of what is currently displayed on the screen or the settings of the −a or −j options.

^K As in forward searches.

^R As in forward searches.

ESC-/pattern
Same as "/*".

ESC-?pattern
Same as "?*".

n Repeat previous search, for N-th line containing the last pattern. If the previous search was modified by ^N, the search is made for the N-th line NOT containing the pattern. If the previous search was modified by ^E, the search continues in the next (or previous) file if not satisfied in the current file. If the previous search was modified by ^R, the search is done without using regular expressions. There is no effect if the previous search was modified by ^F or ^K.

N Repeat previous search, but in the reverse direction.

ESC-n Repeat previous search, but crossing file boundaries. The effect is as if the previous search were modified by *.

ESC-N Repeat previous search, but in the reverse direction and crossing file boundaries.

ESC-u Undo search highlighting. Turn off highlighting of strings matching the current search pattern. If highlighting is already off because of a previous ESC-u command, turn highlighting back on. Any search command will also turn highlighting back on. (Highlighting can also be disabled by toggling the −G option; in that case search commands do not turn highlighting back on.)

:e [filename]
Examine a new file. If the filename is missing, the "current" file (see the :n and :p commands below) from the list of files in the command line is re-examined. A percent sign (%) in the filename is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. However, two consecutive percent signs are simply replaced with a single percent sign. This allows you to enter a filename that contains a percent sign in the name. Similarly, two consecutive pound signs are replaced with a single pound sign. The filename is inserted into the command line list of files so that it can be seen by subsequent :n and :p commands. If the filename consists of several files, they are all inserted into the list of files and the first one is examined. If the filename contains one or more spaces, the entire filename should be enclosed in double quotes (also see the −" option).

^X^V or E
Same as :e. Warning: some systems use ^V as a special literalization character. On such systems, you may not be able to use ^V.

:n Examine the next file (from the list of files given in the command line). If a number N is specified, the N-th next file is examined.

:p Examine the previous file in the command line list. If a number N is specified, the N-th previous file is examined.

:x Examine the first file in the command line list. If a number N is specified, the N-th file in the list is examined.

:d Remove the current file from the list of files.

t Go to the next tag, if there were more than one matches for the current tag. See the −t option for more details about tags.

T Go to the previous tag, if there were more than one matches for the current tag.

= or ^G or :f
Prints some information about the file being viewed, including its name and the line number and byte offset of the bottom line being displayed. If possible, it also prints the length of the file, the number of lines in the file and the percent of the file above the last displayed line.

− Followed by one of the command line option letters (see OPTIONS below), this will change the setting of that option and print a message describing the new setting. If a ^P (CONTROL-P) is entered immediately after the dash, the setting of the option is changed but no message is printed. If the option letter has a numeric value (such as −b or −h), or a string value (such as −P or −t), a new value may be entered after the option letter. If no new value is entered, a message describing the current setting is printed and nothing is changed.

−− Like the − command, but takes a long option name (see OPTIONS below) rather than a single option letter. You must press RETURN after typing the option name. A ^P immediately after the second dash suppresses printing of a message describing the new setting, as in the − command.

−+ Followed by one of the command line option letters this will reset the option to its default setting and print a message describing the new setting. (The "−+*X*" command does the same thing as "−+*X*" on the command line.) This does not work for string-valued options.

−−+ Like the −+ command, but takes a long option name rather than a single option letter.

−! Followed by one of the command line option letters, this will reset the option to the "opposite" of its default setting and print a message describing the new setting. This does not work for numeric or string-valued options.

−−! Like the −! command, but takes a long option name rather than a single option letter.

_ (Underscore.) Followed by one of the command line option letters, this will print a message describing the current setting of that option. The setting of the option is not changed.

__ (Double underscore.) Like the _ (underscore) command, but takes a long option name rather than a single option letter. You must press RETURN after typing the option name.

+cmd Causes the specified cmd to be executed each time a new file is examined. For example, +G causes *less* to initially display each file starting at the end rather than the beginning.

V Prints the version number of *less* being run.

q or Q or :q or :Q or ZZ
Exits *less.*

The following four commands may or may not be valid, depending on your particular installation.

v Invokes an editor to edit the current file being viewed. The editor is taken from the environment variable VISUAL if defined, or EDITOR if VISUAL is not defined, or defaults to "vi" if neither VISUAL nor EDITOR is defined. See also the discussion of LESSEDIT under the section on PROMPTS below.

! shell-command
Invokes a shell to run the shell-command given. A percent sign (%) in the command is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. "!!" repeats the last shell command. "!" with no shell command simply invokes a shell. On Unix systems, the shell is taken from the environment variable SHELL, or defaults to "sh". On MS-DOS and OS/2 systems, the shell is the normal command processor.

| <m> shell-command
<m> represents any mark letter. Pipes a section of the input file to the given shell command. The section of the file to be piped is between the first line on the current screen and the position marked by the letter. <m> may also be ^ or $ to indicate beginning or end of file respectively. If <m> is . or newline, the current screen is piped.

s filename
Save the input to a file. This only works if the input is a pipe, not an ordinary file.

OPTIONS

Command line options are described below. Most options may be changed while *less* is running, via the "−" command.

Most options may be given in one of two forms: either a dash followed by a single letter, or two dashes followed by a long option name. A long option name may be abbreviated as long as the abbreviation is unambiguous. For example, −−quit-at-eof may be abbreviated −−quit, but not --qui, since both −−quit-at-eof and −−quiet begin with −−qui. Some long option names are in uppercase, such as −−QUIT-AT-EOF, as distinct from −−quit-at-eof. Such option names need only have their first letter capitalized; the remainder of the name may be in either case. For example, −−Quit-at-eof is equivalent to −−QUIT-AT-EOF.

Options are also taken from the environment variable "LESS". For example, to avoid typing "less −options ..." each time *less* is invoked, you might tell *csh:*

setenv LESS "-options"

or if you use *sh:*

LESS="-options"; export LESS

On MS-DOS, you don't need the quotes, but you should replace any percent signs in the options string by double percent signs.

The environment variable is parsed before the command line, so command line options override the LESS environment variable. If an option appears in the LESS variable, it can be reset to its default value on the command line by beginning the command line option with "−+".

For options like −P or −D which take a following string, a dollar sign ($) must be used to signal the end of the string. For example, to set two −D options on MS-DOS, you must have a dollar sign between them, like this:

LESS="-Dn9.1$-Ds4.1"

−? or −−help
: This option displays a summary of the commands accepted by *less* (the same as the h command). (Depending on how your shell interprets the question mark, it may be necessary to quote the question mark, thus: "−\?".)

−a or −−search-skip-screen
: Causes searches to start after the last line displayed on the screen, thus skipping all lines displayed on the screen. By default, searches start at the second line on the screen (or after the last found line; see the −j option).

−b*n* or −−buffers=*n*
: Specifies the amount of buffer space *less* will use for each file, in units of kilobytes (1024 bytes). By default 64K of buffer space is used for each file (unless the file is a pipe; see the −B option). The −b option specifies instead that *n* kilobytes of buffer space should be used for each file. If *n* is −1, buffer space is unlimited; that is, the entire file is read into memory.

−B or −−auto-buffers
: By default, when data is read from a pipe, buffers are allocated automatically as needed. If a large amount of data is read from the pipe, this can cause a large amount of memory to be allocated. The −B option disables this automatic allocation of buffers for pipes, so that only 64K (or the amount of space specified by the −b option) is used for the pipe. Warning: use of −B can result in erroneous display, since only the most recently viewed part of the file is kept in memory; any earlier data is lost.

−c or −−clear-screen
: Causes full screen repaints to be painted from the top line down. By default, full screen repaints are done by scrolling from the bottom of the screen.

−C or −−CLEAR-SCREEN
: The −C option is like −c, but the screen is cleared before it is repainted.

−d
: The -d option causes the default prompt to include the basic directions ‘‘[Press space to continue, ’q’ to quit.]’’. The -d option also causes the message ‘‘[Press ’h’ for instructions.]’’ to be displayed when an invalid command is entered (normally, the bell is rung). This option is useful in environments where users may not be experienced with pagers.

−−dumb"

The −−dumb option suppresses the error message normally displayed if the terminal is dumb; that is, lacks some important capability, such as the ability to clear the screen or scroll backward. The −−dumb option does not otherwise change the behavior of *less* on a dumb terminal.

−D**x***color* or −−color=**x***color*

[MS-DOS only] Sets the color of the text displayed. **x** is a single character which selects the type of text whose color is being set: n=normal, s=standout, d=bold, u=underlined, k=blink. *color* is a pair of numbers separated by a period. The first number selects the foreground color and the second selects the background color of the text. A single number *N* is the same as *N.0*.

−e or −−quit-at-eof

Causes *less* to automatically exit the second time it reaches end-of-file. By default, the only way to exit *less* is via the "q" command.

−E or −−QUIT-AT-EOF

Causes *less* to automatically exit the first time it reaches end-of-file.

−f or −−force

Forces non-regular files to be opened. (A non-regular file is a directory or a device special file.) Also suppresses the warning message when a binary file is opened. By default, *less* will refuse to open non-regular files.

−F or −−quit-if-one-screen

Causes *less* to automatically exit if the entire file can be displayed on the first screen.

−g or −−hilite-search

Normally, *less* will highlight ALL strings which match the last search command. The −g option changes this behavior to highlight only the particular string which was found by the last search command. This can cause *less* to run somewhat faster than the default.

−G or −−HILITE-SEARCH

The −G option suppresses all highlighting of strings found by search commands.

−h*n* or −−max-back-scroll=*n*

Specifies a maximum number of lines to scroll backward. If it is necessary to scroll backward more than *n* lines, the screen is repainted in a forward direction instead. (If the terminal does not have the ability to scroll backward, −h0 is implied.)

−i or −−ignore-case

Causes searches to ignore case; that is, uppercase and lowercase are considered identical. This option is ignored if any uppercase letters appear in the search pattern; in other words, if a pattern contains uppercase letters, then that search does not ignore case.

−I or −−IGNORE-CASE

Like −i, but searches ignore case even if the pattern contains uppercase letters.

−j*n* or −−jump-target=*n*

Specifies a line on the screen where the "target" line is to be positioned. A target line is the object of a text search, tag search, jump to a line number, jump to a file percentage, or jump to a marked position. The screen line is specified by a number: the top line on the screen is 1, the next is 2, and so on. The number may be negative to specify a line relative to the bottom of the screen: the bottom line on the screen is −1, the second to the bottom is −2, and so on. If the −j option is used, searches begin at the line immediately after the target line. For example, if "−j4" is used, the target line is the fourth line on the screen, so searches begin at the fifth line on the screen.

−J or −−status-column

Displays a status column at the left edge of the screen. The status column shows the lines that matched the current search. The status column is also used if the −w or −W option is in effect.

−k*filename* or −−lesskey-file=*filename*
Causes *less* to open and interpret the named file as a *lesskey* (1) file. Multiple −k options may be specified. If the LESSKEY or LESSKEY_SYSTEM environment variable is set, or if a lesskey file is found in a standard place (see KEY BINDINGS), it is also used as a *lesskey* file.

−K or −−quit-on-intr
Causes *less* to exit immediately when an interrupt character (usually ^C) is typed. Normally, an interrupt character causes *less* to stop whatever it is doing and return to its command prompt.

−L or −−no-lessopen
Ignore the LESSOPEN environment variable (see the INPUT PREPROCESSOR section below). This option can be set from within *less*, but it will apply only to files opened subsequently, not to the file which is currently open.

−m or −−long-prompt
Causes *less* to prompt verbosely (like *more*), with the percent into the file. By default, *less* prompts with a colon.

−M or −−LONG-PROMPT
Causes *less* to prompt even more verbosely than *more.*

−n or −−line-numbers
Suppresses line numbers. The default (to use line numbers) may cause *less* to run more slowly in some cases, especially with a very large input file. Suppressing line numbers with the −n option will avoid this problem. Using line numbers means: the line number will be displayed in the verbose prompt and in the = command, and the v command will pass the current line number to the editor (see also the discussion of LESSEDIT in PROMPTS below).

−N or −−LINE-NUMBERS
Causes a line number to be displayed at the beginning of each line in the display.

−o*filename* or −−log-file=*filename*
Causes *less* to copy its input to the named file as it is being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, *less* will ask for confirmation before overwriting it.

−O*filename* or −−LOG-FILE=*filename*
The −O option is like −o, but it will overwrite an existing file without asking for confirmation.

If no log file has been specified, the −o and −O options can be used from within *less* to specify a log file. Without a file name, they will simply report the name of the log file. The "s" command is equivalent to specifying −o from within *less.*

−p*pattern* or −−pattern=*pattern*
The −p option on the command line is equivalent to specifying +/*pattern*; that is, it tells *less* to start at the first occurrence of *pattern* in the file.

−P*prompt* or −−prompt=*prompt*
Provides a way to tailor the three prompt styles to your own preference. This option would normally be put in the LESS environment variable, rather than being typed in with each *less* command. Such an option must either be the last option in the LESS variable, or be terminated by a dollar sign. -Ps followed by a string changes the default (short) prompt to that string. -Pm changes the medium (−m) prompt. -PM changes the long (−M) prompt. -Ph changes the prompt for the help screen. -P= changes the message printed by the = command. -Pw changes the message printed while waiting for data (in the F command). All prompt strings consist of a sequence of letters and special escape sequences. See the section on PROMPTS for more details.

−q or −−quiet or −−silent
Causes moderately "quiet" operation: the terminal bell is not rung if an attempt is made to scroll past the end of the file or before the beginning of the file. If the terminal has a "visual bell", it is used instead. The bell will be rung on certain other errors, such as typing an invalid character.

The default is to ring the terminal bell in all such cases.

−Q or −−QUIET or −−SILENT
Causes totally "quiet" operation: the terminal bell is never rung.

−r or −−raw-control-chars
Causes "raw" control characters to be displayed. The default is to display control characters using the caret notation; for example, a control-A (octal 001) is displayed as "^A". Warning: when the −r option is used, *less* cannot keep track of the actual appearance of the screen (since this depends on how the screen responds to each type of control character). Thus, various display problems may result, such as long lines being split in the wrong place.

−R or −−RAW-CONTROL-CHARS
Like −r, but only ANSI "color" escape sequences are output in "raw" form. Unlike −r, the screen appearance is maintained correctly in most cases. ANSI "color" escape sequences are sequences of the form:

ESC [... m

where the "..." is zero or more color specification characters For the purpose of keeping track of screen appearance, ANSI color escape sequences are assumed to not move the cursor. You can make *less* think that characters other than "m" can end ANSI color escape sequences by setting the environment variable LESSANSIENDCHARS to the list of characters which can end a color escape sequence. And you can make *less* think that characters other than the standard ones may appear between the ESC and the m by setting the environment variable LESSANSIMIDCHARS to the list of characters which can appear.

−s or −−squeeze-blank-lines
Causes consecutive blank lines to be squeezed into a single blank line. This is useful when viewing *nroff* output.

−S or −−chop-long-lines
Causes lines longer than the screen width to be chopped rather than folded. That is, the portion of a long line that does not fit in the screen width is not shown. The default is to fold long lines; that is, display the remainder on the next line.

−t*tag* or −−tag=*tag*
The −t option, followed immediately by a TAG, will edit the file containing that tag. For this to work, tag information must be available; for example, there may be a file in the current directory called "tags", which was previously built by *ctags* (1) or an equivalent command. If the environment variable LESSGLOBALTAGS is set, it is taken to be the name of a command compatible with *global* (1), and that command is executed to find the tag. (See http://www.gnu.org/software/global/global.html). The −t option may also be specified from within *less* (using the − command) as a way of examining a new file. The command ":t" is equivalent to specifying −t from within *less*.

−T*tagsfile* or −−tag-file=*tagsfile*
Specifies a tags file to be used instead of "tags".

−u or −−underline-special
Causes backspaces and carriage returns to be treated as printable characters; that is, they are sent to the terminal when they appear in the input.

−U or −−UNDERLINE-SPECIAL
Causes backspaces, tabs and carriage returns to be treated as control characters; that is, they are handled as specified by the −r option.

By default, if neither −u nor −U is given, backspaces which appear adjacent to an underscore character are treated specially: the underlined text is displayed using the terminal's hardware underlining capability. Also, backspaces which appear between two identical characters are treated

specially: the overstruck text is printed using the terminal's hardware boldface capability. Other backspaces are deleted, along with the preceding character. Carriage returns immediately followed by a newline are deleted. other carriage returns are handled as specified by the −r option. Text which is overstruck or underlined can be searched for if neither −u nor −U is in effect.

−V or −−version
Displays the version number of *less.*

−w or −−hilite-unread
Temporarily highlights the first "new" line after a forward movement of a full page. The first "new" line is the line immediately following the line previously at the bottom of the screen. Also highlights the target line after a g or p command. The highlight is removed at the next command which causes movement. The entire line is highlighted, unless the −J option is in effect, in which case only the status column is highlighted.

−W or −−HILITE-UNREAD
Like −w, but temporarily highlights the first new line after any forward movement command larger than one line.

−x*n*,... or −−tabs=*n*,...
Sets tab stops. If only one *n* is specified, tab stops are set at multiples of *n*. If multiple values separated by commas are specified, tab stops are set at those positions, and then continue with the same spacing as the last two. For example, *-x9,17* will set tabs at positions 9, 17, 25, 33, etc. The default for *n* is 8.

−X or −−no-init
Disables sending the termcap initialization and deinitialization strings to the terminal. This is sometimes desirable if the deinitialization string does something unnecessary, like clearing the screen.

−−no-keypad
Disables sending the keypad initialization and deinitialization strings to the terminal. This is sometimes useful if the keypad strings make the numeric keypad behave in an undesirable manner.

−y*n* or −−max-forw-scroll=*n*
Specifies a maximum number of lines to scroll forward. If it is necessary to scroll forward more than *n* lines, the screen is repainted instead. The −c or −C option may be used to repaint from the top of the screen if desired. By default, any forward movement causes scrolling.

−[z]*n* or −−window=*n*
Changes the default scrolling window size to *n* lines. The default is one screenful. The z and w commands can also be used to change the window size. The "z" may be omitted for compatibility with *more.* If the number *n* is negative, it indicates *n* lines less than the current screen size. For example, if the screen is 24 lines, *−z-4* sets the scrolling window to 20 lines. If the screen is resized to 40 lines, the scrolling window automatically changes to 36 lines.

−*"cc* or −−quotes=*cc*
Changes the filename quoting character. This may be necessary if you are trying to name a file which contains both spaces and quote characters. Followed by a single character, this changes the quote character to that character. Filenames containing a space should then be surrounded by that character rather than by double quotes. Followed by two characters, changes the open quote to the first character, and the close quote to the second character. Filenames containing a space should then be preceded by the open quote character and followed by the close quote character. Note that even after the quote characters are changed, this option remains −" (a dash followed by a double quote).

−~ or −−tilde
Normally lines after end of file are displayed as a single tilde (~). This option causes lines after end of file to be displayed as blank lines.

−# or −−shift
Specifies the default number of positions to scroll horizontally in the RIGHTARROW and LEFTARROW commands. If the number specified is zero, it sets the default number of positions to one half of the screen width.

−− A command line argument of "−−" marks the end of option arguments. Any arguments following this are interpreted as filenames. This can be useful when viewing a file whose name begins with a "−" or "+".

+ If a command line option begins with +, the remainder of that option is taken to be an initial command to *less*. For example, +G tells *less* to start at the end of the file rather than the beginning, and +/xyz tells it to start at the first occurrence of "xyz" in the file. As a special case, +<number> acts like +<number>g; that is, it starts the display at the specified line number (however, see the caveat under the "g" command above). If the option starts with ++, the initial command applies to every file being viewed, not just the first one. The + command described previously may also be used to set (or change) an initial command for every file.

LINE EDITING

When entering command line at the bottom of the screen (for example, a filename for the :e command, or the pattern for a search command), certain keys can be used to manipulate the command line. Most commands have an alternate form in [brackets] which can be used if a key does not exist on a particular keyboard. (The bracketed forms do not work in the MS-DOS version.) Any of these special keys may be entered literally by preceding it with the "literal" character, either ^V or ^A. A backslash itself may also be entered literally by entering two backslashes.

LEFTARROW [ESC-h]
Move the cursor one space to the left.

RIGHTARROW [ESC-l]
Move the cursor one space to the right.

^LEFTARROW [ESC-b or ESC-LEFTARROW]
(That is, CONTROL and LEFTARROW simultaneously.) Move the cursor one word to the left.

^RIGHTARROW [ESC-w or ESC-RIGHTARROW]
(That is, CONTROL and RIGHTARROW simultaneously.) Move the cursor one word to the right.

HOME [ESC-0]
Move the cursor to the beginning of the line.

END [ESC-$]
Move the cursor to the end of the line.

BACKSPACE
Delete the character to the left of the cursor, or cancel the command if the command line is empty.

DELETE or [ESC-x]
Delete the character under the cursor.

^BACKSPACE [ESC-BACKSPACE]
(That is, CONTROL and BACKSPACE simultaneously.) Delete the word to the left of the cursor.

^DELETE [ESC-X or ESC-DELETE]
(That is, CONTROL and DELETE simultaneously.) Delete the word under the cursor.

UPARROW [ESC-k]
Retrieve the previous command line.

DOWNARROW [ESC-j]
Retrieve the next command line.

TAB Complete the partial filename to the left of the cursor. If it matches more than one filename, the first match is entered into the command line. Repeated TABs will cycle thru the other matching

filenames. If the completed filename is a directory, a "/" is appended to the filename. (On MS-DOS systems, a "\" is appended.) The environment variable LESSSEPARATOR can be used to specify a different character to append to a directory name.

BACKTAB [ESC-TAB]
Like, TAB, but cycles in the reverse direction thru the matching filenames.

^L Complete the partial filename to the left of the cursor. If it matches more than one filename, all matches are entered into the command line (if they fit).

^U (Unix and OS/2) or ESC (MS-DOS)
Delete the entire command line, or cancel the command if the command line is empty. If you have changed your line-kill character in Unix to something other than ^U, that character is used instead of ^U.

KEY BINDINGS

You may define your own *less* commands by using the program *lesskey* (1) to create a lesskey file. This file specifies a set of command keys and an action associated with each key. You may also use *lesskey* to change the line-editing keys (see LINE EDITING), and to set environment variables. If the environment variable LESSKEY is set, *less* uses that as the name of the lesskey file. Otherwise, *less* looks in a standard place for the lesskey file: On Unix systems, *less* looks for a lesskey file called "$HOME/.less". On MS-DOS and Windows systems, *less* looks for a lesskey file called "$HOME/_less", and if it is not found there, then looks for a lesskey file called "_less" in any directory specified in the PATH environment variable. On OS/2 systems, *less* looks for a lesskey file called "$HOME/less.ini", and if it is not found, then looks for a lesskey file called "less.ini" in any directory specified in the INIT environment variable, and if it not found there, then looks for a lesskey file called "less.ini" in any directory specified in the PATH environment variable. See the *lesskey* manual page for more details.

A system-wide lesskey file may also be set up to provide key bindings. If a key is defined in both a local lesskey file and in the system-wide file, key bindings in the local file take precedence over those in the system-wide file. If the environment variable LESSKEY_SYSTEM is set, *less* uses that as the name of the system-wide lesskey file. Otherwise, *less* looks in a standard place for the system-wide lesskey file: On NetBSD, the system-wide lesskey file is in /etc/sysless. On other Unix systems, the system-wide lesskey file is /usr/local/etc/sysless. (However, if *less* was built with a different sysconf directory than /usr/local/etc, that directory is where the sysless file is found.) On MS-DOS and Windows systems, the system-wide lesskey file is c:_sysless. On OS/2 systems, the system-wide lesskey file is c:\sysless.ini.

INPUT PREPROCESSOR

You may define an "input preprocessor" for *less*. Before *less* opens a file, it first gives your input preprocessor a chance to modify the way the contents of the file are displayed. An input preprocessor is simply an executable program (or shell script), which writes the contents of the file to a different file, called the replacement file. The contents of the replacement file are then displayed in place of the contents of the original file. However, it will appear to the user as if the original file is opened; that is, *less* will display the original filename as the name of the current file.

An input preprocessor receives one command line argument, the original filename, as entered by the user. It should create the replacement file, and when finished, print the name of the replacement file to its standard output. If the input preprocessor does not output a replacement filename, *less* uses the original file, as normal. The input preprocessor is not called when viewing standard input. To set up an input preprocessor, set the LESSOPEN environment variable to a command line which will invoke your input preprocessor. This command line should include one occurrence of the string "%s", which will be replaced by the filename when the input preprocessor command is invoked.

When *less* closes a file opened in such a way, it will call another program, called the input postprocessor, which may perform any desired clean-up action (such as deleting the replacement file created by LESSOPEN). This program receives two command line arguments, the original filename as entered by the user, and the name of the replacement file. To set up an input postprocessor, set the LESSCLOSE

environment variable to a command line which will invoke your input postprocessor. It may include two occurrences of the string "%s"; the first is replaced with the original name of the file and the second with the name of the replacement file, which was output by LESSOPEN.

For example, on many Unix systems, these two scripts will allow you to keep files in compressed format, but still let *less* view them directly:

lessopen.sh:

```
#! /bin/sh
case "$1" in
*.Z)    uncompress −c $1  >/tmp/less.$$  2>/dev/null
        if [ −s /tmp/less.$$ ]; then
                echo /tmp/less.$$
        else
                rm −f /tmp/less.$$
        fi
        ;;
esac
```

lessclose.sh:

```
#! /bin/sh
rm $2
```

To use these scripts, put them both where they can be executed and set LESSOPEN="lessopen.sh %s", and LESSCLOSE="lessclose.sh %s %s". More complex LESSOPEN and LESSCLOSE scripts may be written to accept other types of compressed files, and so on.

It is also possible to set up an input preprocessor to pipe the file data directly to *less*, rather than putting the data into a replacement file. This avoids the need to decompress the entire file before starting to view it. An input preprocessor that works this way is called an input pipe. An input pipe, instead of writing the name of a replacement file on its standard output, writes the entire contents of the replacement file on its standard output. If the input pipe does not write any characters on its standard output, then there is no replacement file and *less* uses the original file, as normal. To use an input pipe, make the first character in the LESSOPEN environment variable a vertical bar (|) to signify that the input preprocessor is an input pipe.

For example, on many Unix systems, this script will work like the previous example scripts:

lesspipe.sh:

```
#! /bin/sh
case "$1" in
*.Z)    uncompress −c $1  2>/dev/null
        ;;
esac
```

To use this script, put it where it can be executed and set LESSOPEN="|lesspipe.sh %s". When an input pipe is used, a LESSCLOSE postprocessor can be used, but it is usually not necessary since there is no replacement file to clean up. In this case, the replacement file name passed to the LESSCLOSE postprocessor is "−".

NATIONAL CHARACTER SETS

There are three types of characters in the input file:

normal characters
: can be displayed directly to the screen.

control characters
: should not be displayed directly, but are expected to be found in ordinary text files (such as backspace and tab).

binary characters
should not be displayed directly and are not expected to be found in text files.

A "character set" is simply a description of which characters are to be considered normal, control, and binary. The LESSCHARSET environment variable may be used to select a character set. Possible values for LESSCHARSET are:

ascii BS, TAB, NL, CR, and formfeed are control characters, all chars with values between 32 and 126 are normal, and all others are binary.

iso8859
Selects an ISO 8859 character set. This is the same as ASCII, except characters between 160 and 255 are treated as normal characters.

latin1 Same as iso8859.

latin9 Same as iso8859.

dos Selects a character set appropriate for MS-DOS.

ebcdic Selects an EBCDIC character set.

IBM-1047
Selects an EBCDIC character set used by OS/390 Unix Services. This is the EBCDIC analogue of latin1. You get similar results by setting either LESSCHARSET=IBM-1047 or LC_CTYPE=en_US in your environment.

koi8-r Selects a Russian character set.

next Selects a character set appropriate for NeXT computers.

utf-8 Selects the UTF-8 encoding of the ISO 10646 character set.

windows
Selects a character set appropriate for Microsoft Windows (cp 1251).

In special cases, it may be desired to tailor *less* to use a character set other than the ones definable by LESSCHARSET. In this case, the environment variable LESSCHARDEF can be used to define a character set. It should be set to a string where each character in the string represents one character in the character set. The character "." is used for a normal character, "c" for control, and "b" for binary. A decimal number may be used for repetition. For example, "bccc4b." would mean character 0 is binary, 1, 2 and 3 are control, 4, 5, 6 and 7 are binary, and 8 is normal. All characters after the last are taken to be the same as the last, so characters 9 through 255 would be normal. (This is an example, and does not necessarily represent any real character set.)

This table shows the value of LESSCHARDEF which is equivalent to each of the possible values for LESSCHARSET:

ascii 8bcccbcc18b95.b
dos 8bcccbcc12bc5b95.b.
ebcdic 5bc6bcc7bcc41b.9b7.9b5.b..8b6.10b6.b9.7b
9.8b8.17b3.3b9.7b9.8b8.6b10.b.b.b.
IBM-1047 4cbcbc3b9cbccbccbb4c6bcc5b3cbbc4bc4bccbc
191.b
iso8859 8bcccbcc18b95.33b.
koi8-r 8bcccbcc18b95.b128.
latin1 8bcccbcc18b95.33b.
next 8bcccbcc18b95.bb125.bb

If neither LESSCHARSET nor LESSCHARDEF is set, but any of the strings "UTF-8", "UTF8", "utf-8" or "utf8" is found in the LC_ALL, LC_TYPE or LANG environment variables, then the default character set is utf-8.

If that string is not found, but your system supports the *setlocale* interface, *less* will use setlocale to

determine the character set. setlocale is controlled by setting the LANG or LC_CTYPE environment variables.

Finally, if the *setlocale* interface is also not available, the default character set is latin1.

Control and binary characters are displayed in standout (reverse video). Each such character is displayed in caret notation if possible (e.g. ^A for control-A). Caret notation is used only if inverting the 0100 bit results in a normal printable character. Otherwise, the character is displayed as a hex number in angle brackets. This format can be changed by setting the LESSBINFMT environment variable. LESSBINFMT may begin with a "*" and one character to select the display attribute: "*k" is blinking, "*d" is bold, "*u" is underlined, "*s" is standout, and "*n" is normal. If LESSBINFMT does not begin with a "*", normal attribute is assumed. The remainder of LESSBINFMT is a string which may include one printf-style escape sequence (a % followed by x, X, o, d, etc.). For example, if LESSBINFMT is "*u[%x]", binary characters are displayed in underlined hexadecimal surrounded by brackets. The default if no LESSBINFMT is specified is "*s<%X>". The default if no LESSBINFMT is specified is "*s<%02X>". Warning: the result of expanding the character via LESSBINFMT must be less than 31 characters.

When the character set is utf-8, the LESSUTFBINFMT environment variable acts similarly to LESSBINFMT but it applies to Unicode code points that were successfully decoded but are unsuitable for display (e.g., unassigned code points). Its default value is "<U+%04lX>". Note that LESSUTFBINFMT and LESSBINFMT share their display attribute setting ("*x") so specifying one will affect both; LESSUTFBINFMT is read after LESSBINFMT so its setting, if any, will have priority. Problematic octets in a UTF-8 file (octets of a truncated sequence, octets of a complete but non-shortest form sequence, illegal octets, and stray trailing octets) are displayed individually using LESSBINFMT so as to facilitate diagnostic of how the UTF-8 file is ill-formed.

PROMPTS

The −P option allows you to tailor the prompt to your preference. The string given to the −P option replaces the specified prompt string. Certain characters in the string are interpreted specially. The prompt mechanism is rather complicated to provide flexibility, but the ordinary user need not understand the details of constructing personalized prompt strings.

A percent sign followed by a single character is expanded according to what the following character is:

%b*X* Replaced by the byte offset into the current input file. The b is followed by a single character (shown as *X* above) which specifies the line whose byte offset is to be used. If the character is a "t", the byte offset of the top line in the display is used, an "m" means use the middle line, a "b" means use the bottom line, a "B" means use the line just after the bottom line, and a "j" means use the "target" line, as specified by the −j option.

%B Replaced by the size of the current input file.

%c Replaced by the column number of the text appearing in the first column of the screen.

%d*X* Replaced by the page number of a line in the input file. The line to be used is determined by the *X*, as with the %b option.

%D Replaced by the number of pages in the input file, or equivalently, the page number of the last line in the input file.

%E Replaced by the name of the editor (from the VISUAL environment variable, or the EDITOR environment variable if VISUAL is not defined). See the discussion of the LESSEDIT feature below.

%f Replaced by the name of the current input file.

%i Replaced by the index of the current file in the list of input files.

%l*X* Replaced by the line number of a line in the input file. The line to be used is determined by the *X*, as with the %b option.

%L Replaced by the line number of the last line in the input file.

%m Replaced by the total number of input files.

%p*X* Replaced by the percent into the current input file, based on byte offsets. The line used is determined by the *X* as with the %b option.

%P*X* Replaced by the percent into the current input file, based on line numbers. The line used is determined by the *X* as with the %b option.

%s Same as %B.

%t Causes any trailing spaces to be removed. Usually used at the end of the string, but may appear anywhere.

%x Replaced by the name of the next input file in the list.

If any item is unknown (for example, the file size if input is a pipe), a question mark is printed instead.

The format of the prompt string can be changed depending on certain conditions. A question mark followed by a single character acts like an "IF": depending on the following character, a condition is evaluated. If the condition is true, any characters following the question mark and condition character, up to a period, are included in the prompt. If the condition is false, such characters are not included. A colon appearing between the question mark and the period can be used to establish an "ELSE": any characters between the colon and the period are included in the string if and only if the IF condition is false. Condition characters (which follow a question mark) may be:

?a True if any characters have been included in the prompt so far.

?b*X* True if the byte offset of the specified line is known.

?B True if the size of current input file is known.

?c True if the text is horizontally shifted (%c is not zero).

?d*X* True if the page number of the specified line is known.

?e True if at end-of-file.

?f True if there is an input filename (that is, if input is not a pipe).

?l*X* True if the line number of the specified line is known.

?L True if the line number of the last line in the file is known.

?m True if there is more than one input file.

?n True if this is the first prompt in a new input file.

?p*X* True if the percent into the current input file, based on byte offsets, of the specified line is known.

?P*X* True if the percent into the current input file, based on line numbers, of the specified line is known.

?s Same as "?B".

?x True if there is a next input file (that is, if the current input file is not the last one).

Any characters other than the special ones (question mark, colon, period, percent, and backslash) become literally part of the prompt. Any of the special characters may be included in the prompt literally by preceding it with a backslash.

Some examples:

?f%f:Standard input.

This prompt prints the filename, if known; otherwise the string "Standard input".

?f%f .?ltLine %lt:?pt%pt\%:?btByte %bt:-...

This prompt would print the filename, if known. The filename is followed by the line number, if known, otherwise the percent if known, otherwise the byte offset if known. Otherwise, a dash is printed. Notice

how each question mark has a matching period, and how the % after the %pt is included literally by escaping it with a backslash.

?n?f%f .?m(file %i of %m) ..?e(END) ?x- Next\: %x..%t

This prints the filename if this is the first prompt in a file, followed by the "file N of N" message if there is more than one input file. Then, if we are at end-of-file, the string "(END)" is printed followed by the name of the next file, if there is one. Finally, any trailing spaces are truncated. This is the default prompt. For reference, here are the defaults for the other two prompts (−m and −M respectively). Each is broken into two lines here for readability only.

?n?f%f .?m(file %i of %m) ..?e(END) ?x- Next\: %x.:
?pB%pB\%:byte %bB?s/%s...%t

?f%f .?n?m(file %i of %m) ..?ltlines %lt-%lb?L/%L. :
byte %bB?s/%s. .?e(END) ?x- Next\: %x.:?pB%pB\%..%t

And here is the default message produced by the = command:

?f%f .?m(file %i of %m) .?ltlines %lt-%lb?L/%L. .
byte %bB?s/%s. ?e(END) :?pB%pB\%..%t

The prompt expansion features are also used for another purpose: if an environment variable LESSEDIT is defined, it is used as the command to be executed when the v command is invoked. The LESSEDIT string is expanded in the same way as the prompt strings. The default value for LESSEDIT is:

%E ?lm+%lm. %f

Note that this expands to the editor name, followed by a + and the line number, followed by the file name. If your editor does not accept the "+linenumber" syntax, or has other differences in invocation syntax, the LESSEDIT variable can be changed to modify this default.

SECURITY

When the environment variable LESSSECURE is set to 1, *less* runs in a "secure" mode. This means these features are disabled:

!	the shell command
|	the pipe command
:e	the examine command.
v	the editing command
s −o	log files
−k	use of lesskey files
−t	use of tags files
	metacharacters in filenames, such as *
	filename completion (TAB, ^L)

Less can also be compiled to be permanently in "secure" mode.

ENVIRONMENT VARIABLES

Environment variables may be specified either in the system environment as usual, or in a *lesskey* (1) file. If environment variables are defined in more than one place, variables defined in a local lesskey file take precedence over variables defined in the system environment, which take precedence over variables defined

in the system-wide lesskey file.

COLUMNS
Sets the number of columns on the screen. Takes precedence over the number of columns specified by the TERM variable. (But if you have a windowing system which supports TIOCGWINSZ or WIOCGETD, the window system's idea of the screen size takes precedence over the LINES and COLUMNS environment variables.)

EDITOR
The name of the editor (used for the v command).

HOME Name of the user's home directory (used to find a lesskey file on Unix and OS/2 systems).

HOMEDRIVE, HOMEPATH
Concatenation of the HOMEDRIVE and HOMEPATH environment variables is the name of the user's home directory if the HOME variable is not set (only in the Windows version).

INIT Name of the user's init directory (used to find a lesskey file on OS/2 systems).

LANG Language for determining the character set.

LC_CTYPE
Language for determining the character set.

LESS Options which are passed to *less* automatically.

LESSANSIENDCHARS
Characters which may end an ANSI color escape sequence (default "m").

LESSANSIMIDCHARS
Characters which may appear between the ESC character and the end character in an ANSI color escape sequence (default "0123456789;[?!"'#%()*+ ".

LESSBINFMT
Format for displaying non-printable, non-control characters.

LESSCHARDEF
Defines a character set.

LESSCHARSET
Selects a predefined character set.

LESSCLOSE
Command line to invoke the (optional) input-postprocessor.

LESSECHO
Name of the lessecho program (default "lessecho"). The lessecho program is needed to expand metacharacters, such as * and ?, in filenames on Unix systems.

LESSEDIT
Editor prototype string (used for the v command). See discussion under PROMPTS.

LESSGLOBALTAGS
Name of the command used by the −t option to find global tags. Normally should be set to "global" if your system has the *global* (1) command. If not set, global tags are not used.

LESSHISTFILE
Name of the history file used to remember search commands and shell commands between invocations of *less*. If set to "−", a history file is not used. The default is "$HOME/.lesshst" on Unix systems, "$HOME/_lesshst" on DOS and Windows systems, or "$HOME/lesshst.ini" or "$INIT/lesshst.ini" on OS/2 systems.

LESSHISTSIZE
The maximum number of commands to save in the history file. The default is 100.

LESSKEY
Name of the default lesskey(1) file.

LESSKEY_SYSTEM
Name of the default system-wide lesskey(1) file.

LESSMETACHARS
List of characters which are considered "metacharacters" by the shell.

LESSMETAESCAPE
Prefix which less will add before each metacharacter in a command sent to the shell. If LESSMETAESCAPE is an empty string, commands containing metacharacters will not be passed to the shell.

LESSOPEN
Command line to invoke the (optional) input-preprocessor.

LESSSECURE
Runs less in "secure" mode. See discussion under SECURITY.

LESSSEPARATOR
String to be appended to a directory name in filename completion.

LESSUTFBINFMT
Format for displaying non-printable Unicode code points.

LINES Sets the number of lines on the screen. Takes precedence over the number of lines specified by the TERM variable. (But if you have a windowing system which supports TIOCGWINSZ or WIOCGETD, the window system's idea of the screen size takes precedence over the LINES and COLUMNS environment variables.)

PATH User's search path (used to find a lesskey file on MS-DOS and OS/2 systems).

SHELL
The shell used to execute the ! command, as well as to expand filenames.

TERM The type of terminal on which *less* is being run.

VISUAL
The name of the editor (used for the v command).

SEE ALSO

lesskey(1)

WARNINGS

The = command and prompts (unless changed by −P) report the line numbers of the lines at the top and bottom of the screen, but the byte and percent of the line after the one at the bottom of the screen.

If the :e command is used to name more than one file, and one of the named files has been viewed previously, the new files may be entered into the list in an unexpected order.

On certain older terminals (the so-called "magic cookie" terminals), search highlighting will cause an erroneous display. On such terminals, search highlighting is disabled by default to avoid possible problems.

In certain cases, when search highlighting is enabled and a search pattern begins with a ^, more text than the matching string may be highlighted. (This problem does not occur when less is compiled to use the POSIX regular expression package.)

When viewing text containing ANSI color escape sequences using the −R option, searching will not find text containing an embedded escape sequence. Also, search highlighting may change the color of some of the text which follows the highlighted text.

On some systems, *setlocale* claims that ASCII characters 0 thru 31 are control characters rather than binary characters. This causes *less* to treat some binary files as ordinary, non-binary files. To workaround this

problem, set the environment variable LESSCHARSET to "ascii" (or whatever character set is appropriate).

This manual is too long.

See http://www.greenwoodsoftware.com/less for the list of known bugs in all versions of less.

COPYRIGHT

AUTHOR

Mark Nudelman <markn@greenwoodsoftware.com>
Send bug reports or comments to the above address or to bug-less@gnu.org.
For more information, see the less homepage at http://www.greenwoodsoftware.com/less.

NAME

lesskey − specify key bindings for less

SYNOPSIS

lesskey [-o output] [--] [input]
lesskey [--output=output] [--] [input]
lesskey -V
lesskey --version

DESCRIPTION

Lesskey is used to specify a set of key bindings to be used by *less.* The input file is a text file which describes the key bindings, If the input file is "-", standard input is read. If no input file is specified, a standard filename is used as the name of the input file, which depends on the system being used: On Unix systems, $HOME/.lesskey is used; on MS-DOS systems, $HOME/_lesskey is used; and on OS/2 systems $HOME/lesskey.ini is used, or $INIT/lesskey.ini if $HOME is undefined. The output file is a binary file which is used by *less.* If no output file is specified, and the environment variable LESSKEY is set, the value of LESSKEY is used as the name of the output file. Otherwise, a standard filename is used as the name of the output file, which depends on the system being used: On Unix and OS-9 systems, $HOME/.less is used; on MS-DOS systems, $HOME/_less is used; and on OS/2 systems, $HOME/less.ini is used, or $INIT/less.ini if $HOME is undefined. If the output file already exists, *lesskey* will overwrite it.

The -V or --version option causes *lesskey* to print its version number and immediately exit. If -V or --version is present, other options and arguments are ignored.

The input file consists of one or more *sections.* Each section starts with a line that identifies the type of section. Possible sections are:

#command
: Defines new command keys.

#line-edit
: Defines new line-editing keys.

#env
: Defines environment variables.

Blank lines and lines which start with a pound sign (#) are ignored, except for the special section header lines.

COMMAND SECTION

The command section begins with the line

#command

If the command section is the first section in the file, this line may be omitted. The command section consists of lines of the form:

string <whitespace> *action* [extra-string] <newline>

Whitespace is any sequence of one or more spaces and/or tabs. The *string* is the command key(s) which invoke the action. The *string* may be a single command key, or a sequence of up to 15 keys. The *action* is the name of the less action, from the list below. The characters in the *string* may appear literally, or be prefixed by a caret to indicate a control key. A backslash followed by one to three octal digits may be used to specify a character by its octal value. A backslash followed by certain characters specifies input characters as follows:

\b
: BACKSPACE

\e
: ESCAPE

\n
: NEWLINE

\r	RETURN
\t	TAB
\ku	UP ARROW
\kd	DOWN ARROW
\kr	RIGHT ARROW
\kl	LEFT ARROW
\kU	PAGE UP
\kD	PAGE DOWN
\kh	HOME
\ke	END
\kx	DELETE

A backslash followed by any other character indicates that character is to be taken literally. Characters which must be preceded by backslash include caret, space, tab and the backslash itself.

An action may be followed by an "extra" string. When such a command is entered while running *less*, the action is performed, and then the extra string is parsed, just as if it were typed in to *less*. This feature can be used in certain cases to extend the functionality of a command. For example, see the "{" and ":t" commands in the example below. The extra string has a special meaning for the "quit" action: when *less* quits, first character of the extra string is used as its exit status.

EXAMPLE

The following input file describes the set of default command keys used by less:

```
#command
\r              forw-line
\n              forw-line
e               forw-line
j               forw-line
\kd     forw-line
^E              forw-line
^N              forw-line
k               back-line
y               back-line
^Y              back-line
^K              back-line
^P              back-line
J               forw-line-force
K               back-line-force
Y               back-line-force
d               forw-scroll
^D              forw-scroll
u               back-scroll
^U              back-scroll
\40     forw-screen
f               forw-screen
^F              forw-screen
^V              forw-screen
\kD     forw-screen
b               back-screen
^B              back-screen
```

```
\ev             back-screen
\kU     back-screen
z               forw-window
w               back-window
\e\40           forw-screen-force
F               forw-forever
R               repaint-flush
r               repaint
^R              repaint
^L              repaint
\eu             undo-hilite
g               goto-line
\kh     goto-line
<               goto-line
\e<             goto-line
p               percent
%               percent
\e[             left-scroll
\e]             right-scroll
\e(             left-scroll
\e)             right-scroll
{               forw-bracket {}
}               back-bracket {}
(               forw-bracket ()
)               back-bracket ()
[               forw-bracket []
]               back-bracket []
\e^F            forw-bracket
\e^B            back-bracket
G               goto-end
\e>             goto-end
>               goto-end
\ke     goto-end
=               status
^G              status
:f              status
/               forw-search
?               back-search
\e/             forw-search *
\e?             back-search *
n               repeat-search
\en             repeat-search-all
N               reverse-search
\eN             reverse-search-all
m               set-mark
'               goto-mark
^X^X            goto-mark
E               examine
:e              examine
^X^V            examine
:n              next-file
:p              prev-file
t               next-tag
T               prev-tag
```

:x	index-file
:d	remove-file
-	toggle-option
:t	toggle-option t
s	toggle-option o
_	display-option
|	pipe
v	visual
!	shell
+	firstcmd
H	help
h	help
V	version
0	digit
1	digit
2	digit
3	digit
4	digit
5	digit
6	digit
7	digit
8	digit
9	digit
q	quit
Q	quit
:q	quit
:Q	quit
ZZ	quit

PRECEDENCE

Commands specified by *lesskey* take precedence over the default commands. A default command key may be disabled by including it in the input file with the action "invalid". Alternatively, a key may be defined to do nothing by using the action "noaction". "noaction" is similar to "invalid", but *less* will give an error beep for an "invalid" command, but not for a "noaction" command. In addition, ALL default commands may be disabled by adding this control line to the input file:

#stop

This will cause all default commands to be ignored. The #stop line should be the last line in that section of the file.

Be aware that #stop can be dangerous. Since all default commands are disabled, you must provide sufficient commands before the #stop line to enable all necessary actions. For example, failure to provide a "quit" command can lead to frustration.

LINE EDITING SECTION

The line-editing section begins with the line:

#line-edit

This section specifies new key bindings for the line editing commands, in a manner similar to the way key bindings for ordinary commands are specified in the #command section. The line-editing section consists of a list of keys and actions, one per line as in the example below.

EXAMPLE

The following input file describes the set of default line-editing keys used by less:

```
#line-edit
\t              forw-complete
\17             back-complete
\e\t            back-complete
^L              expand
^V              literal
^A              literal
\el             right
\kr             right
\eh             left
\kl             left
\eb             word-left
\e\kl       word-left
\ew             word-right
\e\kr       word-right
\ei             insert
\ex             delete
\kx             delete
\eX             word-delete
\ekx            word-delete
\e\b            word-backspace
\e0             home
\kh             home
\e$             end
\ke             end
\ek             up
\ku             up
\ej             down
```

LESS ENVIRONMENT VARIABLES

The environment variable section begins with the line

#env

Following this line is a list of environment variable assignments. Each line consists of an environment variable name, an equals sign (=) and the value to be assigned to the environment variable. White space before and after the equals sign is ignored. Variables assigned in this way are visible only to *less.* If a variable is specified in the system environment and also in a lesskey file, the value in the lesskey file takes precedence. Although the lesskey file can be used to override variables set in the environment, the main purpose of assigning variables in the lesskey file is simply to have all *less* configuration information stored in one file.

EXAMPLE

The following input file sets the -i option whenever *less* is run, and specifies the character set to be "latin1":

```
#env
LESS = -i
LESSCHARSET = latin1
```

SEE ALSO

less(1)

WARNINGS

It is not possible to specify special keys, such as uparrow, in a keyboard-independent manner. The only way to specify such keys is to specify the escape sequence which a particular keyboard sends when such a key is pressed.

On MS-DOS and OS/2 systems, certain keys send a sequence of characters which start with a NUL character (0). This NUL character should be represented as \340 in a lesskey file.

COPYRIGHT

AUTHOR

Mark Nudelman <markn@greenwoodsoftware.com>
Send bug reports or comments to the above address or to bug-less@gnu.org.

NAME

linkfarm — manage symbolic links to package files

SYNOPSIS

linkfarm [**-cDnRVv**] [**-d** *stowdir*] [**-s** *subdir*] [**-t** *target*] *package*

DESCRIPTION

The **linkfarm** command is used to manage a "linkfarm", a directory tree of symbolic links in *target* to the files in the *package* sub-directory of *stowdir*. The default action is to create a linkfarm to a package.

The following command-line options are supported:

-c Check whether a package contains a linkfarm in *target*. If *package* has no symbolic links in *target* that correspond to its files, then return 0, otherwise return 1.

-D Delete the linkfarm for *package*.

-d *stowdir*
Set *stowdir* as the directory in which *package* can be found. The default *stowdir* is the packages sub-directory in the default *target* directory.

-n Don't actually execute the commands for removing and creating the symbolic links and directories.

-R Delete and re-create the linkfarm for *package*.

-s *subdir*
The root of the package hierarchy for the linkfarm is the *subdir* sub-directory in the package. By default, assume the root of the package hierarchy is simply the *package* directory.

-t *target*
Set *target* as the directory in which to create and delete the linkfarm for *package*. The default *target* directory is /usr/pkg but may be overridden by the LOCALBASE environment variable.

-V Print version number and exit.

-v Turn on verbose output. Specifying **-v** multiple times increases the level of verbosity.

ENVIRONMENT

LOCALBASE
The standard packages directory, /usr/pkg, can be overridden by specifying an alternative directory in the LOCALBASE environment variable. This affects the default *target* and *stowdir* directories.

PLIST_IGNORE_FILES
This can be used to specify files in *package* that should ignored when creating and deleting symbolic links in *target*. PLIST_IGNORE_FILES is a space-separated list of shell glob patterns that match files relative to the *package* directory, and it defaults to "info/dir *[~#] *.OLD *.orig *,v".

SEE ALSO

lndir(1), pkg_view(1)

AUTHORS

The **linkfarm** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

lint — a C program verifier

SYNOPSIS

lint [**-abceghprvwxzHFV**] [**-s** | **-t**] [**-i** | **-nu**] [**-MD**] [**-D** *name*[=def]] [**-U** *name*] [**-I** *directory*] [**-d** *directory*] [**-L** *directory*] [**-l** *library*] [**-o** *outputfile*] [**-B** *directory*] [**-X** *id*[,id ...]] *file ...*

lint [**-abceghprvwzHFV**] [**-s** | **-t** | **-S**] [**-MD**] **-C** *library* [**-D** *name*[=def]] [**-U** *name*] [**-I** *directory*] [**-d** *directory*] [**-B** *directory*] [**-X** *id*[,id ...]] *file ...*

DESCRIPTION

lint attempts to detect features of the named C program files that are likely to be bugs, to be non-portable, or to be wasteful. It also performs stricter type checking than does the C compiler. The list of errors **lint** produces are enumerated in lint(7). **lint** runs the C preprocessor as its first phase, with the preprocessor symbol **lint** defined to allow certain questionable code to be altered or skipped by **lint**. Therefore, this symbol should be thought of as a reserved word for all code that is to be checked by **lint**.

Among the possible problems that are currently noted are unreachable statements, loops not entered at the top, variables declared and not used, and logical expressions with constant values. Function calls are checked for inconsistencies, such as calls to functions that return values in some places and not in others, functions called with varying numbers of arguments, function calls that pass arguments of a type other than the type the function expects to receive, functions whose values are not used, and calls to functions not returning values that use the non-existent return value of the function.

Filename arguments ending with .c are taken to be C source files. Filename arguments with names ending with .ln are taken to be the result of an earlier invocation of **lint**, with either the **-i**, **-o** or **-C** option in effect. The .ln files are analogous to the .o (object) files produced by cc(1) from .c files. **lint** also accepts special libraries specified with the **-l** option, which contain definitions of library routines and variables.

lint takes all the .c, .ln, and llib-l*library*.ln (lint library) files and processes them in command-line order. By default, **lint** appends the standard C lint library (llib-lc.ln) to the end of the list of files. When the **-i** option is used, the .ln files are ignored. Also, when the **-o** or **-i** options are used, the llib-l*library*.ln files are ignored. When the **-i** option is *omitted* the second pass of **lint** checks this list of files for mutual compatibility. At this point, if a complaint stems not from a given source file, but from one of its included files, the source filename will be printed followed by a question mark.

The special input file name "-" causes **lint** to take input from standard input (until end of file) and process it as if it were a .c file. If the **-i** flag is given and "-" is named as one of the input files, the **-o** flag must also be specified to provide an output file name.

Options

-a Report assignments of **long** values to variables that are not **long**.

-aa Additional to **-a**, report *all* assignments of integer values to other integer values which cause implicit narrowing conversion.

-b Report **break** statements that cannot be reached. This is not the default because, unfortunately, most lex(1) and many yacc(1) outputs produce many such complaints.

-c Complain about casts which have questionable portability.

-e Complain about unusual operations on **enum**-Types and combinations of **enum**- and **integer**-Types.

-g Don't print warnings for some extensions of `gcc`(1) to the C language. Currently these are nonconstant initializers in automatic aggregate initializations, arithmetic on pointer to void, trailing commas in enum declarations, C++ -style "//" comments, zero sized structures, subscripting of non-lvalue arrays, prototypes overriding old style function declarations and long long integer types. The **-g** flag also turns on the keywords **asm** and **inline** (alternative keywords with leading underscores for both **asm** and **inline** are always available).

-h Apply a number of heuristic tests to attempt to intuit bugs, improve style, and reduce waste.

-i Produce a `.ln` file for every `.c` file on the command line. These `.ln` files are the product of **lint**'s first pass only, and are not checked for compatibility between functions.

-n Do not check compatibility against the standard library.

-p Attempt to check portability of code to other dialects of C.

-r In case of redeclarations report the position of the previous declaration.

-s Strict ANSI C mode. Issue warnings and errors required by ANSI C. Also do not produce warnings for constructs which behave differently in traditional C and ANSI C. With the **-s** flag, `__STRICT_ANSI__` is a predefined preprocessor macro.

-S C9X mode. Currently not fully implemented.

-t Traditional C mode. `__STDC__` is not predefined in this mode. Warnings are printed for constructs not allowed in traditional C. Warnings for constructs which behave differently in traditional C and ANSI C are suppressed. Preprocessor macros describing the machine type (e.g. `sun3`) and machine architecture (e.g. `m68k`) are defined without leading and trailing underscores. The keywords **const**, **volatile** and **signed** are not available in traditional C mode (although the alternative keywords with leading underscores still are).

-u Do not complain about functions and external variables used and not defined, or defined and not used (this is suitable for running **lint** on a subset of files comprising part of a larger program).

-v Suppress complaints about unused arguments in functions.

-x Report variables referred to by **extern** declarations, but never used.

-z Do not complain about structures that are never defined (for example, using a structure pointer without knowing its contents).

-B*path* Path to use when looking for the lint1 and lint2 binaries. Defaults to `/usr/libexec`.

-C*library*
Create a **lint** library with the name `llib-l`*library*`.ln`. This library is built from all `.c` and `.ln` input files. After all global definitions of functions and variables in these files are written to the newly created library, **lint** checks all input files, including libraries specified with the **-l** option, for mutual compatibility.

-D*name*[=def]
Define *name* for `cpp`(1), as if by a `#define` directive. If no definition is given, *name* is defined as 1.

-I*directory*
Add *directory* to the list of directories in which to search for include files.

-d*directory*
Use *directory* instead of /usr/include as the default place to find include files.

-l*library*
Include the lint library `llib-l`*library*`.ln`.

-L*directory*
Search for lint libraries in *directory* and *directory*`/lint` before searching the standard place.

-F Print pathnames of files. **lint** normally prints the filename without the path.

-H If a complaint stems from an included file **lint** prints the name of the included file instead of the source file name followed by a question mark.

-MD Pass **-MD** to `cpp`(1) causing cpp to create files containing dependency information for each source file.

-o*outputfile*
Name the output file *outputfile*. The output file produced is the input that is given to **lint**'s second pass. The **-o** option simply saves this file in the named output file. If the **-i** option is also used the files are not checked for compatibility. To produce a `llib-l`*library*`.ln` without extraneous messages, use of the **-u** option is suggested. The **-v** option is useful if the source file(s) for the lint library are just external interfaces.

-U*name* Remove any initial definition of *name* for the preprocessor.

-V Print the command lines constructed by the controller program to run the C preprocessor and **lint**'s first and second pass.

-w Treat warnings as errors.

-X *id*[,id ...]
Suppress error messages identified by the list of ids. A list of messages and ids can be found in `lint`(7).

Input Grammar

lint's first pass reads standard C source files. **lint** recognizes the following C comments as commands.

`/* ARGSUSED`*n*` */`
Makes **lint** check only the first *n* arguments for usage; a missing *n* is taken to be 0 (this option acts like the **-v** option for the next function).

`/* BITFIELDTYPE */`
Suppress error messages about illegal bitfield types if the type is an integer type, and suppress non-portable bitfield type warnings.

`/* CONSTCOND */` or `/* CONSTANTCOND */` or `/* CONSTANTCONDITION */`
Suppress complaints about constant operands for the next expression.

`/* FALLTHRU */` or `/* FALLTHROUGH */`
Suppress complaints about fall through to a **case** or **default** labeled statement. This directive should be placed immediately preceding the label.

`/* LINTLIBRARY */`
At the beginning of a file, mark all functions and variables defined in this file as *used*. Also shut off complaints about unused function arguments.

`/* LINTED [`*comment*`]*/` or `/* NOSTRICT [`*comment*`]*/`
Suppresses any intra-file warning except those dealing with unused variables or functions. This directive should be placed on the line immediately preceding where the lint warning occurred.

`/* LONGLONG */`
Suppress complaints about use of long long integer types.

`/* NOTREACHED */`
At appropriate points, inhibit complaints about unreachable code. (This comment is typically placed just after calls to functions like `exit`(3)).

`/* PRINTFLIKE`*n*` */`
Makes **lint** check the first (*n*-1) arguments as usual. The *n*-th argument is interpreted as a **printf** format string that is used to check the remaining arguments.

`/* PROTOLIB`*n*` */`
Causes **lint** to treat function declaration prototypes as function definitions if *n* is non-zero. This directive can only be used in conjunction with the `/* LINTLIBRARY */` directive. If *n* is zero, function prototypes will be treated normally.

`/* SCANFLIKE`*n*` */`
Makes **lint** check the first (*n*-1) arguments as usual. The *n*-th argument is interpreted as a **scanf** format string that is used to check the remaining arguments.

`/* VARARGS`*n*` */`
Suppress the usual checking for variable numbers of arguments in the following function declaration. The data types of the first *n* arguments are checked; a missing *n* is taken to be 0.

The behavior of the **-i** and the **-o** options allows for incremental use of **lint** on a set of C source files. Generally, one invokes **lint** once for each source file with the **-i** option. Each of these invocations produces a `.ln` file that corresponds to the `.c` file, and prints all messages that are about just that source file. After all the source files have been separately run through **lint**, it is invoked once more (without the **-i** option), listing all the `.ln` files with the needed **-l***library* options. This will print all the inter-file inconsistencies. This scheme works well with `make`(1); it allows `make`(1) to be used to **lint** only the source files that have been modified since the last time the set of source files were **lint**ed.

ENVIRONMENT

`LIBDIR` The directory where the lint libraries specified by the **-l***library* option must exist. If this environment variable is undefined, then the default path `/usr/libdata/lint` will be used to search for the libraries.

`TMPDIR` Usually the path for temporary files can be redefined by setting this environment variable.

`CC` Location of the C compiler program. Defaults to `/usr/bin/cc`.

FILES

`/usr/libexec/lint[12]`	programs
`/usr/libdata/lint/llib-l*.ln`	various prebuilt lint libraries
`/tmp/lint*`	temporaries

SEE ALSO

`cc`(1), `cpp`(1), `make`(1), `lint`(7)

AUTHORS

Jochen Pohl

BUGS

The routines `exit`(3), `longjmp`(3) and other functions that do not return are not understood; this causes various incorrect diagnostics.

Static functions which are used only before their first extern declaration are reported as unused.

Libraries created by the **-o** option will, when used in later **lint** runs, cause certain errors that were reported when the libraries were created to be reported again, and cause line numbers and file names from the original source used to create those libraries to be reported in error messages. For these reasons, it is recommended to use the **-C** option to create lint libraries.

NAME

ln — make links

SYNOPSIS

ln [**-fhinsv**] *source_file* [*target_file*]
ln [**-fhinsv**] *source_file ... target_dir*

DESCRIPTION

The **ln** utility creates a new directory entry (linked file) which has the same modes as the original file. It is useful for maintaining multiple copies of a file in many places at once without using up storage for the "copies"; instead, a link "points" to the original copy. There are two types of links; hard links and symbolic links. How a link "points" to a file is one of the differences between a hard or symbolic link.

The options are as follows:

-f Unlink any already existing file, permitting the link to occur.

-h If the *target_file* or *target_dir* is a symbolic link, do not follow it. This is most useful with the **-f** option, to replace a symlink which may point to a directory.

-i Cause **ln** to write a prompt to standard error if the target file exists. If the response from the standard input begins with the character 'y' or 'Y', then unlink the target file so that the link may occur. Otherwise, do not attempt the link. (The **-i** option overrides any previous **-f** options.)

-n Same as **-h**, for compatibility with other **ln** implementations.

-s Create a symbolic link.

-v Cause **ln** to be verbose, showing files as they are processed.

By default **ln** makes *hard* links. A hard link to a file is indistinguishable from the original directory entry; any changes to a file are effective independent of the name used to reference the file. Hard links may not normally refer to directories and may not span file systems.

A symbolic link contains the name of the file to which it is linked. The referenced file is used when an open(2) operation is performed on the link. A stat(2) on a symbolic link will return the linked-to file; an lstat(2) must be done to obtain information about the link. The readlink(2) call may be used to read the contents of a symbolic link. Symbolic links may span file systems and may refer to directories.

Given one or two arguments, **ln** creates a link to an existing file *source_file*. If *target_file* is given, the link has that name; *target_file* may also be a directory in which to place the link; otherwise it is placed in the current directory. If only the directory is specified, the link will be made to the last component of *source_file*.

Given more than two arguments, **ln** makes links in *target_dir* to all the named source files. The links made will have the same name as the files being linked to.

SEE ALSO

link(2), lstat(2), readlink(2), stat(2), symlink(2), symlink(7)

STANDARDS

The **ln** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2").

The **-v** option is an extension to IEEE Std 1003.2-1992 ("POSIX.2").

HISTORY

A **ln** utility appeared in Version 6 AT&T UNIX.

NAME

loadfont — load and set font for the NetBSD/x68k console

SYNOPSIS

loadfont *fontfile*

DESCRIPTION

loadfont command reads font data from *fontfile* and replaces the x68k console font with it.

fontfile format is compatible with that of Human68k 'IOCS' font.

BUGS

Replacing kanji fonts is not supported, because current NetBSD/x68k kernel has no ability to use one other than in the ROM.

Font data in *fontfile* must be a full set of JIS X0201 (must contain hankaku-katakana part).

NAME

loadkmap — load and set the x68k console keyboard map

SYNOPSIS

loadkmap *keymapfile*

DESCRIPTION

loadkmap command reads *keymapfile* and replaces the x68k console keyboard map.

NAME

locale — get locale-specific information

SYNOPSIS

locale [**-a** | **-m**]
locale [**-ck**] [*keyword ...*]

DESCRIPTION

The **locale** utility is supposed to provide most locale specific information to the standard output.

When **locale** is invoked without arguments it will print out a summary of the current locale environment depending on environment variable settings and internal status.

When **locale** is invoked with arguments and no options specified it will print out *keyword*'s value determined using current locale settings.

The following options are available:

-a Write names of all available locales. While looking for locales **locale** will respect the PATH_LOCALE environment variable, and use it instead of the system default locale directory.

-c Write the category name for the selected keywords.

-k Write the name and value of the selected keywords.

-m Write names of all available charmaps.

IMPLEMENTATION DETAILS

Special (FreeBSD- / NetBSD-specific) keyword *list* can be used to retrieve a human readable list of available keywords.

DIAGNOSTICS

The **locale** utility exits 0 on success, and >0 if an error occurs.

STANDARDS

locale conforms to IEEE Std 1003.1-2001 ("POSIX.1").

HISTORY

locale first appeared in NetBSD 2.0.

AUTHORS

This implementation of **locale** was originally written by Alexey Zelkin ⟨phantom@FreeBSD.org⟩ for FreeBSD.

BUGS

Since NetBSD does not support *charmap*s in their *POSIX* meaning **locale** emulates the **-m** option via CODESETs listing of all available locales.

NAME

locate — find files

SYNOPSIS

locate [**-d** *dbpath*] *pattern*

DESCRIPTION

locate searches a database for all pathnames which match the specified *pattern*. The database is recomputed periodically, and contains the pathnames of all files which are publicly accessible.

Shell globbing and quoting characters (“∗”, “?”, “\”, “[” and “]”) may be used in *pattern*, although they will have to be escaped from the shell. Preceding any character with a backslash (“\”) eliminates any special meaning which it may have. The matching differs in that no characters must be matched explicitly, including slashes (“/”).

As a special case, a pattern containing no globbing characters (“foo”) is matched as though it were “∗foo∗”.

Options:

-d *dbpath*
: Sets the list of databases to search to *dbpath* which can name one or more database files separated by “:”, an empty component in the list represents the default database. The environment variable *LOCATE_PATH* has the same effect.

EXIT STATUS

locate exits with a 0 if a match is found, and >0 if no match is found or if another problem (such as a missing or corrupted database file) is encountered.

FILES

`/var/db/locate.database` Default database

SEE ALSO

`find`(1), `fnmatch`(3), `locate.conf`(5), `weekly.conf`(5), `locate.updatedb`(8)

Woods, James A., "Finding Files Fast", *;login*, 8:1, pp. 8-10, 1983.

HISTORY

The **locate** command appeared in 4.4BSD.

NAME

lock — reserve a terminal

SYNOPSIS

lock [**-np**] [**-t** *timeout*]

DESCRIPTION

lock requests a password from the user, reads it again for verification and then will normally not relinquish the terminal until the password is repeated. There are two other conditions under which it will terminate: it will timeout after some interval of time and it may be killed by someone with the appropriate permission.

Options:

-n No timeout is used. The terminal will be locked indefinitely or until current challenge is met.

-p A password is not requested, instead the user's current login password is used. If the user has an S/Key key, they may also use it to unlock the terminal. To do this the user should enter "s/key" at the unlock "Key:" prompt. The user will then be issued an S/Key challenge to which they may respond with a six-word S/Key one-time password.

-t *timeout*
The time limit (default 15 minutes) is changed to *timeout* minutes.

SEE ALSO

skey(1)

HISTORY

The **lock** command appeared in 3.0BSD.

NAME

logger — make entries in the system log

SYNOPSIS

logger [**-is**] [**-f** *file*] [**-p** *pri*] [**-t** *tag*] [*message ...*]

DESCRIPTION

logger provides a shell command interface to the syslog(3) system log module.

Options:

-i Log the process id of the logger process with each line.

-s Log the message to standard error, as well as the system log.

-f *file*
Log the specified file.

-p *pri* Enter the message with the specified priority. The priority may be specified numerically or as a ‘‘facility.level’’ pair. For example, ‘‘−p local3.info’’ logs the message(s) as *info*rmational level in the *local3* facility. The default is ‘‘user.notice.’’

-t *tag* Mark every line in the log with the specified *tag*.

message Write the message to log; if not specified, and the **-f** flag is not provided, standard input is logged.

The **logger** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

```
logger System rebooted

logger −p local0.notice −t HOSTIDM −f /dev/idmc
```

SEE ALSO

syslog(3), syslogd(8)

STANDARDS

The **logger** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

NAME

login — authenticate a user and start new session

SYNOPSIS

login [**-fp**] [**-a** *level*] [**-h** *hostname*] *[username]*

DESCRIPTION

This manual page documents the **login** program distributed with the Heimdal Kerberos 5 implementation, it may differ in important ways from your system version.

The **login** programs logs users into the system. It is intended to be run by system daemons like getty(8) or telnetd(8). If you are already logged in, but want to change to another user, you should use su(1).

A username can be given on the command line, else one will be prompted for.

A password is required to login, unless the **-f** option is given (indicating that the calling program has already done proper authentication). With **-f** the user will be logged in without further questions.

For password authentication Kerberos 5, Kerberos 4 (if compiled in), OTP (if compiled in) and local (/etc/passwd) passwords are supported. OTP will be used if the the user is registered to use it, and **login** is given the option **-a** otp. When using OTP, a challenge is shown to the user.

Further options are:

-a *string*
: Which authentication mode to use, the only supported value is currently “otp”.

-f
: Indicates that the user is already authenticated. This happens, for instance, when login is started by telnetd, and the user has proved authentic via Kerberos.

-h *hostname*
: Indicates which host the user is logging in from. This is passed from telnetd, and is entered into the login database.

-p
: This tells **login** to preserve all environment variables. If not given, only the TERM and TZ variables are preserved. It could be a security risk to pass random variables to **login** or the user shell, so the calling daemon should make sure it only passes “safe” variables.

The process of logging user in proceeds as follows.

First a check is made that logins are allowed at all. This usually means checking /etc/nologin. If it exists, and the user trying to login is not root, the contents is printed, and then login exits.

Then various system parameters are set up, like changing the owner of the tty to the user, setting up signals, setting the group list, and user and group id. Also various machine specific tasks are performed.

Next **login** changes to the users home directory, or if that fails, to /. The environment is setup, by adding some required variables (such as PATH), and also authentication related ones (such as KRB5CCNAME). If an environment file exists (/etc/environment), variables are set according to it.

If one or more login message files are configured, their contents is printed to the terminal.

If a login time command is configured, it is executed. A logout time command can also be configured, which makes **login** fork, and wait for the user shell to exit, and then run the command. This can be used to clean up user credentials.

Finally, the user’s shell is executed. If the user logging in is root, and root’s login shell does not exist, a default shell (usually /bin/sh) is also tried before giving up.

ENVIRONMENT

These environment variables are set by login (not including ones set by `/etc/environment`):

`PATH`	the default system path
`HOME`	the user's home directory (or possibly `/`)
`USER, LOGNAME`	both set to the username
`SHELL`	the user's shell
`TERM, TZ`	set to whatever is passed to **`login`**
`KRB5CCNAME`	if the password is verified via Kerberos 5, this will point to the credentials cache file
`KRBTKFILE`	if the password is verified via Kerberos 4, this will point to the ticket file

FILES

`/etc/environment`

Contains a set of environment variables that should be set in addition to the ones above. It should contain sh-style assignments like "VARIABLE=value". Note that they are not parsed the way a shell would. No variable expansion is performed, and all strings are literal, and quotation marks should not be used. Everything after a hash mark is considered a comment. The following are all different (the last will set the variable `BAR`, not `FOO`).

```
FOO=this is a string
FOO="this is a string"
BAR= FOO='this is a string'
```

`/etc/login.access`

See `login.access`(5).

`/etc/login.conf`

This is a termcap style configuration file, that contains various settings used by **`login`**. Currently only the "default" capability record is used. The possible capability strings include:

`environment`
: This is a comma separated list of environment files that are read in the order specified. If this is missing the default `/etc/environment` is used.

`login_program`
: This program will be executed just before the user's shell is started. It will be called without arguments.

`logout_program`
: This program will be executed just after the user's shell has terminated. It will be called without arguments. This program will be the parent process of the spawned shell.

`motd`
: A comma separated list of text files that will be printed to the user's terminal before starting the shell. The string `welcome` works similarly, but points to a single file.

`limits`
: Points to a file containing ulimit settings for various users. Syntax is inspired by what pam_limits uses, and the default is `/etc/security/limits.conf`.

`/etc/nologin`

If it exists, login is denied to all but root. The contents of this file is printed before login exits.

Other **`login`** programs typically print all sorts of information by default, such as last time you logged in, if you have mail, and system message files. This version of **`login`** does not, so there is no reason for `.hushlogin` files or similar. We feel that these tasks are best left to the user's shell, but the `login_program` facility allows for a shell independent solution, if that is desired.

EXAMPLES

A `login.conf` file could look like:

```
default:\
        :motd=/etc/motd,/etc/motd.local:\
        :limits=/etc/limits.conf:
```

The `limits.conf` file consists of a table with four whitespace separated fields. First field is a username or a groupname (prefixed with '@'), or '*'. Second field is 'soft', 'hard', or '-' (the last meaning both soft and hard). Third field is a limit name (such as 'cpu' or 'core'). Last field is the limit value (a number or '-' for unlimited). In the case of data sizes, the value is in kilobytes, and cputime is in minutes.

SEE ALSO

`su`(1), `login.access`(5), `getty`(8), `telnetd`(8)

AUTHORS

This login program was written for the Heimdal Kerberos 5 implementation. The login.access code was written by Wietse Venema.

NAME

login — authenticate users and set up their session environment

SYNOPSIS

login [**-Ffps**] [**-a** *address*] [**-h** *hostname*] [*user*]

DESCRIPTION

The **login** utility logs users (and pseudo-users) into the computer system.

If no user is specified, or if a user is specified and authentication of the user fails, **login** prompts for a user name. Authentication of users is done via passwords. If the user can be authenticated via S/Key, then the S/Key challenge is incorporated in the password prompt. The user then has the option of entering their Kerberos or normal password or the S/Key response. Neither will be echoed.

The options are as follows:

-a The **-a** option specifies the address of the host from which the connection was received. It is used by various daemons such as `telnetd`(8). This option may only be used by the super-user.

-F The **-F** option acts like the **-f** option, but also indicates to `login`(1) that it should attempt to rewrite an existing kerberos5 credentials cache (specified by the KRB5CCNAME environment variable) after dropping permissions to the user logging in. This flag is not supported under `pam`(8).

-f The **-f** option is used when a user name is specified to indicate that proper authentication has already been done and that no password need be requested. This option may only be used by the super-user or when an already logged in user is logging in as themselves.

-h The **-h** option specifies the host from which the connection was received. It is used by various daemons such as `telnetd`(8). This option may only be used by the super-user.

-p By default, **login** discards any previous environment. The **-p** option disables this behavior.

-s Require a secure authentication mechanism like Kerberos or S/Key to be used. This flag is not supported under `pam`(8).

If a user other than the superuser attempts to login while the file `/etc/nologin` exists, **login** displays its contents to the user and exits. This is used by `shutdown`(8) to prevent normal users from logging in when the system is about to go down.

Immediately after logging a user in, **login** displays the system copyright notice, the date and time the user last logged in, the message of the day as well as other information. If the file "`.hushlogin`" exists in the user's home directory, all of these messages are suppressed. This is to simplify logins for non-human users, such as `uucp`(1). **login** then records an entry in the `wtmp`(5) and `utmp`(5) files, executes site-specific login commands via the `ttyaction`(3) facility with an action of "login", and executes the user's command interpreter.

login enters information into the environment (see `environ`(7)) specifying the user's home directory (HOME), command interpreter (SHELL), search path (PATH), terminal type (TERM) and user name (both LOGNAME and USER).

The standard shells, `csh`(1) and `sh`(1), do not fork before executing the **login** utility.

FILES

`/etc/motd`	message-of-the-day
`/etc/nologin`	disallows non-superuser logins

`/var/run/utmp`	list of current logins
`/var/log/lastlog`	last login account records
`/var/log/wtmp`	login account records
`/var/mail/user`	system mailboxes
`.hushlogin`	makes login quieter

SEE ALSO

`chpass`(1), `passwd`(1), `rlogin`(1), `skey`(1), `getpass`(3), `ttyaction`(3), `login.conf`(5), `passwd.conf`(5), `utmp`(5), `environ`(7), `kerberos`(8), `pam`(8)

HISTORY

A **`login`** appeared in Version 6 AT&T UNIX.

TRADEMARKS AND PATENTS

S/Key is a trademark of Bellcore.

NAME

logname — display user's login name

SYNOPSIS

logname

DESCRIPTION

The **logname** utility writes the user's login name to standard output followed by a newline.

The **logname** utility explicitly ignores the LOGNAME and USER environment variables because the environment cannot be trusted.

The **logname** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

who(1), whoami(1), getlogin(2)

STANDARDS

The **logname** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2").

HISTORY

The **logname** command appeared in 4.4BSD.

NAME

look — display lines beginning with a given string

SYNOPSIS

look [**-df**] [**-t** *termchar*] *string* [*file*]

DESCRIPTION

The **look** utility displays any lines in *file* which contain *string* as a prefix. As **look** performs a binary search, the lines in *file* must be sorted.

If *file* is not specified, the file /usr/share/dict/words is used, only alphanumeric characters are compared and the case of alphabetic characters is ignored.

Options:

-d Dictionary character set and order, i.e. only alphanumeric characters are compared.

-f Ignore the case of alphabetic characters.

-t Specify a string termination character, i.e. only the characters in *string* up to and including the first occurrence of *termchar* are compared.

The **look** utility exits 0 if one or more lines were found and displayed, 1 if no lines were found, and >1 if an error occurred.

FILES

/usr/share/dict/words the dictionary

SEE ALSO

grep(1), sort(1)

COMPATIBILITY

The original manual page stated that tabs and blank characters participated in comparisons when the **-d** option was specified. This was incorrect and the current man page matches the historic implementation.

HISTORY

look appeared in Version 7 AT&T UNIX.

NAME

lorder — list dependencies for object files

SYNOPSIS

lorder *file ...*

DESCRIPTION

The **lorder** utility uses nm(1) to determine interdependencies in the list of object files specified on the command line. **lorder** outputs a list of file names where the first file contains a symbol which is defined by the second file.

The output is normally used with tsort(1) when a library is created to determine the optimum ordering of the object modules so that all references may be resolved in a single pass of the loader.

EXAMPLES

```
ar cr library.a ‘lorder ${OBJS} | tsort‘
```

SEE ALSO

ar(1), ld(1), nm(1), ranlib(1), tsort(1)

HISTORY

An **lorder** utility appeared in Version 7 AT&T UNIX.

NAME

lp — front-end to the print spooler

SYNOPSIS

lp [**-cs**] [**-o** *option*] [**-d** *printer*] [**-n** *num*] [*file . . .*]

DESCRIPTION

lp is a front-end to the print spooler as required by the IEEE Std 1003.2 (“POSIX.2”) specification. It effectively invokes lpr(1) with the proper set of arguments. It generally prints the named files on the destination printer.

The following options are available:

-c Make the **lp** command exit only after further access to any of the input files is no longer required. The application can then safely delete or modify the files without affecting the output operation.

-d *dest*
Specify a particular printer. If no **-d** is provided on the command line, the contents of the environment variables LPDEST or PRINTER (with this precedence) are taken as the destination printer.

-n *num*
Specify that *num* copies of each of the named files shall be printed.

-s Silent operation.

-o Printer specific options. Not supported, provided only as a compatibility option for SVR4.

ENVIRONMENT

As described above, the variables LPDEST and PRINTER are examined to select the destination printer.

SEE ALSO

lpr(1)

STANDARDS

The **lp** command is expected to comply with the IEEE Std 1003.2 (“POSIX.2”) specification.

AUTHORS

This implementation of the **lp** command has been written by Jörg Wunsch .

BUGS

The IEEE Std 1003.2 (“POSIX.2”) specification does not provide any means to print non-text files. It rather requires the files to be printed to be text files limited to reasonable line lengths and printable characters.

NAME

lpq — spool queue examination program

SYNOPSIS

lpq [**-al**] [**-P** *printer*] [**-w** *maxwait*] [*job# ...*] [*user ...*]

DESCRIPTION

lpq examines the spooling area used by lpd(8) for printing files on the line printer, and reports the status of the specified jobs or all jobs associated with a user. **lpq** invoked without any arguments reports on any jobs currently in the queue.

Options:

-P Specify a particular printer, otherwise the default line printer is used (or the value of the PRINTER variable in the environment). All other arguments supplied are interpreted as user names or job numbers to filter out only those jobs of interest.

-l Information about each of the files comprising the job entry is printed. Normally, only as much information as will fit on one line is displayed.

-a Report on the local queues for all printers, rather than just the specified printer.

-w *maxwait*
Specify the maximum time to wait in seconds for remote responses. The default is 300 seconds or 5 minutes.

For each job submitted (i.e. invocation of lpr(1)) **lpq** reports the user's name, current rank in the queue, the names of files comprising the job, the job identifier (a number which may be supplied to lprm(1) for removing a specific job), and the total size in bytes. Job ordering is dependent on the algorithm used to scan the spooling directory and is supposed to be FIFO (First in First Out). File names comprising a job may be unavailable (when lpr(1) is used as a sink in a pipeline) in which case the file is indicated as ''(standard input)''.

If **lpq** warns that there is no daemon present (i.e. due to some malfunction), the lpc(8) command can be used to restart the printer daemon.

ENVIRONMENT

If the following environment variable exists, it is used by **lpq**:

PRINTER Specifies an alternative default printer.

FILES

/etc/printcap	To determine printer characteristics.
/var/spool/output/*	The spooling directory, as determined from printcap.
/var/spool/output/*/cf*	Control files specifying jobs.
/var/spool/output/*/lock	The lock file to obtain the currently active job.

DIAGNOSTICS

Unable to open various files. The lock file being malformed. Garbage files when there is no daemon active, but files in the spooling directory.

SEE ALSO

lpr(1), lprm(1), lpc(8), lpd(8)

HISTORY

lpq appeared in 3BSD.

BUGS

Due to the dynamic nature of the information in the spooling directory **lpq** may report unreliably. Output formatting is sensitive to the line length of the terminal; this can results in widely spaced columns.

NAME

lpr — off line print

SYNOPSIS

lpr [**-cdfghlmnopqRrstv**][**-1234** *font*][**-#** *num*][**-C** *class*][**-i** *numcols*]
[**-J** *job*][**-P** *printer*][**-T** *title*][**-U** *user*][**-w** *num*][*name ...*]

DESCRIPTION

lpr uses a spooling daemon to print the named files when facilities become available. If no names appear, the standard input is assumed.

The following single letter options are used to notify the line printer spooler that the files are not standard text files. The spooling daemon will use the appropriate filters to print the data accordingly.

-c The files are assumed to contain data produced by **cifplot**

-d The files are assumed to contain data from *tex* (DVI format from Stanford).

-f Use a filter which interprets the first character of each line as a standard FORTRAN carriage control character.

-g The files are assumed to contain standard plot data as produced by the **plot** routines (see also **plot** for the filters used by the printer spooler).

-l Use a filter which allows control characters to be printed and suppresses page breaks.

-n The files are assumed to contain data from *ditroff* (device independent troff).

-o The files are assumed to be in postscript format.

-p Use pr(1) to format the files (equivalent to **print**).

-t The files are assumed to contain data from troff(1) (cat phototypesetter commands).

-v The files are assumed to contain a raster image for devices like the Benson Varian.

These options apply to the handling of the print job:

-h Suppress the printing of the burst page.

-m Send mail upon completion.

-P Force output to a specific printer. Normally, the default printer is used (site dependent), or the value of the environment variable PRINTER is used.

-q Queue the print job but do not start the spooling daemon.

-r Remove the file upon completion of spooling or upon completion of printing (with the **-s** option).

-s Use symbolic links. Usually files are copied to the spool directory. The **-s** option will use symlink(2) to link data files rather than trying to copy them so large files can be printed. This means the files should not be modified or removed until they have been printed.

Normally **lpr** works silently except for diagnostic messages. The following option changes this behavior.

-R Writes a message to standard output containing the unique number which is used to identify this job. This number can be used to cancel (see lprm(1)) or find the status (see lpq(1)) of the job.

The remaining options apply to copies, the page display, and headers:

-#*num* The quantity *num* is the number of copies desired of each file named. For example,

```
lpr −#3 foo.c bar.c more.c
```

would result in 3 copies of the file foo.c, followed by 3 copies of the file bar.c, etc. On the other hand,

```
cat foo.c bar.c more.c | lpr −#3
```

will give three copies of the concatenation of the files. Often a site will disable this feature to encourage use of a photocopier instead.

−[**1234**] *font*
Specifies a *font* to be mounted on font position *i*. The daemon will construct a `.railmag` file referencing the font pathname.

−C *class*
Job classification to use on the burst page. For example,

```
lpr −C EECS foo.c
```

causes the system name (the name returned by `hostname`(1)) to be replaced on the burst page by EECS, and the file foo.c to be printed.

−i −numcols
The output is indented by (*numcols*).

−J *job*
Job name to print on the burst page. Normally, the first file's name is used.

−T *title*
Title name for `pr`(1), instead of the file name.

−U *user*
User name to print on the burst page, also for accounting purposes. This option is only honored if the real user-id is daemon (or that specified in the printcap file instead of daemon), and is intended for those instances where print filters wish to requeue jobs.

−w*num* Uses *num* as the page width for `pr`(1).

ENVIRONMENT

If the following environment variable exists, it is used by **lpr**:

`PRINTER` Specifies an alternative default printer.

FILES

`/etc/passwd`	Personal identification.
`/etc/printcap`	Printer capabilities data base.
`/usr/sbin/lpd`	Line printer daemons.
`/var/spool/output/*`	Directories used for spooling.
`/var/spool/output/*/cf*`	Daemon control files.
`/var/spool/output/*/df*`	Data files specified in "cf" files.
`/var/spool/output/*/tf*`	Temporary copies of "cf" files.

DIAGNOSTICS

If you try to spool too large a file, it will be truncated. **lpr** will object to printing binary files. If a user other than root prints a file and spooling is disabled, **lpr** will print a message saying so and will not put jobs in the queue. If a connection to `lpd`(8) on the local machine cannot be made, **lpr** will say that the daemon cannot be started. Diagnostics may be printed in the daemon's log file regarding missing spool files by `lpd`(8).

SEE ALSO

`lpq`(1), `lprm`(1), `pr`(1), `symlink`(2), `printcap`(5), `lpc`(8), `lpd`(8)

HISTORY

The **`lpr`** command appeared in 3BSD.

BUGS

Fonts for `troff`(1) and **`tex`** reside on the host with the printer. It is currently not possible to use local font libraries.

NAME

lprm — remove jobs from the line printer spooling queue

SYNOPSIS

lprm [**−**] [**−P** *printer*] [**−w** *maxwait*] [*job# ...*] [*user ...*]

DESCRIPTION

lprm will remove a job, or jobs, from a printer's spool queue. Since the spooling directory is protected from users, using **lprm** is normally the only method by which a user may remove a job. The owner of a job is determined by the user's login name and host name on the machine where the lpr(1) command was invoked.

Options and arguments:

−P*printer*
: Specify the queue associated with a specific *printer* (otherwise the default printer is used).

−w *maxwait*
: Specify the maximum time to wait in seconds for remote responses. The default is 300 seconds or 5 minutes.

−
: If a single '**−**' is given, **lprm** will remove all jobs which a user owns. If the super-user employs this flag, the spool queue will be emptied entirely.

user
: Causes **lprm** to attempt to remove any jobs queued belonging to that user (or users). This form of invoking **lprm** is useful only to the super-user.

job #
: A user may dequeue an individual job by specifying its job number. This number may be obtained from the lpq(1) program, e.g.

```
% lpq −l

1st:ken                         [job #013ucbarpa]
        (standard input)        100 bytes
% lprm 13
```

If neither arguments or options are given, **lprm** will delete the currently active job if it is owned by the user who invoked **lprm**.

lprm announces the names of any files it removes and is silent if there are no jobs in the queue which match the request list.

lprm will kill off an active daemon, if necessary, before removing any spooling files. If a daemon is killed, a new one is automatically restarted upon completion of file removals.

ENVIRONMENT

If the following environment variable exists, it is used by **lprm**.

PRINTER
: If the environment variable PRINTER exists, and a printer has not been specified with the **−P** option, the default printer is assumed from PRINTER.

FILES

/etc/printcap	Printer characteristics file.
/var/spool/output/∗	Spooling directories.
/var/spool/output/∗/lock	Lock file used to obtain the pid of the current daemon and the job number of the currently active job.

DIAGNOSTICS

"Permission denied" if the user tries to remove files other than his own.

SEE ALSO

`lpq`(1), `lpr`(1), `lpd`(8)

HISTORY

The **`lprm`** command appeared in 3.0BSD.

BUGS

Since there are race conditions possible in the update of the lock file, the currently active job may be incorrectly identified.

NAME

lptest — generate lineprinter ripple pattern

SYNOPSIS

lptest [*length* [*count*]]

DESCRIPTION

lptest writes the traditional "ripple test" pattern on standard output. In 96 lines, this pattern will print all 96 printable ASCII characters in each position. While originally created to test printers, it is quite useful for testing terminals, driving terminal ports for debugging purposes, or any other task where a quick supply of random data is needed.

The *length* argument specifies the output line length if the default length of 79 is inappropriate.

The *count* argument specifies the number of output lines to be generated if the default count of 200 is inappropriate. Note that if *count* is to be specified, *length* must be also be specified.

HISTORY

lptest appeared in 4.3BSD.

NAME

ls — list directory contents

SYNOPSIS

ls [**-AaBbCcdFfghikLlmnopqRrSsTtuWwx1**] [*file ...*]

DESCRIPTION

For each operand that names a *file* of a type other than directory, **ls** displays its name as well as any requested, associated information. For each operand that names a *file* of type directory, **ls** displays the names of files contained within that directory, as well as any requested, associated information.

If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical order.

The following options are available:

-A List all entries except for '**.**' and '**..**'. Always set for the super-user.

-a Include directory entries whose names begin with a dot ('.').

-B Force printing of non-graphic characters in file names as \xxx, where xxx is the numeric value of the character in octal.

-b As **-B**, but use C escape codes whenever possible.

-C Force multi-column output; this is the default when output is to a terminal.

-c Use time when file status was last changed, instead of time of last modification of the file for sorting (**-t**) or printing (**-l**).

-d Directories are listed as plain files (not searched recursively) and symbolic links in the argument list are not indirected through.

-F Display a slash ('/') immediately after each pathname that is a directory, an asterisk ('*') after each that is executable, an at sign ('@') after each symbolic link, a percent sign ('%') after each whiteout, an equal sign ('=') after each socket, and a vertical bar ('|') after each that is a FIFO.

-f Output is not sorted.

-g The same as **-l**, except that the owner is not printed.

-h Modifies the **-s** and **-l** options, causing the sizes to be reported in bytes displayed in a human readable format. Overrides **-k**.

-i For each file, print the file's file serial number (inode number).

-k Modifies the **-s** option, causing the sizes to be reported in kilobytes. The rightmost of the **-k** and **-h** flags overrides the previous flag. See also **-h**.

-L For each file, if it's a link, evaluate file information and file type of the referenced file and not the link itself; however still print the link name, unless used with **-l**, for example.

-l (The lowercase letter "ell"). List in long format. (See below.) A total sum for all the file sizes is output on a line before the long listing.

-m Stream output format; list files across the page, separated by commas.

-n Display the user and group IDs numerically rather than converting to a user or group name in a long (**-l**) output.

-o Include the file flags in a long (**-l**) output. If no file flags are set, "-" is displayed. (See `chflags`(1) for a list of possible flags and their meanings.)

-p Display a slash ('/') immediately after each pathname that is a directory.

-q Force printing of non-printable characters in file names as the character '?'; this is the default when output is to a terminal.

-R Recursively list subdirectories encountered.

-r Reverse the order of the sort to get reverse lexicographical order or the smallest or oldest entries first.

-S Sort by size, largest file first.

-s Display the number of file system blocks actually used by each file, in units of 512 bytes or `BLOCKSIZE` (see **ENVIRONMENT**) where partial units are rounded up to the next integer value. If the output is to a terminal, a total sum for all the file sizes is output on a line before the listing.

-T When used with the **-l** (the lowercase letter "ell") option, display complete time information for the file, including month, day, hour, minute, second, and year.

-t Sort by time modified (most recently modified first) before sorting the operands by lexicographical order.

-u Use time of last access, instead of last modification of the file for sorting (**-t**) or printing (**-l**).

-W Display whiteouts when scanning directories.

-w Force raw printing of non-printable characters. This is the default when output is not to a terminal.

-x Multi-column output sorted across the page rather than down the page.

-1 (The numeric digit "one"). Force output to be one entry per line. This is the default when output is not to a terminal.

The **-B**, **-b**, **-w**, and **-q** options all override each other; the last one specified determines the format used for non-printable characters.

The **-1**, **-C**, **-g**, **-l**, **-m**, and **-x** options all override each other; the last one specified determines the format used with the exception that if both **-l** and **-g** are specified, **-l** will always override **-g**, even if **-g** was specified last.

The **-c** and **-u** options override each other; the last one specified determines the file time used.

By default, **ls** lists one entry per line to standard output; the exceptions are to terminals or when the **-C** or **-m** options are specified.

File information is displayed with one or more ⟨blank⟩ separating the information associated with the **-i**, **-s**, and **-l** options.

The Long Format

If the **-l** option is given, the following information is displayed for each file:

file mode
number of links
owner name
group name
file flags (if **-o** given)

number of bytes in the file
abbreviated month file was last modified
day-of-month file was last modified
hour and minute file was last modified
pathname

In addition, for each directory whose contents are displayed, the total number of 512-byte blocks used by the files in the directory is displayed on a line by itself immediately before the information for the files in the directory.

If the owner or group names are not a known user or group name, or the **-n** option is given, the numeric ID's are displayed.

If the file is a character special or block special file, the major and minor device numbers for the file are displayed in the size field. If the file is a symbolic link the pathname of the linked-to file is preceded by "->".

The file mode printed under the **-l** option consists of the entry type, owner permissions, group permissions, and other permissions. The entry type character describes the type of file, as follows:

a Archive state 1.
A Archive state 2.
b Block special file.
c Character special file.
d Directory.
l Symbolic link.
s Socket link.
p FIFO.
w Whiteout.
- Regular file.

The next three fields are three characters each: owner permissions, group permissions, and other permissions. Each field has three character positions:

1. If **r**, the file is readable; if **-**, it is not readable.
2. If **w**, the file is writable; if **-**, it is not writable.
3. The first of the following that applies:

   S If in the owner permissions, the file is not executable and set-user-ID mode is set. If in the group permissions, the file is not executable and set-group-ID mode is set.

   s If in the owner permissions, the file is executable and set-user-ID mode is set. If in the group permissions, the file is executable and setgroup-ID mode is set.

   x The file is executable or the directory is searchable.

   - The file is neither readable, writable, executable, nor set-user-ID nor set-group-ID mode, nor sticky. (See below.)

   These next two apply only to the third character in the last group (other permissions).

   T The sticky bit is set (mode `1000`), but not execute or search permission. (See `chmod`(1) or `sticky`(7).)

   t The sticky bit is set (mode `1000`), and is searchable or executable. (See `chmod`(1) or `sticky`(7).)

The number of bytes displayed for a directory is a function of the number of `dirent`(5) structures in the directory, not all of which may be allocated to any existing file.

EXIT STATUS

The **ls** utility exits 0 on success, and >0 if an error occurs.

ENVIRONMENT

The following environment variables affect the execution of **ls**:

BLOCKSIZE If the environment variable BLOCKSIZE is set, and the **-h** and **-k** options are not specified, the block counts (see **-s**) will be displayed in units of that size block.

COLUMNS If this variable contains a string representing a decimal integer, it is used as the column position width for displaying multiple-text-column output. The **ls** utility calculates how many pathname text columns to display based on the width provided. (See **-C**.)

TZ The timezone to use when displaying dates. See environ(7) for more information.

COMPATIBILITY

The group field is now automatically included in the long listing for files in order to be compatible with the IEEE Std 1003.2 ("POSIX.2") specification.

SEE ALSO

chflags(1), chmod(1), stat(2), getbsize(3), dir(5), sticky(7), symlink(7)

STANDARDS

The **ls** utility is expected to be a superset of the IEEE Std 1003.2 ("POSIX.2") specification.

HISTORY

An **ls** utility appeared in Version 5 AT&T UNIX.

NAME

m4 — macro language processor

SYNOPSIS

m4 [**-Pg**] [**-D***name*[=*value*]] [**-I** *dirname*] [**-U***name*] [**-d** *flags*] [**-o** *trfile*] [**-t** *name*]

DESCRIPTION

The **m4** utility is a macro processor that can be used as a front end to any language (e.g., C, ratfor, fortran, lex, and yacc). **m4** reads from the standard input and writes the processed text to the standard output.

Macro calls have the form name(argument1[, argument2, ..., argumentN]).

There cannot be any space following the macro name and the open parenthesis (‘(’). If the macro name is not followed by an open parenthesis it is processed with no arguments.

Macro names consist of a leading alphabetic or underscore possibly followed by alphanumeric or underscore characters, e.g., valid macro names match the pattern “[a-zA-Z_][a-zA-Z0-9_]∗”.

In arguments to macros, leading unquoted space, tab, and newline (‘\n’) characters are ignored. To quote strings, use left and right single quotes (e.g., ‘ this is a string with a leading space’). You can change the quote characters with the **changequote** built-in macro.

Most built-ins don’t make any sense without arguments, and hence are not recognized as special when not followed by an open parenthesis.

The options are as follows:

-D*name*[=*value*] Define the symbol *name* to have some value (or NULL).

-I *dirname* Add directory *dirname* to the include path.

-P Prefixes all **m4** builtin macros with the string “m4_”. This changes the macro names “dnl” to “m4_dnl”, “index” to “m4_index”, and so forth.

-U*name* Undefine the symbol *name*.

-d *flags* Set trace flags. *flags* may hold the following:

- *a* print macro arguments.
- *c* print macro expansion over several lines.
- *e* print result of macro expansion.
- *f* print filename location.
- *l* print line number.
- *q* quote arguments and expansion with the current quotes.
- *t* start with all macros traced.
- *x* number macro expansions.
- *V* turn on all options.

By default, trace is set to "eq".

-g Activate GNU-m4 compatibility mode. In this mode, changequote with two empty parameters deactivates quotes, translit handles simple character ranges (e.g., a-z), regular expressions mimic emacs behavior, and the number of diversions is unlimited.

-o *trfile* Specify the tracing output file for **-t**; tracing defaults to printing to stderr.

-t *macro* Turn tracing on for *macro*.

SYNTAX

m4 provides the following built-in macros. They may be redefined, losing their original meaning. Return values are null unless otherwise stated.

builtin Calls a built-in by its name, overriding possible redefinitions.

changecom Change the start and end comment sequences. The default is the pound sign ('#') and the newline character. With no arguments comments are turned off. The maximum length for a comment marker is five characters.

changequote Defines the quote symbols to be the first and second arguments. The symbols may be up to five characters long. If no arguments are given it restores the default open and close single quotes.

decr Decrements the argument by 1. The argument must be a valid numeric string.

define Define a new macro named by the first argument to have the value of the second argument. Each occurrence of "$n" (where *n* is 0 through 9) is replaced by the *n*'th argument. "$0" is the name of the calling macro. Undefined arguments are replaced by a null string. "$#" is replaced by the number of arguments; "$*" is replaced by all arguments comma separated; "$@" is the same as "$*" but all arguments are quoted against further expansion.

defn Returns the quoted definition for each argument. This can be used to rename macro definitions (even for built-in macros).

divert There are 10 output queues (numbered 0-9). At the end of processing **m4** concatenates all the queues in numerical order to produce the final output. Initially the output queue is 0. The divert macro allows you to select a new output queue (an invalid argument passed to divert causes output to be discarded).

divnum Returns the current output queue number.

dnl Discard input characters up to and including the next newline.

dumpdef Prints the names and definitions for the named items, or for everything if no arguments are passed.

errprint Prints the first argument on the standard error output stream.

esyscmd Pass its first argument to a shell and returns the shell's standard output. Note that the shell shares its standard input and standard error with **m4**.

eval Computes the first argument as an arithmetic expression using 32-bit arithmetic. Operators are the standard C ternary, arithmetic, logical, shift, relational, bitwise, and parentheses operators. You can specify octal, decimal, and hexadecimal numbers as in C. The second argument (if any) specifies the radix for the result and the third argument (if any) specifies the minimum number of digits in the result.

expr This is an alias for **eval**.

ifdef If the macro named by the first argument is defined then return the second argument, otherwise the third. If there is no third argument, the value is NULL. The word "unix" is predefined.

ifelse If the first argument matches the second argument then **ifelse** returns the third argument. If the match fails the three arguments are discarded and the next three arguments are used until there is zero or one arguments left, either this last argument or NULL is returned if no other matches were found.

include Returns the contents of the file specified in the first argument. If the file is not found as is, look through the include path: first the directories specified with **-I** on the command line, then the environment variable M4PATH, as a colon-separated list of directories. Include aborts with an error message if the file cannot be included.

incr Increments the argument by 1. The argument must be a valid numeric string.

index Returns the index of the second argument in the first argument (e.g., **index(the quick brown fox jumped, fox)** returns 16). If the second argument is not found index returns −1.

indir Indirectly calls the macro whose name is passed as the first arguments, with the remaining arguments passed as first, ... arguments.

len Returns the number of characters in the first argument. Extra arguments are ignored.

m4exit Immediately exits with the return value specified by the first argument, 0 if none.

m4wrap Allows you to define what happens at the final EOF, usually for cleanup purposes (e.g., **m4wrap("cleanup(tempfile)")** causes the macro cleanup to be invoked after all other processing is done).

maketemp Translates the string "XXXXX" in the first argument with the current process ID leaving other characters alone. This can be used to create unique temporary file names.

paste Includes the contents of the file specified by the first argument without any macro processing. Aborts with an error message if the file cannot be included.

patsubst Substitutes a regular expression in a string with a replacement string. Usual substitution patterns apply: an ampersand ('&') is replaced by the string matching the regular expression. The string "\#", where '#' is a digit, is replaced by the corresponding back-reference.

popdef Restores the **pushdef**ed definition for each argument.

pushdef Takes the same arguments as **define**, but it saves the definition on a stack for later retrieval by **popdef**.

regexp Finds a regular expression in a string. If no further arguments are given, it returns the first match position or −1 if no match. If a third argument is provided, it returns the replacement string, with sub-patterns replaced.

shift Returns all but the first argument, the remaining arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.

sinclude Similar to **include**, except it ignores any errors.

spaste Similar to **paste**, except it ignores any errors.

substr Returns a substring of the first argument starting at the offset specified by the second argument and the length specified by the third argument. If no third argument is present it returns the rest of the string.

syscmd Passes the first argument to the shell. Nothing is returned.

sysval Returns the return value from the last **syscmd**.

traceon Enables tracing of macro expansions for the given arguments, or for all macros if no argument is given.

traceoff Disables tracing of macro expansions for the given arguments, or for all macros if no argument is given.

translit Transliterate the characters in the first argument from the set given by the second argument to the set given by the third. You cannot use tr(1) style abbreviations.

undefine Removes the definition for the macro specified by the first argument.

undivert Flushes the named output queues (or all queues if no arguments).

unix A pre-defined macro for testing the OS platform.

__line__ Returns the current file's line number.

__file__ Returns the current file's name.

STANDARDS

The **m4** utility is expected to be a superset of the X/Open Commands and Utilities Issue 5 ("XCU5") specification, and includes some extensions also present in GNU **m4**.

The **-s** option (cpp(1)'s #line directives) is currently not supported. Flags **-I**, **-d**, **-t** are non-standard.

The output format of tracing and of **dumpdef** are not specified in any standard, are likely to change and should not be relied upon. The current format of tracing is closely modelled on GNU **m4**, to allow **autoconf** to work.

For portability, one should not use the macros **builtin**, **esycmd**, **expr**, **indir**, **paste**, **patsubst**, **regexp**, **spaste**, **unix**, **__line__**, **__file__**.

All builtins do expand without arguments in many other **m4**.

AUTHORS

Ozan Yigit <oz@sis.yorku.ca> and Richard A. O'Keefe (ok@cs.rmit.edu.au). GNU-m4 compatibility extensions by Marc Espie <espie@cvs.openbsd.org>.

NAME

machine — print machine type

SYNOPSIS

machine

DESCRIPTION

The **machine** command displays the machine type. It is equivalent to `uname -m`.

SEE ALSO

make(1), uname(1)

HISTORY

A **machine** command appeared in 4.3BSD−Reno.

NAME

mail, **mailx**, **Mail** — send and receive mail

SYNOPSIS

mail [**-EIinv**] [**-a** *file*] [**-b** *bcc-addr*] [**-c** *cc-addr*] [**-s** *subject*] *to-addr ...*
[*sendmail-flags*]
mail [**-EIiNnv**] [**-H**[colon-modifier]] **-f** [*name*]
mail [**-EIiNnv**] [**-H**[colon-modifier]] [**-u** *user*]

DESCRIPTION

mail is an intelligent mail processing system, which has a command syntax reminiscent of ed(1) with lines replaced by messages.

-a Attach *file* to the message.

-b Send blind carbon copies to *list*. List should be a comma-separated list of names.

-c Send carbon copies to *list* of users.

-E Don't send messages with an empty body. This is useful for piping errors from cron scripts.

-f Read in the contents of your *mbox* (or the specified file) for processing; when you **quit**, **mail** writes undeleted messages back to this file.

-H Print the header summaries and exit. The optional colon-modifier string must begin with a '**:**' and be followed by one or more of the characters described in the **Specifying messages** section below. E.g., "**mail -H**:n" will display just new message headers.

-I Forces mail to run in interactive mode even when input isn't a terminal. In particular, the **~** special character when sending mail is only active in interactive mode.

-i Ignore tty interrupt signals. This is particularly useful when using **mail** on noisy phone lines.

-N Inhibits the initial display of message headers when reading mail or editing a mail folder.

-n Inhibits reading /etc/mail.rc upon startup.

-s Specify subject on command line (only the first argument after the **-s** flag is used as a subject; be careful to quote subjects containing spaces.)

-u Is equivalent to:

```
mail -f /var/mail/user
```

-v Verbose mode. The details of delivery are displayed on the user's terminal.

Sending mail

To send a message to one or more people, **mail** can be invoked with arguments which are the names of people to whom the mail will be sent. You are then expected to type in your message, followed by a control-D at the beginning of a line.

Any flags following the list of recipients, will be passed, together with their arguments, directly to sendmail(1). For example to change your From address to somebody@somewhere.net you can specify:

```
mail recipient -f somebody@somewhere.net
```

To prevent multiple copies of a message being sent to the same address, duplicate addresses (after alias expansion) are removed from the *bcc-addr*, *cc-addr*, and *to-addr* lists. In addition, addresses on the *cc-addr* and *to-addr* lists are removed if they occur on the *bcc-addr* list and addresses on the

cc-addr list are removed if they occur on the *to-addr* list. If the *to-addr* list is empty after these deletions, most systems will insert the line "To: undisclosed recipients:;".

The section below **Replying to or originating mail**, describes some features of **mail** available to help you compose your letter.

Reading mail

In normal usage **mail** is given no arguments and checks your mail out of the post office, then prints out a one line header of each message found. The current message is initially the first message (numbered 1) and can be printed using the **print** command (which can be abbreviated **p**). You can move among the messages much as you move between lines in ed(1), with the commands **+** and **−** moving backwards and forwards, and simple numbers.

Disposing of mail

After examining a message you can **delete** (**d**) the message or **reply** (**r**) to it. Deletion causes the **mail** program to forget about the message. This is not irreversible; the message can be **undeleted** (**u**) by giving its number, or the **mail** session can be aborted by giving the **exit** (**x**) command. Deleted messages will, however, usually disappear never to be seen again.

Specifying messages

Many commands (e.g., **delete**, **from**, and **print**) accept a list of messages as an argument. Messages may be specified by their message number, by a range of messages, or by a pattern string matching certain fields in the header as described below. These message "specs" may be combined by the usual binary boolean operations '&', '|', and '^', which denote, respectively, a logical "and", "or", and "xor". Logical expressions may be grouped with parentheses '(' and ')' and negated with '!'. If the binary operator is missing between two message specs, it is assumed to be a '|'. This is for simplicity, backwards compatibility, and also to to facilitate using the '|' symbol to denote a pipe. (See *enable-pipes*.)

Besides the obvious (base10) message numbers, the characters '^', '-', '.', '+', and '$' denote, respectively, the first message, the message before the "dot" (the current message), the "dot" message, the message following the "dot", and the last message.

A "message range" consists of two message numbers separated by a '-'. A '*' denotes all messages and is equivalent to ^-$.

A pattern is a string (not beginning with any of the above special characters). If it does not begin with a '/', it is compared with the senders address. If it begins with a '/', and *searchheaders* is not defined, the remainder of the string is compared with the subject field. (See *searchheaders* for searching other header fields or the message body.) If *regex-search* is not defined, then the comparison is a simple case insensitive substring match. (See *regex-search* for regular expression matches.)

A list of messages may be restricted by a "colon-modifier" string, i.e., a ':' followed by one or more of the characters:

```
d       deleted
e       edited
m       mboxed
n       new
o       old
p       preserved
r       read
s       saved
t       tagged
u       unread and not new
!       invert the meaning of the colon-modifiers
```

If there are no address specifications other than colon-modifiers, the colon-modifiers apply to all messages. Thus "`from netbsd :n`" would display the headers of all new messages with `netbsd` in the sender's address, while "`from :!r`" and "`from :nu`" would both display all new and unread messages. Multiple colon-modifiers may be specified and a single '`:`' with no letters following indicates the colon-modifier from the preceding command.

For example:

```
from 1 12 3-5
```

would display the headers from messages 1, 3, 4, 5, and 12.

```
from anon & ( /foo | /bar )
```

would display all headers that had `anon` in the sender's address and either `foo` or `bar` in the subject line.

Generally, commands cannot select messages that are not displayed, such as deleted or hidden messages, the exception being the **`undelete`** command.

Replying to or originating mail

You can use the **`reply`** command to set up a response to a message, sending it back to the person who it was from. Text you then type in, up to an end-of-file, defines the contents of the message. While you are composing a message, **`mail`** treats lines beginning with the character `~` specially. For instance, typing **`~m`** (alone on a line) will place a copy of the current message into the response right shifting it by a tabstop (see *indentprefix* variable, below). Other escapes will set up subject fields, add and delete recipients to the message, and allow you to escape to an editor to revise the message or to a shell to run some commands. (These options are given in the summary below.)

Ending a mail processing session

You can end a **`mail`** session with the **`quit`** (**`q`**) command. Messages which have been examined go to your *`mbox`* file unless they have been deleted in which case they are discarded. Unexamined messages go back to the post office. (See the **`-f`** option above).

Personal and systemwide distribution lists

It is also possible to create a personal distribution lists so that, for instance, you can send mail to "`cohorts`" and have it go to a group of people. Such lists can be defined by placing a line like

```
alias cohorts bill ozalp jkf mark kridle@ucbcory
```

in the file `.mailrc` in your home directory. The current list of such aliases can be displayed with the **`alias`** command in **`mail`**. System wide distribution lists can be created by editing `/etc/mail/aliases`, see `aliases`(5) and `sendmail`(1); these are kept in a different syntax. In mail you send, personal aliases will be expanded in mail sent to others so that they will be able to **`reply`** to the recipients. System wide **`aliases`** are not expanded when the mail is sent, but any reply returned to the machine will have the system wide alias expanded as all mail goes through `sendmail`(1).

Network mail (ARPA, UUCP, Berknet)

See `mailaddr`(7) for a description of network addresses.

`mail` has a number of options which can be set in the `.mailrc` file to alter its behavior; thus "`set askcc`" enables the *`askcc`* feature. (These options are summarized below.)

SUMMARY

(Adapted from the "Mail Reference Manual")

Each command is typed on a line by itself, and may take arguments following the command word. The command need not be typed in its entirety − the first command which matches the typed prefix is used. For commands which take message lists as arguments, if no message list is given, then the next message forward which satisfies the command's requirements is used. If there are no messages forward of the current message, the search proceeds backwards, and if there are no good messages at all, **mail** types "`No applicable messages`" and aborts the command.

! Executes the shell (see `sh`(1) and `csh`(1)) command which follows.

− Print out the preceding message. If given a numeric argument *n*, goes to the *n*'th previous message and prints it.

= With no argument, it displays the current message number. Otherwise, set the current message number to its first argument.

? Prints a brief summary of commands.

| Pipe the current message body through the shell (see `sh`(1) and `csh`(1)) command which follows.

Detach Like **detach** but also saves MIME parts that don't have a filename associated with them. For the unnamed parts, a filename is suggested containing the message and part numbers, and the subtype.

More (**M**) Like **more** but also prints out ignored header fields.

Page (**Pa**) A synonym for **More**.

Print (**P**) Like **print** but also prints out ignored header fields. See also **print**, **more**, **page**, **type**, **view**, **ignore**, and **retain**.

Reply (**R**) Reply to originator. Does not reply to other recipients of the original message. (See **reply**.)

Save (**S**) Same as **save** except that all header fields are saved ignoring the **saveignore** or **saveretain** lists.

Type (**T**) Identical to the **Print** command.

View (**V**) Like **Print** but has the opposite MIME decoding behavior. (See the *mime-decode-message* variable.)

alias (**a**) With no arguments, prints out all currently-defined aliases. With one argument, prints out that alias. With more than one argument, creates a new alias or changes an old one.

alternates
(**alt**) The **alternates** command is useful if you have accounts on several machines. It can be used to inform **mail** that the listed addresses are really you. When you **reply** to messages, **mail** will not send a copy of the message to any of the addresses listed on the **alternates** list. If the **alternates** command is given with no argument, the current set of alternative names is displayed.

bounce Takes a list of messages and prompts for an address to bounce the messages to. If no message is specified, the current message is used. All the original header fields are preserved except for the `Delivered-To`, `X-Original-To` and `Status` fields. The new '`To`' field contains the bounce address(es) plus any addresses in the old '`To`' field minus the user's local address and any on the alternates list. (See the **alternates** command.)

chdir (**c**) Changes the user's working directory to that specified, if given. If no directory is given, then changes to the user's login directory.

copy (**co**) The **copy** command does the same thing that **save** does, except that it does not mark the messages it is used on for deletion when you quit.

deldups
Delete duplicate messages based on their `Message-Id` field, keeping the first one in the current sort order. This can be useful with replies to a mailing list that are also CCed to a subscriber. (The same thing can also be accomplished with the threading and tagging commands.)

delete (**d**) Takes a list of messages as an argument and marks them all as deleted. Deleted messages will not be saved in *mbox*, nor will they be available for most other commands.

detach Takes a message list followed by a target directory as arguments, decodes each MIME part in the message list, and saves it in the target directory. If the message list is empty, use the current message. If the directory is not specified, use the directory specified by *mime-detach-dir* variable and, if that is empty, default to the directory **mail** was started in. For each MIME part in the message list, the filename is displayed for confirmation or changes. If an empty name is entered, the part is skipped. If the filename already exists, the user will be prompted before overwriting it. (See the *mime-detach-batch* and *mime-detach-overwrite* variables to change this behavior.) Only MIME parts with an associated filename in the `Content-Type` or `Content-Disposition` fields are decoded. (See **Detach** to detach all parts.) The MIME extension hooks and character set conversion are ignored.

dp (also **dt**) Deletes the current message and prints the next message. If there is no next message, **mail** says "`at EOF`".

down Go down one level in the thread. If given a message number, it descends the thread below that message, otherwise it descends from the current message (dot).

edit (**e**) Takes a list of messages and points the text editor at each one in turn. On return from the editor, the message is read back in.

else Switch the command execution condition set by the previous **if**, **ifdef**, or **ifndef** command.

endif Terminate an **if**, **ifdef**, or **ifndef** command.

exit (**ex** or **x**) Effects an immediate return to the Shell without modifying the user's system mailbox, his *mbox* file, or his edit file in **-f**.

expose Expose the thread structure so all messages appear in header listings. (See *hide* for the inverse.) The default header prompt will indent each header line one space for each level in the threading. The "`%?* ?`" format string does this.

file (**fi**) The same as **folder**.

flatten
For each message number in the argument list, or the current thread if no message list is given, promote all exposed children to the same thread level.

folders
List the names of the folders in your folder directory.

folder (**fo**) The **folder** command switches to a new mail file or folder. With no arguments, it tells you which file you are currently reading. If you give it an argument, it will write out changes (such as deletions) you have made in the current file and read in the new file. Some special conventions are recognized for the name. '#' means the previous file, '%' means your system mailbox, `%user` means user's system mailbox, '&' means your *mbox* file, and `+file` means a file in your folder directory.

forward
Takes a list of messages and prompts for an address (or addresses) to forward each message to. If no message list is specified, the current message is used. The mail editor is run for each message allowing the user to enter a message that will precede the forward message. The message is sent as a multipart/mixed MIME encoded message. All header fields except the `Status` field are included.

from (**f**) Takes a list of messages and prints their message headers.

headers
(**h**) Lists the current range of headers, which is an 18−message group. If a '+' argument is given, then the next 18−message group is printed, and if a '−' argument is given, the previous 18−message group is printed.

help A synonym for **?**

hide Collapse the threads so that only the head of each thread is shown, hiding the subthreads. (See *expose* for the inverse.)

hidetags
Restrict the display to untagged messages. In threaded mode, subthreads that connect directly to an untagged message are also displayed, including tagged messages in the connecting chain.

hidethreads
The same as *hide*.

hold (**ho**, also **preserve**) Takes a message list and marks each message therein to be saved in the user's system mailbox instead of in *mbox*. Does not override the **delete** command.

if Execute commands that follow depending on the operating mode. The current supported modes are `receiving`, `sending`, and `headersonly`. For example, one use might be something like:

```
if headersonly
  set header-format="%P%Q%3i %-21.20f %m/%d %R %3K \"%q\""
else
  set header-format="%P%Q%?& ?%3i %-21.20f %a %b %e %R %3K/%-5O \"%q\""
endif
```

ifdef Execute commands that follow if the specified variable is defined. Note: This includes environment variables.

ifndef Execute commands that follow if the specified variable is not defined.

ignore Add the list of header fields named to the *ignored list*. Header fields in the ignore list are not printed on your terminal when you print a message. This command is very handy for suppression of certain machine-generated header fields. The **Type** and **Print** commands can be used to print a message in its entirety, including ignored fields. If **ignore** is executed with no arguments, it lists the current set of ignored fields.

inc Incorporate any new messages that have arrived while mail is being read. The new messages are added to the end of the message list, and the current message is reset to be the first new mail message. This does not renumber the existing message list, nor does it cause any changes made so far to be saved.

invtags
Invert the tags on a list of messages or the current message if none are given. Note: this will not affect any currently deleted messages.

mail (**m**) Takes as argument login names and distribution group names and sends mail to those people.

mbox Indicate that a list of messages be sent to **mbox** in your home directory when you quit. This is the default action for messages if you do *not* have the **hold** option set.

mkread (**mk**) Takes a message list and marks each message as having been read.

more (**mo**) Takes a message list and invokes the pager on that list.

next (**n**, like **+** or CR) Goes to the next message in sequence and types it. With an argument list, types the next matching message.

page (**pa**) A synonym for **more**.

preserve
(**pre**) A synonym for **hold**.

print (**p**) Takes a message list and types out each message on the user's terminal.

quit (**q**) Terminates the session, saving all undeleted, unsaved messages in the user's *mbox* file in his login directory, preserving all messages marked with **hold** or **preserve** or never referenced in his system mailbox, and removing all other messages from his system mailbox. If new mail has arrived during the session, the message "`You have new mail`" is given. If given while editing a mailbox file with the **-f** flag, then the edit file is rewritten. A return to the Shell is effected, unless the rewrite of edit file fails, in which case the user can escape with the **exit** command.

reply (**r**) Takes a message list and sends mail to the sender and all recipients of the specified message. The default message must not be deleted. (See the **Reply** command and the *Replyall* variable.)

respond
A synonym for **reply**.

retain Add the list of header fields named to the *retained list*. Only the header fields in the retained list are shown on your terminal when you print a message. All other header fields are suppressed. The **Type** and **Print** commands can be used to print a message in its entirety. If **retain** is executed with no arguments, it lists the current set of retained fields. **Retain** overwrides **save**.

reverse
Reverse the order of the messages in at the current thread level. This is completely equivalent to "`sort !`".

save (**s**) Takes a message list and a filename and appends each message in turn to the end of the file. The filename in quotes, followed by the line count and character count is echoed on the user's terminal.

set (**se**) With no arguments, prints all variable values. Otherwise, sets option. Arguments are of the form *option=value* (no space before or after =) or *option*. Quotation marks may be placed around any part of the assignment statement to quote blanks or tabs, i.e. "`set indentprefix="->"`"

saveignore
Saveignore is to **save** what **ignore** is to **print** and **type**. Header fields thus marked are filtered out when saving a message by **save** or when automatically saving to *mbox*.

saveretain
Saveretain is to **save** what **retain** is to **print** and **type**. Header fields thus marked are the only ones saved with a message when saving by **save** or when automatically saving to *mbox*. **Saveretain** overrides **saveignore**.

shell (**sh**) Invokes an interactive version of the shell.

show (**sho**) Takes a list of variables and prints out their values in the form *option=value*. If the list is empty, all variable values are shown.

showtags
Display all current messages, tagged or not, unless they are in a hidden thread.

showthreads
The same as *expose*.

size Takes a message list and prints out the size in characters of each message.

smopts Takes an "address-spec" followed by the sendmail flags that should be used when sending mail to an address that matches that "address-spec". If no sendmail flags are specified, then list the sendmail flags in effect for the "address-spec". If the "address-spec" is also omitted, then list all **smopts** settings. The "address-spec" may be an alias, address, domain (beginning with a '@'), or subdomain (beginning with a '.'). If mail is sent to multiple users, the sendmail flags are used only if the flags are the same for each recipients. If *smopts-verify* is set, then you will be asked to verify the sendmail flags (if there are any) before the mail is sent. Address matching is case insensitive and done from most specific to least.

For example if you have:

```
smopts mylist -F "List Maintainer"
smopts @NetBSD.org -f anon@somewhere.net -F "Anon Ymous"
smopts friend@NetBSD.org ""
```

then mail sent to any of the addresses that the `mylist` alias expands to would have the sender's name set to `List Maintainer`. Mail sent to anyone at NetBSD.org other than `friend@NetBSD.org` would look like it was sent from `anon@somewhere.net` by `Anon Ymous`. Mail sent to `friend@NetBSD.org` would not have any sendmail flags set (unless they are set by the **~h** escape).

sort With no argument, **sort** does nothing. Otherwise it will sort based on the header field name given as an argument. A few names are special:

```
blines          sort based on the number of body lines.
hlines          sort on the number of header lines.
tlines          sort on the total number of lines.
size            sort on the message size
sday            sent day (ignores the hour/min/sec)
rday            received day (ignores the hour/min/sec)
sdate           sent date
rdate           received date
subject         sort on the subject, ignoring "Re:" prefixes.
from            sort on the sender's address.
```

The check for these special names is case sensitive while the header field name comparisons are case insensitive, so changing the case on any of these special names will sort based on the header field ignoring the special keyword.

There are also three modifiers which may preceed the argument:

```
!       reverse the sorting order.
^       case insensitive sorting.
-       skin the field (removing RFC 822 comments and
        keep the address).
```

The same keywords and modifiers also apply to threading. (See the **thread** command.)

Note: **sort** has no effect on the threading, sorting only on the heads of the threads if threads exist.

source The **source** command reads commands from a file.

tag Tag a list of messages or the current message if none are given. In hidden thread mode, the entire thread will be tagged, i.e., **tag** is recursive

tagbelow
Tag all messages of the current thread below the level of the current message (dot) or the supplied message number if given.

thread By default this threads the current message list based on the `In-Reply-To` and `References` header fields (intended for this purpose by RFC 2822). If given an argument, it will thread on that header field name instead. The same field keywords and modifiers recognized by the sort command are also recognized here. Display of the threads is controlled by the **hide** and **expose** commands; navigation of threads is done with the **down**, **up**, and **tset** commands.

If *recursive-commands* is defined, many commands (e.g., **print**) act on the entire thread (when it is hidden), otherwise they act on just the current message.

Note: the `In-Reply-To` and `Reference` header fields are necessary to do threading correctly. This version of **mail** now emits these header fields when replying.

top Takes a message list and prints the top few lines of each. The number of lines printed is controlled by the variable **toplines** and defaults to five.

tset Set the current thread (thread set) so that the supplied message number (or the current message if no argument is given) is at the top level of the current thread.

type (**t**) A synonym for **print**.

unalias
Takes a list of names defined by **alias** commands and discards the remembered groups of users. The group names no longer have any significance.

undelete
(**u**) Takes a message list and marks each message as **not** being deleted.

unread (**unr**) Takes a message list and marks each message as *not* having been read.

unset Takes a list of option names and discards their remembered values; the inverse of **set**.

unsmopts
Takes a list of "address-specs" defined by **smopts** commands and discards them from the smopts datebase.

untag Untag a list of messages or the current message if none are given. Like the **tag** command, **untag** is recursive on hidden threads.

unthread
Undo all threading and sorting, restoring the original display order, i.e., the order in the mail file.

up Go up one level in the thread. This also takes an optional (positive) argument to go up multiple levels in the thread.

view (**vie**) Like **print** but has the opposite MIME decoding behavior. (See the *mime-decode-message* variable.)

visual (**v**) Takes a message list and invokes the display editor on each message.

write (**w**) Similar to **save**, except that *only* the message body (*without* the header) is saved. Extremely useful for such tasks as sending and receiving source program text over the message system.

xit (**x**) A synonym for **exit**.

z **mail** presents message headers in windowfuls as described under the **headers** command. You can move **mail**'s attention forward to the next window with the **z** command. Also, you can move to the previous window by using **z-**.

Tilde/Escapes

Here is a summary of the tilde escapes, which are used when composing messages to perform special functions. Tilde escapes are only recognized at the beginning of lines. The name "*tilde escape*" is somewhat of a misnomer since the actual escape character can be set by the option **escape**.

~!*command*
Execute the indicated shell command, then return to the message.

~@ [*filelist*]
Add the files in the white-space delimited *filelist* to the attachment list. If *filelist* is omitted, edit the attachment list, possibly appending to it: For each file in the list the user is prompted to change its attachment data. Changing the filename to empty will delete it from the list. Upon reaching the end of the attachment list, the user is prompted for additional files to attach until an empty filename is given. Filenames containing white-space can only be added in this "edit" mode.

~a Inserts the autograph string from the sign= option into the message.

~A Inserts the autograph string from the Sign= option into the message.

~b*name ...*
Add the given names to the list of carbon copy recipients but do not make the names visible in the Cc: line ("blind" carbon copy).

~c*name ...*
Add the given names to the list of carbon copy recipients.

~d Read the file "`dead.letter`" from your home directory into the message.

~e Invoke the text editor on the message collected so far. After the editing session is finished, you may continue appending text to the message.

~f*messages*
Read the named messages into the message being sent. If no messages are specified, read in the current message. Message headers currently being ignored (by the **ignore** or **retain** command) are not included.

~F*messages*
Identical to **~f**, except all message headers are included.

~h Edit the message header fields, and the options passed to sendmail (the `Smopts`), by typing each one in turn and allowing the user to append text to the end or modify the field by using the current terminal erase and kill characters. If `editline`(3) support is included, then that line editor is used.

~i*string*
Inserts the value of the named option into the text of the message.

~m*messages*
Read the named messages into the message being sent, indented by a tab or by the value of *indentprefix*. If no messages are specified, read the current message. Message headers cur-

rently being ignored (by the **ignore** or **retain** command) are not included.

~M*messages*
Identical to ~m, except all message headers are included.

~p Print out the message collected so far, prefaced by the message header fields.

~q Abort the message being sent, copying the message to "dead.letter" in your home directory if **save** is set.

~x Exits as with ~q, except the message is not saved in "dead.letter".

~r*filename*

~<*filename*
Reads the named file into the message. If the argument begins with '!', the rest of the string is taken as an arbitrary system command and is executed, with the standard output inserted into the message.

~s*string*
Cause the named string to become the current subject field.

~t*name ...*
Add the given names to the direct recipient list.

~v Invoke an alternative editor (defined by the VISUAL option) on the message collected so far. Usually, the alternative editor will be a screen editor. After you quit the editor, you may resume appending text to the end of your message.

~w*filename*
Write the message onto the named file.

~|*command*
Pipe the message body through the command as a filter. If the command gives no output or terminates abnormally, retain the original text of the message. The command fmt(1) is often used as **command** to rejustify the message.

~:*mail-command*
Execute the given mail command. Not all commands, however, are allowed.

~~*string*
Insert the string of text in the message prefaced by a single ~. If you have changed the escape character, then you should double that character in order to send it.

Mail Options

Options are controlled via **set** and **unset** commands. Options may be either binary, in which case it is only significant to see whether they are set or not; or string, in which case the actual value is of interest. The binary options include the following:

append Causes messages saved in *mbox* to be appended to the end rather than prepended. This should always be set (perhaps in /etc/mail.rc).

ask, *asksub*
Causes **mail** to prompt you for the subject of each message you send. If you respond with simply a newline, no subject field will be sent.

askcc Causes you to be prompted for additional carbon copy recipients at the end of each message. Responding with a newline indicates your satisfaction with the current list.

autoinc
Causes new mail to be automatically incorporated when it arrives. Setting this is similar to issuing the **inc** command at each prompt, except that the current message is not reset when new mail arrives.

askbcc Causes you to be prompted for additional blind carbon copy recipients at the end of each message. Responding with a newline indicates your satisfaction with the current list.

autoprint
Causes the **delete** command to behave like **dp** − thus, after deleting a message, the next one will be typed automatically.

crt If *crt* is set, then the PAGER will be used for the **print**, **Print**, **type**, and **Type** commands. Normally these commands do not envoke the pager.

debug Setting the binary option *debug* is the same as specifying **−d** on the command line and causes **mail** to output all sorts of information useful for debugging **mail**.

dot The binary option *dot* causes **mail** to interpret a period alone on a line as the terminator of a message you are sending.

enable-pipes
If defined, the output of most commands can be piped into a shell command or redirected to a file. The pipe/redirection is signaled by the first occurrence of a '|' or '>' character that is not in a quoted string or in a parenthetical group. This character terminates the mail command line and the remaining string is passed to the shell. For example, assuming normal headers, something like

```
from john@ | fgrep -i ' "Re:' | wc
```

could be used to count how may replies were made by senders with john@ in their address and

```
from john@ >> /tmp/john
```

would append all the headers from such senders to /tmp/john.

Note: With piping enabled, you cannot use the '|' as a logical "or" operator outside of a parenthetical group. This should not be a problem as it is the default logical operator. (See the **Specifying messages** section.)

hold This option is used to hold messages in the system mailbox by default.

ignore Causes interrupt signals from your terminal to be ignored and echoed as @'s.

metoo Usually, when a group is expanded that contains the sender, the sender is removed from the expansion. Setting this option causes the sender to be included in the group.

mime-attach-list
If set, the command line flag **−a** will accept a whitespace delimited list of files. Otherwise, its argument is interpreted as a single filename. Warning: If enabled, care must be taken to properly quote files that contain whitespace, both from the shell and from this second expansion done by **mail**.

mime-decode-header
If set, decode the headers along with the body when *mime-decode-message* is set. The header decode follows the same rules as the body (see *mime-decode-message*).

mime-decode-insert
When inserting a message into the mail buffer (**~f** or **~F**), the text inserted will be decoded according to the settings of the *mime-decode-message* and *mime-decode-header* variables.

mime-decode-message
If set, the **More**, **more**, **Page**, **page**, **Print**, **print**, **Type**, and **type** commands will display decoded the MIME messages. Otherwise, they display the undecoded message. Recall that the **View** and **view** commands always have the opposite MIME decoding behavior from these commands.

mime-decode-quote
When quoting a message into the mail buffer (**~m** or **~M**), the text inserted will be decoded according to the settings of the *mime-decode-message* and *mime-decode-header* variables.

mime-detach-batch
If set, the detach command does not prompt for anything (unless *mime-detach-overwrite* is set to `ask`), overwriting target files depending on the setting of *mime-detach-overwrite*.

noheader
Setting the option *noheader* is the same as giving the **-N** flag on the command line.

nosave Normally, when you abort a message with two RUBOUT (erase or delete) **mail** copies the partial letter to the file "`dead.letter`" in your home directory. Setting the binary option *nosave* prevents this.

quiet Suppresses the printing of the version when first invoked.

recursive-commands
When defined, and threading is in effect, the following commands operate on the entire thread (if it is "hidden") rather than just the top displayed message of the thread:

```
More Page Print Type View more page print type view
top
Save copy save write
Detach detach
delete dp dt
undelete
hold preserve
mbox mkread touch unread
tag untag invtags
```

If not defined, or if the threads are "exposed", commands behave exactly as they do in non-threaded mode, i.e., each operates on individual messages.

Replyall
Reverses the sense of **reply** and **Reply** commands.

searchheaders
If this option is set, then a message-list specifier in the form "/x:y" will expand to all messages containing the substring "y" in the header field "x". The string search is case insensitive. If "x" is omitted, it will default to the `Subject` header field. If "y" is omitted, only those messages that contain the field "x" will be matched. The three forms "/from:y", "/to:y", and "/body:y" are special. The first will match all messages which contain the substring "y" in the headline (which is added locally at receipt time and begins with "From "). The second will match all messages containing the substring "y" in the '`To`', '`Cc`', or `Bcc` header fields. The third will match all messages which contain the substring "y" in a line of the message body. The check for "from", "to", and "body" is case sensitive, so that "/From:y" and "/To:y" can be used to search the `From` and '`To`' fields, respectively. (See also *regex-search*.)

smopts-verify
Verify the sendmail options used on outgoing mail if they were obtained from a *smopts* match. This has no effect if there are no sendmail flags or if the flags were set by the **~h** escape.

verbose
Setting the option *verbose* is the same as using the **-v** flag on the command line. When mail runs in verbose mode, the actual delivery of messages is displayed on the user's terminal.

Option String Values

EDITOR Pathname of the text editor to use in the **edit** command and **~e** escape. If not defined, then a default editor is used.

LISTER Pathname of the directory lister to use in the **folders** command. Default is /bin/ls.

PAGER Pathname of the program to use in the **more** command or when **crt** variable is set. The default paginator more(1) is used if this option is not defined.

SHELL Pathname of the shell to use in the **!** command and the **~!** escape. A default shell is used if this option is not defined.

VISUAL Pathname of the text editor to use in the **visual** command and **~v** escape.

el-completion-keys
A comma or space delimited list of keys to do editline(3) completion. For example **set el-completion-keys=^I,^D** will bind completion to both the tab and CTRL-D keys. (Requires editline(3) support.)

el-editor The line editing mode: must be emacs or 'vi'. If unset, editing is not enabled. (Requires editline(3) support.)

el-history-size
The number of lines of history to remember. If unset, history is not enable. (Requires editline(3) support.)

escape If defined, the first character of this option gives the character to use in the place of ~ to denote escapes.

folder The name of the directory to use for storing folders of messages. If this name begins with a '/', **mail** considers it to be an absolute pathname; otherwise, the folder directory is found relative to your home directory.

header-format
If set, use this format string when displaying headers in command mode. The format string supports the following conversions in addition to those of strftime(3):

%?key? The header field with name key. Note: if key[0] is '-', ignore the '-' and extract the address portion of the field (i.e., "skin" the field).

*%?*string?*
If the depth is *n*, substitute string *n* times. This is intended to be used when displaying an "exposed thread".

%?&string?
Like *%?*string?*, but uses the depth relative to the current depth rather than the absolute depth.

%J The number of header lines in the message.

%K The number of body lines in the message.

%L The total number of lines in the message.
%N The sender's full name (as in the `From` or `Sender` fields).
%O The message size.
%P The current "dot" (`>`) message.
%Q The message status flag.
%Z The time zone name (if it exists).
%f The email address of sender.
%i The message number.
%n The sender's login name (taken from the address).
%q The subject.
%t The total number of messages.
%z The GMT offset (if found).

If the format string begins with `%??` then the date will be extracted from the headline. Otherwise it will be extracted from the `Date` header falling back to the headline if that extraction fails. For example, the default format is:

```
set header-format="%??%P%Q%?* ?%3i %-21.20f %a %b %e %R %3K/%-5O \"%q\""
```

Note 1: The message status flag '`%Q`' will display the single character '`+`' for the parent of a subthread. This will be overwritten by a '`T`', '`E`', '`*`', '`P`', '`U`', '`N`', '`M`' indicating, respectively, a tagged, modified, saved, preserved, unread, new, or modified message, in that order with the last matching condition being the one displayed. In the case of hidden threads, the entire subthread is searched and the letters above will be displayed in lower case if the property is that of a hidden child with the case '`*`' being displayed as '`&`'.

Note 2: *%n* and *%t* as used by `strftime`(3) were redundant with \t and \n, respectively, so nothing is lost using them here.

ignoreeof An option related to *dot* is *ignoreeof* which makes **mail** refuse to accept a `control-D` as the end of a message. If given a numeric argument *n*, a `control-D` will be accepted after *n* tries. *Ignoreeof* also applies to **mail** command mode.

indentpreamble
If set, this format string will be inserted before quoting a message (**~m** or **~M**). The format syntax is the same as for *header-format*. For example, the following:

```
set indentpreamble=
  "On %b %e %T, %Y %z (%Z), %n (%.50N) wrote:\n-- Subject: %.65q\n"
```

would insert something like

```
On Oct 27 11:00:07, 2006 -0400 (EDT), anon (Anon Ymous) wrote:
-- Subject: suggestions for mail(1)
```

before the quoted message.

indentprefix
String used by the **~m** and **~M** tilde escapes for indenting messages, in place of the normal tab character ('^I'). Be sure to quote the value if it contains spaces or tabs.

indentpostscript
If set, this format string will be inserted after quoting a message (**~m** or **~M**). The format syntax is the same as for *header-format*. For example, the following:

```
set indentpostscript="-- End of excerpt from %.50N"
```

would insert something like

```
-- End of excerpt from Anon Ymous
```

after the quoted message.

mime-body-TYPE-SUBTYPE
MIME-hook for the body of a MIME block of `Content-Type: TYPE/SUBTYPE`. (See **MIME Enhancements** below.)

mime-charset
Convert `Content-type: text` messages to this character set or `us-ascii` if the value is empty. If unset, no character set conversion is done.

mime-detach-dir
The directory to detach files to if the *detach* command is given no arguments. (See *detach*.)

mime-detach-overwrite
This controls overwriting of existing files by the detach command. If set to `ask` the user will be prompted before overwriting a file. If set to `yes`, or to the empty string, existing target files will be overwritten. If set to '`no`', no target files will be overwritten.

mime-encode-message
If set, encode the body of the message as required. Typically, this is just an issue of whether `quoted-printable` encoding is used or not. If it has a value, then use it to determine the encoding type. Allowed values are `7bit`, `8bit`, `binary`, `quoted-printable`, or `base64`.

mime-head-TYPE-SUBTYPE
MIME-hook for the header of a MIME block of `Content-Type: TYPE/SUBTYPE`. (See **MIME Enhancements** below.)

mime-hook-TYPE-SUBTYPE
MIME-hook for MIME block of `Content-Type: TYPE/SUBTYPE`. (See **MIME Enhancements** below.)

`MBOX` The name of the *mbox* file. It can be the name of a folder. The default is "`mbox`" in the user's home directory.

prompt If defined, it specifies the prompt to use when in command mode. Otherwise, the default '`&`' is used. The format syntax is the same as for *header-format*.

record If defined, gives the pathname of the file used to record all outgoing mail. If not defined, then outgoing mail is not so saved.

regex-search
If set, regular expression searches are used, instead of simple case insensitive substring matches, when determining message lists by searching sender names, subjects, or header fields (if *searchheaders* is defined); see the **Specifying message** section. The value of the variable is a space or comma delimited list of options. Valid options are `icase` to do case insensitive searches, `extended` to use extended (rather than basic) regular expressions, and `nospec` to turn off all special character meanings and do literal string searches. Note that `extended` and `nospec` are not compatable (see `regcomp`(3)).

ReplyAsRecipient
This is used when replying to email (see the **`reply`** or **`Reply`** commands). It is useful if you have multiple email addresses and wish to ensure that replies respect them. If set, grab the email address(es) from the '`To`' field of the message being replied to. If there is only one

such address, and if it does not match any address in the value of *ReplyAsRecipient* (a comma or space delimited list of addresses, possibly empty), then use this address in the From field of the reply. This is accomplished by passing the address to sendmail(1) with the **-f** option. Note: the sendmail options can be edited with the **~h** escape. (See also the **smopts** command.)

toplines If defined, gives the number of lines of a message to be printed out with the **top** command; normally, the first five lines are printed.

MIME Enhancements

A MIME message is (recursively) divided into a series of MIME parts that can be thought of as sub-messages, each with a header and body. When MIME support is enabled (by setting *mime-decode-message*), **mail** splits a message into a series of its smallest MIME parts and processes those parts as if they were messages themselves, passing the header and body through a pipeline of the form:

```
mail -> MIME-decoder -> MIME-hook -> pager -> screen
```

The *MIME-decoder* decodes base64 or quoted-printable encoding and is enabled according to the Content-Transfer-Encoding of the part. The *MIME-hook* is an external program to further process the part (see below). The *pager* is the program that pages the message (see PAGER). Any of these intermediate pipe stages may be missing and/or different for the head and body of each MIME part. Certain Content-Types may disable the entire pipeline (e.g., application/octet).

The *MIME-hook* stage is not present unless one of the following variables is set:

```
mime-hook-TYPE-SUBTYPE    applies to the entire MIME part
mime-head-TYPE-SUBTYPE    applies to the header of the MIME part
mime-body-TYPE-SUBTYPE    applies to the body of the MIME part
```

where TYPE and SUBTYPE are the Content-Type type and subtype (respectively) of the MIME part to which the hook applies. If the "-SUBTYPE" is missing, any subtype is matched. The value of these variables has the format:

[*flags*] *command*

where the *command* is expected to read from stdin and write to stdout, and the possible flags are

! Execute *command* in a sub-shell rather than doing an exec(3) (see *SHELL*).

+ Use this hook when selecting the part to display in a multipart/alternative block. Multipart blocks contain "alternative" versions with the same information, in increasing order of preference (and decoding complexity). The last one the mail agent understands is the one to be displayed. This is typically used for sending a message in both "plain text" and "html", but more complex subtypes are also possible.

- Do not decode before executing *command*.

If your *command* begins with one of these flags, precede it with a space to signal the end of the flags.

WARNING: automatically running a program is a potential security risk if that program has bugs, so be careful what you run.

Examples: View all Content-Type: image/jpeg parts with xv(1) (assuming it is installed):

```
set mime-body-image-jpeg="/usr/pkg/bin/xv -"
```

Decode all Content-Type: images/* blocks with uudeview(1) (assuming it is installed), placing the results in /tmp:

```
set mime-hook-image="-/usr/pkg/bin/uudeview -p /tmp -i -a +o -q -"
```

Read all `Content-Type: text/html` parts using `lynx`(1) (assuming it is installed) and add this support to `multipart/alternative` blocks:

```
set mime-body-text-html="+/usr/pkg/bin/lynx -force_html -dump -stdin"
```

ENVIRONMENT

`mail` uses the `HOME`, `TMPDIR`, and `USER` environment variables.

FILES

`/var/mail/*`	Post office.
~/mbox	User's old mail.
~/.mailrc	File giving initial mail commands. This can be overridden by setting the `MAILRC` environment variable.
`/tmp/mail.R*`	Temporary files.
`/usr/share/misc/mail.*help`	Help files.
`/etc/mail.rc`	System initialization file.

SEE ALSO

`fmt`(1), `newaliases`(1), `sendmail`(1), `vacation`(1), `aliases`(5), `mailaddr`(7) and

The Mail Reference Manual.

HISTORY

A **`mail`** command appeared in Version 6 AT&T UNIX. This man page is derived from *The Mail Reference Manual* originally written by Kurt Shoens.

BUGS

There are some flags and commands that are not documented here. Most are not useful to the general user.

Usually, **`mail`** is just a link to **`Mail`**, which can be confusing.

The name of the **`alternates`** list is incorrect English (it should be "alternatives"), but is retained for compatibility.

There must be sufficient space on $TMPDIR for various temporary files.

If an unrecoverable character set conversion error occurs (during display), the message is truncated and a warning is printed. This seems to be rare, but probably the remainder of the message should be printed without conversion.

The internal sh-like parser is not terribly sh-like.

Selecting messages by their content (i.e., with `/body:`) is rather slow.

NAME

make − GNU make utility to maintain groups of programs

SYNOPSIS

make [**−f** *makefile*] [option] ... target ...

WARNING

This man page is an extract of the documentation of *GNU make* . It is updated only occasionally, because the GNU project does not use nroff. For complete, current documentation, refer to the Info file **make.info** which is made from the Texinfo source file **make.texinfo**.

DESCRIPTION

The purpose of the *make* utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them. The manual describes the GNU implementation of *make*, which was written by Richard Stallman and Roland McGrath. Our examples show C programs, since they are most common, but you can use *make* with any programming language whose compiler can be run with a shell command. In fact, *make* is not limited to programs. You can use it to describe any task where some files must be updated automatically from others whenever the others change.

To prepare to use *make*, you must write a file called the *makefile* that describes the relationships among files in your program, and the states the commands for updating each file. In a program, typically the executable file is updated from object files, which are in turn made by compiling source files.

Once a suitable makefile exists, each time you change some source files, this simple shell command:

make

suffices to perform all necessary recompilations. The *make* program uses the makefile data base and the last-modification times of the files to decide which of the files need to be updated. For each of those files, it issues the commands recorded in the data base.

make executes commands in the *makefile* to update one or more target *names*, where *name* is typically a program. If no **−f** option is present, *make* will look for the makefiles *GNUmakefile*, *makefile*, and *Makefile*, in that order.

Normally you should call your makefile either *makefile* or *Makefile*. (We recommend *Makefile* because it appears prominently near the beginning of a directory listing, right near other important files such as *README*.) The first name checked, *GNUmakefile*, is not recommended for most makefiles. You should use this name if you have a makefile that is specific to GNU *make*, and will not be understood by other versions of *make*. If *makefile* is '−', the standard input is read.

make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist.

OPTIONS

−b

−m These options are ignored for compatibility with other versions of *make*.

−C *dir* Change to directory *dir* before reading the makefiles or doing anything else. If multiple **−C** options are specified, each is interpreted relative to the previous one: **−C** / **−C** etc is equivalent to **−C** /etc. This is typically used with recursive invocations of *make*.

−d Print debugging information in addition to normal processing. The debugging information says which files are being considered for remaking, which file-times are being compared and with what results, which files actually need to be remade, which implicit rules are considered and which are applied---everything interesting about how *make* decides what to do.

−e Give variables taken from the environment precedence over variables from makefiles.

−f *file* Use *file* as a makefile.

−i Ignore all errors in commands executed to remake files.

−I *dir* Specifies a directory *dir* to search for included makefiles. If several **−I** options are used to specify several directories, the directories are searched in the order specified. Unlike the arguments to other flags of *make*, directories given with **−I** flags may come directly after the flag: **−I***dir* is allowed, as well as **−I** *dir*. This syntax is allowed for compatibility with the C preprocessor's **−I** flag.

−j *jobs* Specifies the number of jobs (commands) to run simultaneously. If there is more than one **−j** option, the last one is effective. If the **−j** option is given without an argument, *make* will not limit the number of jobs that can run simultaneously.

−k Continue as much as possible after an error. While the target that failed, and those that depend on it, cannot be remade, the other dependencies of these targets can be processed all the same.

−l

−l *load* Specifies that no new jobs (commands) should be started if there are others jobs running and the load average is at least *load* (a floating-point number). With no argument, removes a previous load limit.

−n Print the commands that would be executed, but do not execute them.

−o *file* Do not remake the file *file* even if it is older than its dependencies, and do not remake anything on account of changes in *file*. Essentially the file is treated as very old and its rules are ignored.

−p Print the data base (rules and variable values) that results from reading the makefiles; then execute as usual or as otherwise specified. This also prints the version information given by the **−v** switch (see below). To print the data base without trying to remake any files, use **make −p −f***/dev/null*.

−q ''Question mode''. Do not run any commands, or print anything; just return an exit status that is zero if the specified targets are already up to date, nonzero otherwise.

−r Eliminate use of the built-in implicit rules. Also clear out the default list of suffixes for suffix rules.

−s Silent operation; do not print the commands as they are executed.

−S Cancel the effect of the **−k** option. This is never necessary except in a recursive *make* where **−k** might be inherited from the top-level *make* via MAKEFLAGS or if you set **−k** in MAKEFLAGS in your environment.

−t Touch files (mark them up to date without really changing them) instead of running their commands. This is used to pretend that the commands were done, in order to fool future invocations of *make*.

−v Print the version of the *make* program plus a copyright, a list of authors and a notice that there is no warranty.

−w Print a message containing the working directory before and after other processing. This may be useful for tracking down errors from complicated nests of recursive *make* commands.

−W *file*

Pretend that the target *file* has just been modified. When used with the **−n** flag, this shows you what would happen if you were to modify that file. Without **−n**, it is almost the same as running a *touch* command on the given file before running *make*, except that the modification time is changed only in the imagination of *make*.

SEE ALSO

The GNU Make Manual

BUGS

See the chapter 'Problems and Bugs' in *The GNU Make Manual* .

AUTHOR

This manual page contributed by Dennis Morse of Stanford University. It has been reworked by Roland McGrath.

NAME

make — maintain program dependencies

SYNOPSIS

make [**-BeikNnqrstWX**] [**-D** *variable*] [**-d** *flags*] [**-f** *makefile*] [**-I** *directory*] [**-J** *private*] [**-j** *max_jobs*] [**-m** *directory*] [**-T** *file*] [**-V** *variable*] [*variable=value*] [*target ...*]

DESCRIPTION

make is a program designed to simplify the maintenance of other programs. Its input is a list of specifications as to the files upon which programs and other files depend. If no **-f** *makefile* makefile option is given, **make** will try to open `makefile` then `Makefile` in order to find the specifications. If the file `.depend` exists, it is read (see `mkdep`(1)).

This manual page is intended as a reference document only. For a more thorough description of **make** and makefiles, please refer to *Make − A Tutorial*.

make will prepend the contents of the *MAKEFLAGS* environment variable to the command line arguments before parsing them.

The options are as follows:

-B Try to be backwards compatible by executing a single shell per command and by executing the commands to make the sources of a dependency line in sequence.

-D *variable*
Define *variable* to be 1, in the global context.

-d *[-]flags*
Turn on debugging, and specify which portions of **make** are to print debugging information. Unless the flags are preceded by '-' they are added to the *MAKEFLAGS* environment variable and will be processed by any child make processes. *Flags* is one or more of the following:

A Print all possible debugging information; equivalent to specifying all of the debugging flags.

a Print debugging information about archive searching and caching.

c Print debugging information about conditional evaluation.

d Print debugging information about directory searching and caching.

e Print debugging information about failed commands and targets.

F Use the rest of `flags` as the name of the file to which the debug output is written. If the filename ends `.%d` then the '`%d`' is replaced by the pid.

f Print debugging information about loop evaluation.

g1 Print the input graph before making anything.

g2 Print the input graph after making everything, or before exiting on error.

g3 Print the input graph before exiting on error.

j Print debugging information about running multiple shells.

l Print commands in Makefiles regardless of whether or not they are prefixed by '@' or other "quiet" flags. Also known as "loud" behavior.

m Print debugging information about making targets, including modification dates.

n Don't delete the temporary command scripts created in `/tmp` when running commands. These are created via `mkstemp`(3) and have names of the form `/tmp/makeXXXXX`. *NOTE*: This can create many file in `/tmp` so use with care.

p Print debugging information about makefile parsing.

s Print debugging information about suffix-transformation rules.

t Print debugging information about target list maintenance.

v Print debugging information about variable assignment.

x Run shell commands with **-x** so the actual commands are printed as they are executed.

-e Specify that environment variables override macro assignments within makefiles.

-f *makefile*
Specify a makefile to read instead of the default `makefile`. If *makefile* is '**-**', standard input is read. Multiple makefiles may be specified, and are read in the order specified.

-I *directory*
Specify a directory in which to search for makefiles and included makefiles. The system makefile directory (or directories, see the **-m** option) is automatically included as part of this list.

-i Ignore non-zero exit of shell commands in the makefile. Equivalent to specifying '**-**' before each command line in the makefile.

-J *private*
This option should *not* be specified by the user.

When the *j* option is in use in a recursive build, this option is passed by a make to child makes to allow all the make processes in the build to cooperate to avoid overloading the system.

-j *max_jobs*
Specify the maximum number of jobs that **make** may have running at any one time. Turns compatibility mode off, unless the *B* flag is also specified.

-k Continue processing after errors are encountered, but only on those targets that do not depend on the target whose creation caused the error.

-m *directory*
Specify a directory in which to search for sys.mk and makefiles included via the ⟨*file*⟩-style include statement. The **-m** option can be used multiple times to form a search path. This path will override the default system include path: /usr/share/mk. Furthermore the system include path will be appended to the search path used for "*file*"-style include statements (see the **-I** option).

If a file or directory name in the **-m** argument (or the `MAKESYSPATH` environment variable) starts with the string ".../" then **make** will search for the specified file or directory named in the remaining part of the argument string. The search starts with the current directory of the Makefile and then works upward towards the root of the filesystem. If the search is successful, then the resulting directory replaces the ".../" specification in the **-m** argument. If used, this feature allows **make** to easily search in the current source tree for customized sys.mk files (e.g., by using ".../mk/sys.mk" as an argument).

-n Display the commands that would have been executed, but do not actually execute them unless the target depends on the .MAKE special source (see below).

-N Display the commands which would have been executed, but do not actually execute any of them; useful for debugging top-level makefiles without descending into subdirectories.

-q Do not execute any commands, but exit 0 if the specified targets are up-to-date and 1, otherwise.

-r Do not use the built-in rules specified in the system makefile.

-s Do not echo any commands as they are executed. Equivalent to specifying **@** before each command line in the makefile.

-T *tracefile*
When used with the **-j** flag, append a trace record to *tracefile* for each job started and completed.

-t Rather than re-building a target as specified in the makefile, create it or update its modification time to make it appear up-to-date.

-V *variable*
Print **make**'s idea of the value of *variable*, in the global context. Do not build any targets. Multiple instances of this option may be specified; the variables will be printed one per line, with a blank line for each null or undefined variable. If *variable* contains a '$' then the value will be expanded before printing.

-W Treat any warnings during makefile parsing as errors.

-X Don't export variables passed on the command line to the environment individually. Variables passed on the command line are still exported via the *MAKEFLAGS* environment variable. This option may be useful on systems which have a small limit on the size of command arguments.

variable=value
Set the value of the variable *variable* to *value*. Normally, all values passed on the command line are also exported to sub-makes in the environment. The **-X** flag disables this behavior. Variable assignments should follow options for POSIX compatibility but no ordering is enforced.

There are seven different types of lines in a makefile: file dependency specifications, shell commands, variable assignments, include statements, conditional directives, for loops, and comments.

In general, lines may be continued from one line to the next by ending them with a backslash ('\'). The trailing newline character and initial whitespace on the following line are compressed into a single space.

FILE DEPENDENCY SPECIFICATIONS

Dependency lines consist of one or more targets, an operator, and zero or more sources. This creates a relationship where the targets ''depend'' on the sources and are usually created from them. The exact relationship between the target and the source is determined by the operator that separates them. The three operators are as follows:

: A target is considered out-of-date if its modification time is less than those of any of its sources. Sources for a target accumulate over dependency lines when this operator is used. The target is removed if **make** is interrupted.

! Targets are always re-created, but not until all sources have been examined and re-created as necessary. Sources for a target accumulate over dependency lines when this operator is used. The target is removed if **make** is interrupted.

:: If no sources are specified, the target is always re-created. Otherwise, a target is considered out-of-date if any of its sources has been modified more recently than the target. Sources for a target do not accumulate over dependency lines when this operator is used. The target will not be removed if **make** is interrupted.

Targets and sources may contain the shell wildcard values ‘?’, ‘*’, ‘[]’, and ‘{}’. The values ‘?’, ‘*’, and ‘[]’ may only be used as part of the final component of the target or source, and must be used to describe existing files. The value ‘{}’ need not necessarily be used to describe existing files. Expansion is in directory order, not alphabetically as done in the shell.

SHELL COMMANDS

Each target may have associated with it a series of shell commands, normally used to create the target. Each of the commands in this script *must* be preceded by a tab. While any target may appear on a dependency line, only one of these dependencies may be followed by a creation script, unless the **::** operator is used.

If the first characters of the command line are any combination of **@**, **+**, or **−**, the command is treated specially. A **@** causes the command not to be echoed before it is executed. A **+** causes the command to be executed even when **−n** is given. This is similar to the effect of the .MAKE special source, except that the effect can be limited to a single line of a script. A **−** causes any non-zero exit status of the command line to be ignored.

VARIABLE ASSIGNMENTS

Variables in make are much like variables in the shell, and, by tradition, consist of all upper-case letters.

Variable assignment modifiers

The five operators that can be used to assign values to variables are as follows:

= Assign the value to the variable. Any previous value is overridden.

+= Append the value to the current value of the variable.

?= Assign the value to the variable if it is not already defined.

:= Assign with expansion, i.e. expand the value before assigning it to the variable. Normally, expansion is not done until the variable is referenced. *NOTE*: References to undefined variables are *not* expanded. This can cause problems when variable modifiers are used.

!= Expand the value and pass it to the shell for execution and assign the result to the variable. Any newlines in the result are replaced with spaces.

Any white-space before the assigned *value* is removed; if the value is being appended, a single space is inserted between the previous contents of the variable and the appended value.

Variables are expanded by surrounding the variable name with either curly braces (‘{}’) or parentheses (‘()’) and preceding it with a dollar sign (‘$’). If the variable name contains only a single letter, the surrounding braces or parentheses are not required. This shorter form is not recommended.

Variable substitution occurs at two distinct times, depending on where the variable is being used. Variables in dependency lines are expanded as the line is read. Variables in shell commands are expanded when the shell command is executed.

Variable classes

The four different classes of variables (in order of increasing precedence) are:

Environment variables
: Variables defined as part of **make**’s environment.

Global variables
: Variables defined in the makefile or in included makefiles.

Command line variables
: Variables defined as part of the command line.

Local variables

Variables that are defined specific to a certain target. The seven local variables are as follows:

.ALLSRC The list of all sources for this target; also known as >.

.ARCHIVE The name of the archive file.

.IMPSRC The name/path of the source from which the target is to be transformed (the ''implied'' source); also known as <.

.MEMBER The name of the archive member.

.OODATE The list of sources for this target that were deemed out-of-date; also known as *?*.

.PREFIX The file prefix of the file, containing only the file portion, no suffix or preceding directory components; also known as ∗.

.TARGET The name of the target; also known as @.

The shorter forms @, *?*, <, >, and ∗ are permitted for backward compatibility with historical makefiles and are not recommended. The six variables *@F*, *@D*, *<F*, *<D*, *∗F*, and *∗D* are permitted for compatibility with AT&T System V UNIX makefiles and are not recommended.

Four of the local variables may be used in sources on dependency lines because they expand to the proper value for each target on the line. These variables are *.TARGET*, *.PREFIX*, *.ARCHIVE*, and *.MEMBER*.

Additional inbuilt variables

In addition, **make** sets or knows about the following variables:

*$* A single dollar sign '$', i.e. '$$' expands to a single dollar sign.

.ALLTARGETS The list of all targets encountered in the Makefile. If evaluated during Makefile parsing, lists only those targets encountered thus far.

.CURDIR A path to the directory where **make** was executed. Refer to the description of PWD for more details.

MAKE The name that **make** was executed with (*argv[0]*). For compatibility **make** also sets *.MAKE* with the same value. The preferred variable to use is the environment variable MAKE because it is more compatible with other versions of **make** and cannot be confused with the special target with the same name.

.MAKE.EXPORTED The list of variables exported by **make**.

.MAKE.MAKEFILES

The list of makefiles read by **make**, which is useful for tracking dependencies. Each makefile is recorded only once, regardless of the number of times read.

.MAKE.PID The process-id of **make**.

.MAKE.PPID The parent process-id of **make**.

.MAKE.JOB.PREFIX

If **make** is run with *-j* then output for each target is prefixed with a token `--- target ---` the first part of which can be controlled via *.MAKE.JOB.PREFIX*. For example:
`.MAKE.JOB.PREFIX=${.newline}---${.MAKE:T}[${.MAKE.PID}]`
would produce tokens like `---make[1234] target ---` making it easier to track the degree of parallelism being achieved.

MAKEFLAGS The environment variable MAKEFLAGS may contain anything that may be specified on **make**'s command line. Anything specified on **make**'s command line is appended to the MAKEFLAGS variable which is then entered into the environment for all programs which **make** executes.

.MAKEOVERRIDES This variable is used to record the names of variables assigned to on the command line, so that they may be exported as part of MAKEFLAGS. This behaviour can be disabled by assigning an empty value to *.MAKEOVERRIDES* within a makefile. Extra variables can be exported from a makefile by appending their names to *.MAKEOVERRIDES*. MAKEFLAGS is re-exported whenever *.MAKEOVERRIDES* is modified.

MAKE_PRINT_VAR_ON_ERROR
When **make** stops due to an error, it prints its name and the value of *.CURDIR* as well as the value of any variables named in *MAKE_PRINT_VAR_ON_ERROR*.

.newline This variable is simply assigned a newline character as its value. This allows expansions using the **:@** modifier to put a newline between iterations of the loop rather than a space. For example, the printing of *MAKE_PRINT_VAR_ON_ERROR* could be done as ${MAKE_PRINT_VAR_ON_ERROR:@v@$v='${$v}'${.newline}@}.

.OBJDIR A path to the directory where the targets are built. Its value is determined by trying to chdir(2) to the following directories in order and using the first match:

1. ${MAKEOBJDIRPREFIX}${.CURDIR}

   (Only if MAKEOBJDIRPREFIX is set in the environment or on the command line.)

2. ${MAKEOBJDIR}

   (Only if MAKEOBJDIR is set in the environment or on the command line.)

3. ${.CURDIR}/obj.${MACHINE}

4. ${.CURDIR}/obj

5. /usr/obj/${.CURDIR}

6. ${.CURDIR}

Variable expansion is performed on the value before it's used, so expressions such as

```
${.CURDIR:C,^/usr/src,/var/obj,}
```

may be used.

.OBJDIR may be modified in the makefile as a global variable. In all cases, **make** will chdir(2) to *.OBJDIR* and set PWD to that directory before executing any targets.

.PARSEDIR A path to the directory of the current Makefile being parsed.

.PARSEFILE The basename of the current Makefile being parsed. This variable and *.PARSEDIR* are both set only while the Makefiles are being parsed.

.PATH A variable that represents the list of directories that **make** will search for files. The search list should be updated using the target *.PATH* rather than the variable.

PWD Alternate path to the current directory. **make** normally sets *.CURDIR* to the canonical path given by getcwd(3). However, if the environment variable PWD is set and gives a path to the current directory, then **make** sets *.CURDIR* to the value of PWD instead. This behaviour is disabled if MAKEOBJDIRPREFIX is set or MAKEOBJDIR contains a variable transform. PWD is set to the value of *.OBJDIR* for all programs which **make** executes.

Variable modifiers

Variable expansion may be modified to select or modify each word of the variable (where a ‘‘word’’ is white-space delimited sequence of characters). The general format of a variable expansion is as follows:

```
${variable[:modifier[:...]]}
```

Each modifier begins with a colon, which may be escaped with a backslash (‘\’).

A set of modifiers can be specified via a variable, as follows:

```
modifier_variable=modifier[:...]
${variable:${modifier_variable}[:...]}
```

In this case the first modifier in the modifier_variable does not start with a colon, since that must appear in the referencing variable. If any of the modifiers in the modifier_variable contain a dollar sign (‘$’), these must be doubled to avoid early expansion.

The supported modifiers are:

:E Replaces each word in the variable with its suffix.

:H Replaces each word in the variable with everything but the last component.

:M*pattern*
Select only those words that match *pattern*. The standard shell wildcard characters (‘*’, ‘?’, and ‘[ ]’) may be used. The wildcard characters may be escaped with a backslash (‘\’).

:N*pattern*
This is identical to **:M**, but selects all words which do not match *pattern*.

:O Order every word in variable alphabetically. To sort words in reverse order use the **:O:[-1..1]** combination of modifiers.

:Ox Randomize words in variable. The results will be different each time you are referring to the modified variable; use the assignment with expansion (**:=**) to prevent such behaviour. For example,

```
LIST=                   uno due tre quattro
RANDOM_LIST=            ${LIST:Ox}
STATIC_RANDOM_LIST:=    ${LIST:Ox}

all:
        @echo "${RANDOM_LIST}"
        @echo "${RANDOM_LIST}"
        @echo "${STATIC_RANDOM_LIST}"
        @echo "${STATIC_RANDOM_LIST}"
```

may produce output similar to:

```
quattro due tre uno
tre due quattro uno
due uno quattro tre
due uno quattro tre
```

:Q Quotes every shell meta-character in the variable, so that it can be passed safely through recursive invocations of **make**.

:R Replaces each word in the variable with everything but its suffix.

:tl Converts variable to lower-case letters.

:ts*c*
Words in the variable are normally separated by a space on expansion. This modifier sets the separator to the character *c*. If *c* is omitted, then no separator is used.

:tu Converts variable to upper-case letters.

:tW Causes the value to be treated as a single word (possibly containing embedded white space). See also **:[***]**.

:tw Causes the value to be treated as a sequence of words delimited by white space. See also **:[@]**.

:S/*old_string*/*new_string*/[**1gW**]
Modify the first occurrence of *old_string* in the variable's value, replacing it with *new_string*. If a 'g' is appended to the last slash of the pattern, all occurrences in each word are replaced. If a '1' is appended to the last slash of the pattern, only the first word is affected. If a 'W' is appended to the last slash of the pattern, then the value is treated as a single word (possibly containing embedded white space). If *old_string* begins with a caret ('^'), *old_string* is anchored at the beginning of each word. If *old_string* ends with a dollar sign ('$'), it is anchored at the end of each word. Inside *new_string*, an ampersand ('&') is replaced by *old_string* (without any '^' or '$'). Any character may be used as a delimiter for the parts of the modifier string. The anchoring, ampersand and delimiter characters may be escaped with a backslash ('\').

Variable expansion occurs in the normal fashion inside both *old_string* and *new_string* with the single exception that a backslash is used to prevent the expansion of a dollar sign ('$'), not a preceding dollar sign as is usual.

:C/*pattern*/*replacement*/[**1gW**]
The **:C** modifier is just like the **:S** modifier except that the old and new strings, instead of being simple strings, are a regular expression (see regex(3)) string *pattern* and an ed(1)−style string *replacement*. Normally, the first occurrence of the pattern *pattern* in each word of the value is substituted with *replacement*. The '1' modifier causes the substitution to apply to at most one word; the 'g' modifier causes the substitution to apply to as many instances of the search pattern *pattern* as occur in the word or words it is found in; the 'W' modifier causes the value to be treated as a single word (possibly containing embedded white space). Note that '1' and 'g' are orthogonal; the former specifies whether multiple words are potentially affected, the latter whether multiple substitutions can potentially occur within each affected word.

:T Replaces each word in the variable with its last component.

:u Remove adjacent duplicate words (like uniq(1)).

:?*true_string*:*false_string*
If the variable (actually an expression; see below) evaluates to true, return as its value the *true_string*, otherwise return the *false_string*.

:old_string=new_string
This is the AT&T System V UNIX style variable substitution. It must be the last modifier specified. If *old_string* or *new_string* do not contain the pattern matching character *%* then it is assumed that they are anchored at the end of each word, so only suffixes or entire words may be replaced. Otherwise *%* is the substring of *old_string* to be replaced in *new_string*.

Variable expansion occurs in the normal fashion inside both *old_string* and *new_string* with the single exception that a backslash is used to prevent the expansion of a dollar sign ('$'), not a preceding dollar sign as is usual.

:@*temp*@*string*@
This is the loop expansion mechanism from the OSF Development Environment (ODE) make. Unlike **.for** loops expansion occurs at the time of reference. Assign *temp* to each word in the variable and

evaluate *string*. The ODE convention is that *temp* should start and end with a period. For example.

```
${LINKS:@.LINK.@${LN} ${TARGET} ${.LINK.}@}
```

:U*newval*

If the variable is undefined *newval* is the value. If the variable is defined, the existing value is returned. This is another ODE make feature. It is handy for setting per-target CFLAGS for instance:

```
${_${.TARGET:T}_CFLAGS:U${DEF_CFLAGS}}
```

If a value is only required if the variable is undefined, use:

```
${VAR:D:Unewval}
```

:D*newval*

If the variable is defined *newval* is the value.

:L The name of the variable is the value.

:P The path of the node which has the same name as the variable is the value. If no such node exists or its path is null, then the name of the variable is used.

:!*cmd***!**

The output of running *cmd* is the value.

:sh If the variable is non-empty it is run as a command and the output becomes the new value.

::=*str*

The variable is assigned the value *str* after substitution. This modifier and its variations are useful in obscure situations such as wanting to apply modifiers to **.for** loop iteration variables which won't work due to the way **.for** loops are implemented. These assignment modifiers always expand to nothing, so if appearing in a rule line by themselves should be preceded with something to keep **make** happy. As in:

```
use_foo: .USE
.for i in ${.TARGET} ${.TARGET:R}.gz
        @: ${t::=$i}
        @echo t:R:T=${t:R:T}
.endfor
```

The **::** helps avoid false matches with the AT&T System V UNIX style **:=** modifier and since substitution always occurs the **::=** form is vaguely appropriate.

::?=*str*

As for **::=** but only if the variable does not already have a value.

::+=*str*

Append *str* to the variable.

::!=*cmd*

Assign the output of *cmd* to the variable.

:[*range***]**

Selects one or more words from the value, or performs other operations related to the way in which the value is divided into words.

Ordinarily, a value is treated as a sequence of words delimited by white space. Some modifiers suppress this behaviour, causing a value to be treated as a single word (possibly containing embedded white space). An empty value, or a value that consists entirely of white-space, is treated as a single word. For the purposes of the **:[]** modifier, the words are indexed both forwards using positive integers (where index 1 represents the first word), and backwards using negative integers (where index -1 represents the last word).

The *range* is subjected to variable expansion, and the expanded result is then interpreted as follows:

index Selects a single word from the value.

start..*end*
Selects all words from *start* to *end*, inclusive. For example, **:[2..-1]** selects all words from the second word to the last word. If *start* is greater than *end*, then the words are output in reverse order. For example, **:[-1..1]** selects all the words from last to first.

* Causes subsequent modifiers to treat the value as a single word (possibly containing embedded white space). Analogous to the effect of "$*" in Bourne shell.

0 Means the same as **:[*]**.

@ Causes subsequent modifiers to treat the value as a sequence of words delimited by white space. Analogous to the effect of "$@" in Bourne shell.

Returns the number of words in the value.

INCLUDE STATEMENTS, CONDITIONALS AND FOR LOOPS

Makefile inclusion, conditional structures and for loops reminiscent of the C programming language are provided in **make**. All such structures are identified by a line beginning with a single dot (‘.’) character. Files are included with either **.include** ⟨*file*⟩ or **.include** "*file*". Variables between the angle brackets or double quotes are expanded to form the file name. If angle brackets are used, the included makefile is expected to be in the system makefile directory. If double quotes are used, the including makefile’s directory and any directories specified using the **-I** option are searched before the system makefile directory. For compatibility with other versions of **make** `include file ...` is also accepted. If the include statement is written as **.-include** or as **.sinclude** then errors locating and/or opening include files are ignored.

Conditional expressions are also preceded by a single dot as the first character of a line. The possible conditionals are as follows:

.export *variable*
Export the specified global variable. If no variable is provided, all globals are exported except for internal variables (those that start with ‘.’). This is not affected by the **-X** flag, so should be used with caution. Appending a variable name to *.MAKE.EXPORTED* is equivalent to exporting a variable.

.undef *variable*
Un-define the specified global variable. Only global variables may be un-defined.

.if [!]*expression* [*operator expression ...*]
Test the value of an expression.

.ifdef [!]*variable* [*operator variable ...*]
Test the value of a variable.

.ifndef [!]*variable* [*operator variable ...*]
Test the value of a variable.

.ifmake [!]*target* [*operator target ...*]
Test the target being built.

.ifnmake [!] *target* [*operator target ...*]
Test the target being built.

.else Reverse the sense of the last conditional.

.elif [!] *expression* [*operator expression ...*]
A combination of **.else** followed by **.if**.

.elifdef [!]*variable* [*operator variable ...*]
A combination of **.else** followed by **.ifdef**.

.elifndef [!]*variable* [*operator variable ...*]
A combination of **.else** followed by **.ifndef**.

.elifmake [!]*target* [*operator target ...*]
A combination of **.else** followed by **.ifmake**.

.elifnmake [!]*target* [*operator target ...*]
A combination of **.else** followed by **.ifnmake**.

.endif
End the body of the conditional.

The *operator* may be any one of the following:

|| Logical OR.

&& Logical AND; of higher precedence than "||".

As in C, **make** will only evaluate a conditional as far as is necessary to determine its value. Parentheses may be used to change the order of evaluation. The boolean operator **!** may be used to logically negate an entire conditional. It is of higher precedence than **&&**.

The value of *expression* may be any of the following:

defined Takes a variable name as an argument and evaluates to true if the variable has been defined.

make Takes a target name as an argument and evaluates to true if the target was specified as part of **make**'s command line or was declared the default target (either implicitly or explicitly, see *.MAIN*) before the line containing the conditional.

empty Takes a variable, with possible modifiers, and evaluates to true if the expansion of the variable would result in an empty string.

exists Takes a file name as an argument and evaluates to true if the file exists. The file is searched for on the system search path (see *.PATH*).

target Takes a target name as an argument and evaluates to true if the target has been defined.

commands
Takes a target name as an argument and evaluates to true if the target has been defined and has commands associated with it.

Expression may also be an arithmetic or string comparison. Variable expansion is performed on both sides of the comparison, after which the integral values are compared. A value is interpreted as hexadecimal if it is preceded by 0x, otherwise it is decimal; octal numbers are not supported. The standard C relational operators are all supported. If after variable expansion, either the left or right hand side of a **==** or **!=** operator is not an integral value, then string comparison is performed between the expanded variables. If no relational operator is given, it is assumed that the expanded variable is being compared against 0 or an empty string in the case of a string comparison.

When **make** is evaluating one of these conditional expression, and it encounters a word it doesn't recognize, either the ''make'' or ''defined'' expression is applied to it, depending on the form of the conditional. If the form is **.ifdef** or **.ifndef**, the ''defined'' expression is applied. Similarly, if the form is **.ifmake** or

.ifnmake, **the ‘‘make’’** expression is applied.

If the conditional evaluates to true the parsing of the makefile continues as before. If it evaluates to false, the following lines are skipped. In both cases this continues until a **.else** or **.endif** is found.

For loops are typically used to apply a set of rules to a list of files. The syntax of a for loop is:

.for *variable* [*variable ...*] **in** *expression*
⟨make-rules⟩
.endfor

After the for **expression** is evaluated, it is split into words. On each iteration of the loop, one word is taken and assigned to each **variable**, in order, and these **variables** are substituted into the **make-rules** inside the body of the for loop. The number of words must come out even; that is, if there are three iteration variables, the number of words provided must be a multiple of three.

COMMENTS

Comments begin with a hash (‘#’) character, anywhere but in a shell command line, and continue to the end of an unescaped new line.

SPECIAL SOURCES (ATTRIBUTES)

.EXEC Target is never out of date, but always execute commands anyway.

.IGNORE Ignore any errors from the commands associated with this target, exactly as if they all were preceded by a dash (‘−’).

.MADE Mark all sources of this target as being up-to-date.

.MAKE Execute the commands associated with this target even if the **−n** or **−t** options were specified. Normally used to mark recursive **make**’s.

.NOPATH Do not search for the target in the directories specified by **.PATH**.

.NOTMAIN Normally **make** selects the first target it encounters as the default target to be built if no target was specified. This source prevents this target from being selected.

.OPTIONAL
If a target is marked with this attribute and **make** can’t figure out how to create it, it will ignore this fact and assume the file isn’t needed or already exists.

.PHONY The target does not correspond to an actual file; it is always considered to be out of date, and will not be created with the **−t** option.

.PRECIOUS
When **make** is interrupted, it normally removes any partially made targets. This source prevents the target from being removed.

.RECURSIVE
Synonym for **.MAKE**.

.SILENT Do not echo any of the commands associated with this target, exactly as if they all were preceded by an at sign (‘@’).

.USE Turn the target into **make**’s version of a macro. When the target is used as a source for another target, the other target acquires the commands, sources, and attributes (except for **.USE**) of the source. If the target already has commands, the **.USE** target’s commands are appended to them.

.USEBEFORE
Exactly like **.USE**, but prepend the **.USEBEFORE** target commands to the target.

.WAIT If **.WAIT** appears in a dependency line, the sources that precede it are made before the sources that succeed it in the line. Since the dependents of files are not made until the file itself could be made, this also stops the dependents being built unless they are needed for another branch of the dependency tree. So given:

```
x: a .WAIT b
        echo x
a:
        echo a
b: b1
        echo b
b1:
        echo b1
```

the output is always ‘b1’, ‘b’, ‘a’, ‘x’.
The ordering imposed by **.WAIT** is only relevant for parallel makes.

SPECIAL TARGETS

Special targets may not be included with other targets, i.e. they must be the only target specified.

.BEGIN Any command lines attached to this target are executed before anything else is done.

.DEFAULT
This is sort of a **.USE** rule for any target (that was used only as a source) that **make** can’t figure out any other way to create. Only the shell script is used. The **.IMPSRC** variable of a target that inherits **.DEFAULT**’s commands is set to the target’s own name.

.END Any command lines attached to this target are executed after everything else is done.

.IGNORE Mark each of the sources with the **.IGNORE** attribute. If no sources are specified, this is the equivalent of specifying the **-i** option.

.INTERRUPT
If **make** is interrupted, the commands for this target will be executed.

.MAIN If no target is specified when **make** is invoked, this target will be built.

.MAKEFLAGS
This target provides a way to specify flags for **make** when the makefile is used. The flags are as if typed to the shell, though the **-f** option will have no effect.

.NOPATH Apply the **.NOPATH** attribute to any specified sources.

.NOTPARALLEL
Disable parallel mode.

.NO_PARALLEL
Synonym for **.NOTPARALLEL**, for compatibility with other pmake variants.

.ORDER The named targets are made in sequence. This ordering does not add targets to the list of targets to be made. Since the dependents of a target do not get built until the target itself could be built, unless ‘a’ is built by another part of the dependency graph, the following is a dependency loop:

```
.ORDER a b
b: a
```

The ordering imposed by **.ORDER** is only relevant for parallel makes.

.PATH The sources are directories which are to be searched for files not found in the current directory. If no sources are specified, any previously specified directories are deleted. If the source is the special **.DOTLAST** target, then the current working directory is searched last.

.PHONY Apply the **.PHONY** attribute to any specified sources.

.PRECIOUS
Apply the **.PRECIOUS** attribute to any specified sources. If no sources are specified, the **.PRECIOUS** attribute is applied to every target in the file.

.SHELL Sets the shell that **make** will use to execute commands. The sources are a set of *field*=*value* pairs.

name This is the minimal specification, used to select one of the builtin shell specs; *sh*, *ksh*, and *csh*.

path Specifies the path to the shell.

hasErrCtl Indicates whether the shell supports exit on error.

check The command to turn on error checking.

ignore The command to disable error checking.

echo The command to turn on echoing of commands executed.

quiet The command to turn off echoing of commands executed.

filter The output to filter after issuing the *quiet* command. It is typically identical to *quiet*.

errFlag The flag to pass the shell to enable error checking.

echoFlag The flag to pass the shell to enable command echoing.

newline The string literal to pass the shell that results in a single newline character when used outside of any quoting characters.

Example:

```
.SHELL: name=ksh path=/bin/ksh hasErrCtl=true \
        check="set -e" ignore="set +e" \
        echo="set -v" quiet="set +v" filter="set +v" \
        echoFlag=v errFlag=e newline="'\n'"
```

.SILENT Apply the **.SILENT** attribute to any specified sources. If no sources are specified, the **.SILENT** attribute is applied to every command in the file.

.SUFFIXES
Each source specifies a suffix to **make**. If no sources are specified, any previously specified suffixes are deleted.

ENVIRONMENT

make uses the following environment variables, if they exist: MACHINE, MACHINE_ARCH, MAKE, MAKEFLAGS, MAKEOBJDIR, MAKEOBJDIRPREFIX, MAKESYSPATH, and PWD.

MAKEOBJDIRPREFIX and MAKEOBJDIR may only be set in the environment or on the command line to **make** and not as makefile variables; see the description of *.OBJDIR* for more details.

FILES

.depend	list of dependencies
Makefile	list of dependencies
makefile	list of dependencies
sys.mk	system makefile
/usr/share/mk	system makefile directory

COMPATIBILITY

The basic make syntax is compatible between different versions of make, however the special variables, variable modifiers and conditionals are not.

The way that parallel makes are scheduled changed in NetBSD 4.0 so that .ORDER and .WAIT apply recursively to the dependant nodes. The algorithms used may change again in the future.

SEE ALSO

mkdep(1)

HISTORY

A **make** command appeared in Version 7 AT&T UNIX.

NAME

makeinfo − translate Texinfo documents

SYNOPSIS

makeinfo [*OPTION*]... *TEXINFO-FILE*...

DESCRIPTION

Translate Texinfo source documentation to various other formats, by default Info files suitable for reading online with Emacs or standalone GNU Info.

General options:

−−error−limit=*NUM*
quit after NUM errors (default 100).

−−force
preserve output even if errors.

−−help display this help and exit.

−−no−validate
suppress node cross-reference validation.

−−no−warn
suppress warnings (but not errors).

−−reference−limit=*NUM*
warn about at most NUM references (default 1000).

−v, **−−verbose**
explain what is being done.

−−version
display version information and exit.

Output format selection (default is to produce Info):

−−docbook
output Docbook XML rather than Info.

−−html
output HTML rather than Info.

−−xml output Texinfo XML rather than Info.

−−plaintext
output plain text rather than Info.

General output options:

−E, **−−macro−expand** FILE
output macro-expanded source to FILE. ignoring any @setfilename.

−−no−headers
suppress node separators, Node: lines, and menus from Info output (thus producing plain text) or from HTML (thus producing shorter output); also, write to standard output by default.

−−no−split
suppress splitting of Info or HTML output, generate only one output file.

−−number−sections
output chapter and sectioning numbers.

−o, **−−output**=*FILE*
output to FILE (directory if split HTML),

Options for Info and plain text:

−−enable−encoding
output accented and special characters in Info output based on @documentencoding.

−−fill−column=*NUM*
break Info lines at NUM characters (default 72).

−−footnote−style=*STYLE*
output footnotes in Info according to STYLE: ‘separate’ to put them in their own node; ‘end’ to put them at the end of the node

in which they are defined (default).

−−paragraph−indent=*VAL*
indent Info paragraphs by VAL spaces (default 3). If VAL is ‘none’, do not indent; if VAL is ‘asis’, preserve existing indentation.

−−split−size=*NUM*
split Info files at size NUM (default 300000).

Options for HTML:

−−css−include=*FILE*
include FILE in HTML <style> output; read stdin if FILE is -.

Options for XML and Docbook:

−−output−indent=*VAL*
indent XML elements by VAL spaces (default 2). If VAL is 0, ignorable whitespace is dropped.

Input file options:

−−commands−in−node−names
allow @ commands in node names.

−D VAR
define the variable VAR, as with @set.

−I DIR append DIR to the @include search path.

−P DIR
prepend DIR to the @include search path.

−U VAR
undefine the variable VAR, as with @clear.

Conditional processing in input:

−−ifdocbook
process @ifdocbook and @docbook even if not generating Docbook.

−−ifhtml
process @ifhtml and @html even if not generating HTML.

−−ifinfo
process @ifinfo even if not generating Info.

−−ifplaintext
process @ifplaintext even if not generating plain text.

−−iftex process @iftex and @tex; implies **−−no−split**.

−−ifxml
process @ifxml and @xml.

−−no−ifdocbook
do not process @ifdocbook and @docbook text.

−−no−ifhtml
do not process @ifhtml and @html text.

−−no−ifinfo
do not process @ifinfo text.

−−no−ifplaintext
do not process @ifplaintext text.

−−no−iftex
do not process @iftex and @tex text.

−−no−ifxml
do not process @ifxml and @xml text.

The defaults for the @if... conditionals depend on the output format: if generating HTML, **−−ifhtml** is on and the others are off; if generating Info, **−−ifinfo** is on and the others are off; if generating plain text, **−−ifplaintext** is on and the others are off; if generating XML, **−−ifxml** is on and the others are off.

EXAMPLES

makeinfo foo.texi
write Info to foo's @setfilename

makeinfo **−−html** foo.texi
write HTML to @setfilename

makeinfo **−−xml** foo.texi
write Texinfo XML to @setfilename

makeinfo **−−docbook** foo.texi
write DocBook XML to @setfilename

makeinfo **−−no−headers** foo.texi
write plain text to standard output

makeinfo **−−html −−no−headers** foo.texi write html without node lines, menus makeinfo **−−number−sections** foo.texi write Info with numbered sections makeinfo **−−no−split** foo.texi write one Info file however big

REPORTING BUGS

Email bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org. Texinfo home page: http://www.gnu.org/software/texinfo/

COPYRIGHT

SEE ALSO

The full documentation for **makeinfo** is maintained as a Texinfo manual. If the **info** and **makeinfo** programs are properly installed at your site, the command

info makeinfo

should give you access to the complete manual.

NAME

man — display the on-line manual pages (aka "*man pages*")

SYNOPSIS

man [**-acw**|**-h**] [**-C** *file*] [**-M** *path*] [**-m** *path*] [**-S** *srch*] [[**-s**] *section*] *name ...*
man [**-k**] [**-C** *file*] [**-M** *path*] [**-m** *path*] *keyword ...*

DESCRIPTION

The **man** utility displays the manual pages named on the command line. Its options are as follows:

-a Display all of the man pages for a specified *section* and *name* combination. (Normally, only the first man page found is displayed.)

-C Use the specified *file* instead of the default configuration file. This permits users to configure their own man environment. See `man.conf`(5) for a description of the contents of this file.

-c Copy the man page to the standard output instead of using `more`(1) to paginate it. This is done by default if the standard output is not a terminal device.

-h Display only the "SYNOPSIS" lines of the requested man pages. For commands, this is typically the command line usage information. For library functions, this usually contains the required include files and function prototypes.

-k Display the header lines for any man pages matching *keyword*`(s)`, in the same manner as `apropos`(1).

-M Override the list of standard directories which **man** searches for man pages. The supplied *path* must be a colon (":") separated list of directories. This search path may also be set using the environment variable `MANPATH`. The subdirectories to be searched, and their search order, is specified by the "_subdir" line in the **man** configuration file.

-m Augment the list of standard directories which **man** searches for man pages. The supplied *path* must be a colon (":") separated list of directories. These directories will be searched before the standard directories or the directories specified using the **-M** option or the `MANPATH` environment variable. The subdirectories to be searched, and their search order, is specified by the "_subdir" line in the **man** configuration file.

-s Restrict the directories that **man** will search to the specified section. The **man** configuration file (see `man.conf`(5)) specifies the possible *section* values that are currently available.

-S Display only man pages that have the specified string in the directory part of their filenames. This allows the man page search process criteria to be narrowed without having to change the MANPATH or "_default" variables.

-w List the pathnames of the man pages which **man** would display for the specified *section* and *name* combination.

If the **-s** option is not specified, there is more than one argument, the **-k** option is not used, and the first argument is a valid section, then that argument will be used as if specified by the **-s** option.

ENVIRONMENT

`MACHINE` As some man pages are intended only for specific architectures, **man** searches any subdirectories, with the same name as the current architecture, in every directory which it searches. Machine specific areas are checked before general areas. The current machine type may be overridden by setting the environment variable `MACHINE` to the name of a specific architecture.

`MANPATH` The standard search path used by **man** may be overridden by specifying a path in the `MANPATH` environment variable. The format of the path is a colon (":") separated list of directories. The subdirectories to be searched as well as their search order is specified by the "_subdir" line in the **man** configuration file.

`PAGER` The pagination command used for writing the output. If the `PAGER` environment variable is null or not set, the standard pagination program `more`(1) will be used.

FILES

`/etc/man.conf` default man configuration file.

`/usr/{share,X11R6,pkg,local}/man/whatis.db` standard whatis/apropos database search path, set in `/etc/man.conf`.

SEE ALSO

`apropos`(1), `whatis`(1), `whereis`(1), `man.conf`(5), `mdoc`(7), `mdoc.samples`(7)

STANDARDS

man conforms to X/Open Commands and Utilities Issue 5 ("XCU5").

BUGS

The on-line man pages are, by necessity, forgiving toward stupid display devices, causing a few man pages to be not as nicely formatted as their typeset counterparts.

NAME

menuc — menu compiler

SYNOPSIS

menuc [**-o** *name*] *file*

DESCRIPTION

This implements a curses based menu system. A source file that describes menus, their options, and how to process the options is given to **menuc** and produces both a .c and a .h file that implement the menu system. The standard base name of the files is `menu_defs`. The **-o** *name* can be used to specify a different base name.

ENVIRONMENT

`MENUDEF` Can be set to point to a different set of definition files for **menuc**. The current location defaults to `/usr/share/misc`.

MENU DESCRIPTIONS

The input *file* defines static menus and options for processing those menus. It also contains comments, initial C code that is required to provide for definitions and other code necessary for the menu system, and an option declaration if dynamic menus are requested.

Comments may appear anywhere in the input *file* and are like a space in the input. They are like C comments starting with /* and ending with */. They are unlike C comments in that they may be nested. A comment does not end until a matching end comment is found.

In many places, C code is included in the definition *file*. All C code is passed verbatim to the C output file. **menuc** comments do not start in C code and comments in the C code are passed verbatim to the output. The C comments are not recognized by **menuc**. In all cases, C code starts with a left brace (*{*) and ends with the matching right brace (*}*). It is important to recognize that in code segments, any brace will be counted, even if it is in a C comment inside the code.

The *file* contains an initial (and optional) code block followed by any number of menu definition elements in any order. The initial code block usually contains includes of header files used by code in the menu code blocks later in the *file*. If `USER_MENU_INIT` is #defined, then it will be evaluated before the rest of the menu is initialised, if it evaluates to a non-zero value then the initialisation will fail. The file is free format, so the actual formatting of the input *file* is to the taste of the programmer.

All other C code that will appear in an *action*. This will be specified as ⟨*action*⟩ in later text. Such an action will appear as:

```
action <opt_endwin> <code>
```

in the *file*. The ⟨*opt_endwin*⟩, if present is:

```
(endwin)
```

and specifies that the curses **endwin**() function should be called before executing the code and then reinstating the current curses window after the code has been run. The ⟨*code*⟩ is as described above.

There are four kinds of menu definition elements. The first one just declares whether the programmer wants dynamic menus available. The default is static menus only. The static menus are the ones defined by the menu definitions and do not change at run time. The dynamic menus provide the programmer with a method to create and modify menus during the running of the program. To include dynamic menus, one needs only add the declaration:

```
allow dynamic menus;
```

The semicolon is required to terminate this declaration. This declaration may appear anywhere in the *file*, but usually appears before any menus are defined.

The next element is a code block to execute if the curses screen can not be successfully initialized. The declaration

```
error code;
```

tells the menu system to execute the associated code block if the initialization fails. If no code is provided, a default code block is used that prints

```
Could not initialize curses.
```

and exits. This element may appear anywhere in the *file* but usually appears before any menus are defined.

The next element defines default options for menus. Each menu is built from a list of options. These options include the location of the upper left corner of the menu, whether there is a "box" drawn around the menu, whether the menu is scrollable, the menu's title, whether shortcut letters are allowed, whether a standard exit option should be included in the menu and text associated with the standard exit option. The general format is:

```
default <comma separated option list>;
```

The supported options are:

x = *startx* The column number of the upper left corner of the menu window. If *startx* is -1 the menu will be centered horizontally.

y = *starty* The row number of the upper left corner of the menu window. If *starty* is negative then the menu will be placed below any message text, but in at least row -*starty*.

h = *height* Specifies the number of menu entries to be displayed. If zero, the height will be based on the number of entries.

h = *width* Specifies the width of the menu window. If zero, the width will be that of the longest menu text line.

title *text*
The specified *text* will be displayed at the top of the menu window (inside any box).

box If specified, draw a box around the menu.

clear If specified, clear the window before performing the *action*.

exit If specified add an addition option to exit the menu.

exitstring *text*
The menu label for the *exit* option. If not specified defaults to "exit".

default exit
If specified, place the cursor on the *exit* line of the menu, instead of the top line.

shortcut If specified, add alphabetic tags to each menu line.

scrollable
If specified, and the menu has more lines than will fit in its window, then only part of the menu will be displayed and the '<' and '>' keys will scroll the displayed menu lines.

always scroll
If specified, allow for the scroll message line even if the menu doesn't appear to have too many lines. Useful for dynamic menus, when the number of entries isn't known when the menu window is created..

sub menu If specified, the screen contents that the menu window overwrites are saved and restored when the menu exits.

The **box**, **clear**, **exit**, **default exit**, **shortcut**, **scrollable**, **always scroll**, and **sub menu** options can be preceded by **no** in order to negate a default.

The *text* arguments can be either a quoted text string or a name #defined to something suitable for initialising a const char * field.

The default declaration may appear multiple times. Each time, it sets the default values for menu definitions that follow in the *file*. In each menu definition, any or all of these default definitions may be overridden for that menu.

The final element is the actual static menu definitions. The format and order for a menu definition is:

```
menu <name> <options> ;
  <display action>
  <menu items>
  <exit action>
  <help text>
```

Names are unquoted strings of alpha-numeric and underscore characters. They must start with an alpha character. In C source, a menu named "foo" is appears as "MENU_foo". (Capitalization is important.) This is important, because the menu is displayed and processed by calling the function

```
process_menu (MENU_foo, arg);
```

The options are a comma separated list of options as in the "default" declaration. These override the options from the most recent default declaration.

The display action is optional and provides C code to execute at each and every time the menu is displayed for processing. If it is included, the format is:

```
display <action>;
```

The bulk of the menu definition is the specification of the menu items. The general format of a menu item is:

```
option <string>, <element_list>;
```

The ⟨*string*⟩ is the text displayed for the menu item, this must be a quoted string or a name #defined to something that will initialise a const char * field. There may be an arbitrary number of these items. (If there are shortcuts in the menu, a practical limit of 51 should be recognized. It produces shortcuts a to w, y, z, and A to Z. x is the shortcut for the exit item.)

The ⟨*element_list*⟩ is a comma separated list of what to do when the item is selected. They may appear in any order.

The first element processed when a menu item is selected is the associated action. The next element to be processed is the sub or next menu option. They are declared as:

```
next menu <name>
sub menu <name>
```

The difference between these two is that a sub menu will return to the current menu when exited. The next menu will just replace the current menu and when exited, will return to where the current menu would have gone. Only one of menu element may be used for each menu item. Finally, after processing both the action and a sub menu, the current menu will be exited if the element

```
exit
```

is specified. *Note*: If *exit* is specified, next menu will not work because the menu system will exit the *current* menu, even if current has been set by *next menu*.

After all menu items, the final two menu definition elements may appear. The exit action is optional and provides C code to execute in the process of exiting a menu. If it is included, the format is:

```
exit <action>;
```

The final part of the menu definition is the optional help string. The format is:

```
help <text>;
```

This text is displayed in a full page help window if the question mark is typed. The actual help text starts with a left brace (*{*) and ends with the matching right brace (*}*). The braces are not included in the help

string, but all other characters between them are included. Newlines in the code translate to newlines in the help text. Alternatively, the name of a const char * variable may be given.

DYNAMIC MENUS

If requested, **menuc** supports dynamic menus by allowing the user to create new menus. The related definitions for using dynamic menus are:

```
struct menudesc;

typedef
struct menu_ent {
        const char  *opt_name;
        int         opt_menu;
        int         opt_flags;
        int         (*opt_action)(struct menudesc *, void *);
} menu_ent ;

/* For opt_menu */
#define OPT_NOMENU -1

/* For opt_flags */
#define OPT_SUB    1
#define OPT_ENDWIN 2
#define OPT_EXIT   4

typedef
struct menudesc {
        const char  *title;
        int         y, x;
        int         h, w;
        int         mopt;
        int         numopts;
        int         cursel;
        int         topline;
        menu_ent    *opts;
        WINDOW      *mw;
        WINDOW      *sv_mw;
        const char  *helpstr;
        const char  *exitstr;
        void       (*post_act)(struct menudesc *, void *);
        void       (*exit_act)(struct menudesc *, void *);
        void       (*draw_line)(struct menudesc *, int, void *);
} menudesc ;

/* defines for mopt field. */
#define MC_NOEXITOPT 1
#define MC_NOBOX 2
#define MC_SCROLL 4
#define MC_NOSHORTCUT 8
#define MC_NOCLEAR 16
#define MC_DFLTEXIT 32
#define MC_ALWAYS_SCROLL 64
```

```
#define MC_SUBMENU 128

int new_menu(const char *title, menu_ent *opts, int numopts,
        int x, int y, int h, int w, int mopt,
        void (*post_act)(struct menudesc *, void *),
        void (*draw_line)(struct menudesc *, int, void *),
        void (*exit_act)(struct menudesc *, void *),
       const char *help, const char *exitstr);

void free_menu (int menu_no);
```

The *title* is the title displayed at the top of the menu. The *opts* is an array of menu entry definitions that has *numopts* elements. The programmer must build this array and fill in all of the fields before processing calling **process_menu**() for the new menu. The fields of the *opts* may change at any time. For example, *opt_name* may change as a result of selecting that option. When the menu is redisplayed, the new text is printed. Arguments, *x*, *y*, *h*, and *w* are the same as the options in the menu description. *mopt* is the boolean options. Note, box, clear, exit and shortcuts are enabled by default. You need to add option flags to turn them off or turn on scrollable menus. The options *post_act*, and *exit_act* are function pointers to the display action and the exit action. If they are NULL, no call will be made. *draw_line* will be called to display the menu line if the corresponding opt_name field is NULL. *help* is the text to display in a help screen. And finally, *exitstr* is the text for the 'exit' line of the menu. If NULL, "Exit" is used. A NULL help pointer will disable the help feature for the menu.

FILES

/usr/share/misc/menu_sys.def

EXAMPLES

The following is a simple menu definition file. It is complete in that the output of **menuc** may be compiled into a complete program. For example, if the following was in a file called example.mc, an executable program could be produced by the following commands.

```
menuc -o example example.mc
cc -o example example.c -lcurses
```

A much more complete example is available with the source distribution in a subdirectory called *testm*.

```
/* This is an example menu definition file for menuc. */

{
#include <stdio.h>
#include <unistd.h>

/* Main program! This is often in a different file. */
int
main()
  {
    process_menu (MENU_main, NULL);
    endwin();
    return 0;
  }

/* Example initialize function! */
void
init_main()
```

```
  {
  }
}

default x=20, y=10, box, scrollable, exit;

error action {
   fprintf (stderr, "Example Menu: Could not initialize curses.");
   exit(1);
};

menu main, title "Main Menu", no exit, no shortcut;
   display action { init_main(); };
   option "Option 1",
      action (endwin) {
        printf ("That was option 1!");
        sleep(3);
      };
   option "Sub Menu", sub menu othermenu;
   option "Next Menu", next menu othermenu;
   option "Quit", exit;
   help {
This is a simple help screen for an example menu definition file.
};

menu othermenu, title "Sub/Next Menu", x=5, y=5, no box;
   option "Do Nothing!", action { };
```

SEE ALSO

msgc(1)

AUTHORS

Philip A. Nelson for Piermont Information Systems Inc. Initial ideas for this were developed and implemented in Pascal at the Leiden University, Netherlands, in the summer of 1980.

BUGS

Both **menuc** and **msgc** are probably only used by **sysinst**. The features of both have been tailored for **sysinst**, and further changes are likely to occur.

NAME

merge − three-way file merge

SYNOPSIS

merge [*options*] *file1 file2 file3*

DESCRIPTION

merge incorporates all changes that lead from *file2* to *file3* into *file1*. The result ordinarily goes into *file1*. **merge** is useful for combining separate changes to an original. Suppose *file2* is the original, and both *file1* and *file3* are modifications of *file2*. Then **merge** combines both changes.

A conflict occurs if both *file1* and *file3* have changes in a common segment of lines. If a conflict is found, **merge** normally outputs a warning and brackets the conflict with <<<<<<< and >>>>>>> lines. A typical conflict will look like this:

<<<<<<< file A
lines in file A
=======
lines in file B
>>>>>>> file B

If there are conflicts, the user should edit the result and delete one of the alternatives.

OPTIONS

−A Output conflicts using the **−A** style of **diff3**(1), if supported by **diff3**. This merges all changes leading from *file2* to *file3* into *file1*, and generates the most verbose output.

−E, **−e** These options specify conflict styles that generate less information than **−A**. See **diff3**(1) for details. The default is **−E**. With **−e**, **merge** does not warn about conflicts.

−L *label*
This option may be given up to three times, and specifies labels to be used in place of the corresponding file names in conflict reports. That is, **merge −L x −L y −L z a b c** generates output that looks like it came from files **x**, **y** and **z** instead of from files **a**, **b** and **c**.

−p Send results to standard output instead of overwriting *file1*.

−q Quiet; do not warn about conflicts.

−V Print version number.

DIAGNOSTICS

Exit status is 0 for no conflicts, 1 for some conflicts, 2 for trouble.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

diff3(1), diff(1), rcsmerge(1), co(1).

BUGS

It normally does not make sense to merge binary files as if they were text, but **merge** tries to do it anyway.

NAME

mesg — display (do not display) messages from other users

SYNOPSIS

mesg [**n** | **y**]

DESCRIPTION

The **mesg** utility is invoked by a user to control write access others have to the terminal device associated with the standard error output. Write access is allowed by default, and programs such as talk(1) and write(1) may display messages on the terminal.

Options available:

n Disallows messages.

y Permits messages to be displayed.

If no arguments are given, **mesg** displays the present message status to the standard error output.

The **mesg** utility exits with one of the following values:

0 Messages are allowed.
1 Messages are not allowed.
>1 An error has occurred.

FILES

/dev/[pt]ty[pq]?

SEE ALSO

biff(1), talk(1), write(1)

HISTORY

A **mesg** command appeared in Version 6 AT&T UNIX.

NAME

midiplay — play MIDI and RMID files

SYNOPSIS

midiplay [**-d** *devno*] [**-f** *file*] [**-l**] [**-m**] [**-p** *pgm*] [**-q**] [**-t** *tempo*] [**-v**] [**-x**] [*file ...*]

DESCRIPTION

The **midiplay** command plays MIDI and RMID files using the sequencer device. If no file name is given it will play from standard input, otherwise it will play the named files.

RMID files are Standard MIDI Files embedded in a RIFF container and can usually be found with the 'rmi' extension. They contain some additional information in other chunks which are not parsed by **midiplay** yet.

The program accepts the following options:

-d *devno* specifies the number of the MIDI device used for output (as listed by the **-l** flag). There is no way at present to have **midiplay** map playback to more than one device. The default is device is given by environment variable `MIDIUNIT`.

-f *file* specifies the name of the sequencer device.

-l list the possible devices without playing anything.

-m show MIDI file meta events (copyright, lyrics, etc).

-p *pgm* force all channels to play with the single specified program (or instrument patch, range 1-128). Program change events in the file will be suppressed. There is no way at present to have **midiplay** selectively map channels or instruments.

-q specifies that the MIDI file should not be played, just parsed.

-t *tempo-adjust*
specifies an adjustment (in percent) to the tempi recorded in the file. The default of 100 plays as specified in the file, 50 halves every tempo, and so on.

-v be verbose. If the flag is repeated the verbosity increases.

-x play a small sample sound instead of the a file.

A file containing no tempo indication will be played as if it specified 150 beats per minute. You have been warned.

ENVIRONMENT

`MIDIUNIT` the default number of the MIDI device used for output. The default is 0.

FILES

`/dev/music` MIDI sequencer device

SEE ALSO

`midi`(4)

HISTORY

The **midiplay** command first appeared in NetBSD 1.4.

BUGS

It may take a long while before playing stops when **midiplay** is interrupted, as the data already buffered in the sequencer will contain timing events.

NAME

mixerctl — control audio mixing

SYNOPSIS

mixerctl [**-d** *file*] [**-n**] [**-v**] **-a**
mixerctl [**-d** *file*] [**-n**] [**-v**] *name ...*
mixerctl [**-d** *file*] [**-n**] **-w** *name=value ...*
mixerctl [**-d** *file*] [**-n**] **-w** *name++ ...*
mixerctl [**-d** *file*] [**-n**] **-w** *name-- ...*
mixerctl [**-d** *file*] [**-n**] **-w** *name+=value ...*
mixerctl [**-d** *file*] [**-n**] **-w** *name-=value ...*

DESCRIPTION

The **mixerctl** command displays or sets various audio system mixing variables. If a list of variables is present on the command line, then **mixerctl** prints the current value of those variables for the specified device. If the **-a** flag is specified, all variables for the device are printed. If the **-w** flag is specified, **mixerctl** attempts to set the specified variables to the given values.

The **-d** flag can be used to give an alternative mixer device. The default is `/dev/mixer`.

The **-n** flag suppresses printing of the variable name.

The **-v** flag shows the possible values of enumeration and set valued variables. Enumerated values are shown in brackets

and set values are shown in curly braces (“{}”).

The exact set of controls that can be manipulated depends on the mixer. The general format (in both getting and setting a value) is
class.name = value
class can have values like `inputs` or `outputs`, indicating that the control affects the input or output of the mixer, respectively. *name* indicates what part of the mixer the control affects. Continuous mixer values, e.g., volume, have numeric values in the range 0−255. If *value* can be set for each channel independently, the values are printed separated by commas. Discrete mixer values, e.g., the recording source, have symbolic names. Depending on the mixer it may either be an enumeration or a set.

The suffixes ++ and -- can be used to step through the values of a mixer control. For numeric controls, these operators increase or decrease, respectively, the value by an amount (the delta) suitable to make the control assume the next possible value.

The operators += and -= change the value of a mixer control by the indicated number of steps.

ENVIRONMENT

`MIXERDEVICE` the mixer device to use.

FILES

`/dev/mixer` mixer audio device
`/etc/mixerctl.conf` mixer configuration file

EXAMPLES

The command

```
mixerctl -a -v
```

can produce

```
inputs.mic=0,0 volume
inputs.mic.mute=off  [ off on ]
inputs.cd=220,220 volume
inputs.cd.mute=off  [ off on ]
inputs.dac=220,220 volume
inputs.dac.mute=off  [ off on ]
record.record=220,220 volume
record.record.source=mic  [ mic cd dac ]
monitor.monitor=0 volume
```

SEE ALSO

`audioctl`(1), `audio`(4), `mixerctl.conf`(5), `sysctl`(8)

HISTORY

The **`mixerctl`** command first appeared in NetBSD 1.3.

COMPATIBILITY

The old **`-f`** flag is still supported. This support will be removed eventually.

NAME

mkdep — construct Makefile dependency list

SYNOPSIS

mkdep [**-aDdopq**] [**-f** *file*] [**-s** *suffixes*] -- [*flags*] *file ...*

DESCRIPTION

mkdep takes a set of flags for the C compiler and a list of C source files as arguments and constructs a set of include file dependencies which are written into the file ‘‘.depend’’. An example of its use in a Makefile might be:

```
CFLAGS= -O -I../include
SRCS= file1.c file2.c

depend:
        mkdep -- ${CFLAGS} ${SRCS}
```

where the macro SRCS is the list of C source files and the macro CFLAGS is the list of flags for the C compiler.

The options are as follows:

-a Append to the output file, so that multiple **mkdep**’s may be run from a single Makefile.

-D Post process (as **-d**) but read the list of filenames from stdin.

-d Post process and merge previously created (for example by “cc -MD x.c”) depend files into a single file.

-f Write the include file dependencies to *file*, instead of the default ‘‘.depend’’.

-o Add an additional .OPTIONAL line for each dependant file.

-p Cause **mkdep** to produce dependencies of the form:

```
program: program.c
```

so that subsequent makes will produce *program* directly from its C module rather than using an intermediate `.o` module. This is useful for programs whose source is contained in a single module. **-p** is equivalent to specifying a null suffix with **-s**.

-q Do not print a warning for inaccessible files when **-d** is given.

-s Expand each target filename to a list, replacing the ‘`.o`’ suffix with each element of *suffixes*. The list of suffixes may be space or comma separated.

FILES

`.depend` File containing list of dependencies.

SEE ALSO

`cc`(1), `cpp`(1), `make`(1)

HISTORY

The **mkdep** command appeared in 4.3BSD−Tahoe.

NAME

mkdir — make directories

SYNOPSIS

mkdir [**-p**] [**-m** *mode*] *directory_name ...*

DESCRIPTION

mkdir creates the directories named as operands, in the order specified, using mode `rwxrwxrwx (0777)` as modified by the current `umask`(2).

The options are as follows:

-m Set the file permission bits of the final created directory to the specified mode. The mode argument can be in any of the formats specified to the `chmod`(1) utility. If a symbolic mode is specified, the operation characters "+" and "-" are interpreted relative to an initial mode of "a=rwx".

-p Create intermediate directories as required. If this option is not specified, the full path prefix of each operand must already exist. Intermediate directories are created with permission bits of `rwxrwxrwx (0777)` as modified by the current umask, plus write and search permission for the owner. Do not consider it an error if the argument directory already exists.

The user must have write permission in the parent directory.

EXIT STATUS

mkdir exits 0 if successful, and >0 if an error occurred.

SEE ALSO

`chmod`(1), `rmdir`(1), `mkdir`(2), `umask`(2)

STANDARDS

The **mkdir** utility is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

NAME

mkfifo — make fifos

SYNOPSIS

mkfifo [**-m** *mode*] *fifo_name ...*

DESCRIPTION

mkfifo creates the fifos requested, in the order specified, using mode `0666` modified by the current `umask`(2).

The options are as follows:

-m Set the file permission bits of newly-created fifos to *mode*. The mode is specified as in `chmod`(1). In symbolic mode strings, the "+" and "-" operators are interpreted relative to an assumed initial mode of "a=rw"

mkfifo requires write permission in the parent directory.

mkfifo exits 0 if successful, and >0 if an error occurred.

SEE ALSO

`mkdir`(1), `rm`(1), `mkfifo`(2), `mknod`(8)

STANDARDS

The **mkfifo** utility is expected to be IEEE Std 1003.2-1992 ("POSIX.2") compliant.

HISTORY

mkfifo command appeared in 4.4BSD.

NAME

mkfilters − generate a minimal firewall ruleset for ipfilter

SYNOPSIS

mkfilters

DESCRIPTION

mkfilters is a perl script that generates a minimal filter rule set for use with **ipfilter** by parsing the output of **ifconfig**.

FILES

/usr/share/examples/ipf/mkfilters

SEE ALSO

ipf(8), ipf(5), ipfilter(4), ifconfig(8)

NAME

mklocale — make LC_CTYPE locale files

SYNOPSIS

mklocale [**-d**] < *src-file* > *language/LC_CTYPE*
mklocale [**-d**] **-o** *language/LC_CTYPE* *src-file*

DESCRIPTION

The **mklocale** utility reads an LC_CTYPE source file from standard input and produces an LC_CTYPE binary file on standard output suitable for placement in /usr/share/locale/<language>/LC_CTYPE.

The format of *src-file* is quite simple. It consists of a series of lines which start with a keyword and have associated data following. C style comments are used to place comments in the file.

Following options are available:

-d Turns on debugging messages.

-o Specify output file.

Besides the keywords which will be listed below, the following are valid tokens in *src-file*:

RUNE A RUNE may be any of the following:

'x'	The ASCII character *x*.
'\x'	The ANSI C character *\x* where *\x* is one of \a, \b, \f, \n, \r, \t, or \v.
*0x[0-9a-z]**	A hexadecimal number representing a rune code.
*0[0-7]**	An octal number representing a rune code.
*[1-9][0-9]**	A decimal number representing a rune code.

STRING A string enclosed in double quotes (").

THRU Either . . . or -. Used to indicate ranges.

literal The follow characters are taken literally:

< ([	Used to start a mapping. All are equivalent.
>)]	Used to end a mapping. All are equivalent.
:	Used as a delimiter in mappings.

Key words which should only appear once are:

ENCODING Followed by a STRING which indicates the encoding mechanism to be used for this locale. The current encodings are:

NONE No translation and the default.

UTF2 Universal character set Transformation Format adopted from **Plan 9 from Bell Labs**.

EUC EUC encoding as used by several vendors of UNIX systems.

VARIABLE This keyword must be followed by a single tab or space character, after which encoding specific data is placed. Currently only the EUC encoding requires variable data.

`INVALID` A single `RUNE` follows and is used as the invalid rune for this locale.

The following keywords may appear multiple times and have the following format for data:

⟨`RUNE1 RUNE2`⟩	`RUNE1` is mapped to `RUNE2`.
⟨`RUNE1 THRU RUNEn: RUNE2`⟩	Runes `RUNE1` through `RUNEn` are mapped to `RUNE2` through `RUNE2` + n-1.

`MAPLOWER` Defines the tolower mappings. `RUNE2` is the lower case representation of `RUNE1`.

`MAPUPPER` Defines the toupper mappings. `RUNE2` is the upper case representation of `RUNE1`.

`TODIGIT` Defines a map from runes to their digit value. `RUNE2` is the integer value represented by `RUNE1`. For example, the ASCII character '0' would map to the decimal value 0. Only values up to 255 are allowed.

The following keywords may appear multiple times and have the following format for data:

`RUNE`	This rune has the property defined by the keyword.
`RUNE1 THRU RUNEn`	All the runes between and including `RUNE1` and `RUNEn` have the property defined by the keyword.

`ALPHA` Defines runes which are alphabetic, printable, and graphic.

`CONTROL` Defines runes which are control characters.

`DIGIT` Defines runes which are decimal digits, printable, and graphic.

`GRAPH` Defines runes which are graphic and printable.

`LOWER` Defines runes which are lower case, printable, and graphic.

`PUNCT` Defines runes which are punctuation, printable, and graphic.

`SPACE` Defines runes which are spaces.

`UPPER` Defines runes which are upper case, printable, and graphic.

`XDIGIT` Defines runes which are hexadecimal digits, printable, and graphic.

`BLANK` Defines runes which are blank.

`PRINT` Defines runes which are printable.

`IDEOGRAM` Defines runes which are ideograms, printable, and graphic.

`SPECIAL` Defines runes which are special characters, printable, and graphic.

`PHONOGRAM` Defines runes which are phonograms, printable, and graphic.

`SWIDTHn` Defines runes with specific glyph width. *n* takes 0 to 3.

`CHARSET` Controls character set for subsequent runes.

SEE ALSO

`setlocale`(3), `nls`(7)

HISTORY

The **`mklocale`** utility first appeared in 4.4BSD.

BUGS

The **mklocale** utility is overly simplistic.

We should switch to **localedef** and its file format, which is more standard.

NAME

mkstr — create an error message file by massaging C source

SYNOPSIS

mkstr [**-**] *messagefile prefix file ...*

DESCRIPTION

mkstr creates files containing error messages extracted from C source, and restructures the same C source, to use the created error message file. The intent of **mkstr** was to reduce the size of large programs and reduce swapping (see **BUGS** section below).

mkstr processes each of the specified *files*, placing a restructured version of the input in a file whose name consists of the specified *prefix* and the original name. A typical usage of **mkstr** is

```
mkstr pistrings xx *.c
```

This command causes all the error messages from the C source files in the current directory to be placed in the file *pistrings* and restructured copies of the sources to be placed in files whose names are prefixed with *xx*.

Options:

- Error messages are placed at the end of the specified message file for recompiling part of a large **mkstr** ed program.

mkstr finds error messages in the source by searching for the string `'error("'` in the input stream. Each time it occurs, the C string starting at the '"' is stored in the message file followed by a null character and a new-line character; The new source is restructured with `lseek`(2) pointers into the error message file for retrieval.

```
char efilname = "/usr/lib/pi_strings";
int efil = -1;

error(a1, a2, a3, a4)
{
        char buf[256];

        if (efil < 0) {
                efil = open(efilname, 0);
                if (efil < 0) {
oops:
                        perror(efilname);
                        exit 1 ;
                }
        }
        if (lseek(efil, a1, 0) < 0 || read(efil, buf, 256) ≤ 0)
                goto oops;
        printf(buf, a2, a3, a4);
}
```

SEE ALSO

`xstr`(1), `lseek`(2)

HISTORY

mkstr appeared in 3.0BSD.

BUGS

mkstr was intended for the limited architecture of the PDP-11 family. Very few programs actually use it. It is not an efficient method, the error messages should be stored in the program text.

NAME

mktemp — make temporary file name (unique)

SYNOPSIS

mktemp [**-dqu**] { **-t** *prefix* | *template* **...** }

DESCRIPTION

The **mktemp** utility takes each of the given file name templates and overwrites a portion of it to create a file name. This file name is unique and suitable for use by the application. The template may be any file name with some number of 'Xs' appended to it, for example `/tmp/temp.XXXX`. The trailing 'Xs' are replaced with the current process number and/or a unique letter combination. The number of unique file names **mktemp** can return depends on the number of 'Xs' provided; six 'Xs' will result in **mktemp** testing roughly 26 ** 6 combinations.

If **mktemp** can successfully generate a unique file name, the file is created with mode 0600 (unless the **-u** flag is given) and the filename is printed to standard output.

If the **-t** *prefix* option is given, **mktemp** will generate a template string based on the *prefix* and the `TMPDIR` environment variable, if set. The default location if `TMPDIR` is not set is `/tmp`. The template string created will consist of the *prefix* followed by a '.' and an eight character unique letter combination. 'Xs' in the *prefix* string will be treated as literal. If an additional *template* argument is passed, a second file will be created. Care should be taken to ensure that it is appropriate to use an environment variable potentially supplied by the user.

Any number of temporary files may be created in a single invocation using multiple *template* arguments, also a single one based on the internal template with the **-t** option value as filename prefix.

At least one *template* argument or the **-t** option must be present.

mktemp is provided to allow shell scripts to safely use temporary files. Traditionally, many shell scripts take the name of the program with the pid as a suffix and use that as a temporary file name. This kind of naming scheme is predictable and the race condition it creates is easy for an attacker to win. A safer, though still inferior, approach is to make a temporary directory using the same naming scheme. While this does allow one to guarantee that a temporary file will not be subverted, it still allows a simple denial of service attack. For these reasons it is suggested that **mktemp** be used instead.

OPTIONS

The available options are as follows:

-d Make a directory instead of a file.

-q Fail silently if an error occurs. This is useful if a script does not want error output to go to standard error.

-t *prefix*
Generate a template (using the supplied *prefix* and `TMPDIR` if set) to create a filename template. If **-t** *prefix* and *template* are both given, *prefix* will not apply to *template*.

-u Operate in "unsafe" mode. The temp file will be unlinked before **mktemp** exits. This is slightly better than `mktemp`(3) but still introduces a race condition. Use of this option is not encouraged.

EXIT STATUS

The **mktemp** utility exits with a value of 0 on success, and 1 on any failure.

EXAMPLES

The following sh(1) fragment illustrates a simple use of **mktemp** where the script should quit if it cannot get a safe temporary file.

```
TMPFILE=‘mktemp /tmp/${0##*/}.XXXXXX‘ || exit 1
echo "program output" >> $TMPFILE
```

To allow the use of $TMPDIR:

```
TMPFILE=‘mktemp -t ${0##*/}‘ || exit 1
echo "program output" >> $TMPFILE
```

In this case, we want the script to catch the error itself.

```
TMPFILE=‘mktemp -q /tmp/${0##*/}.XXXXXX‘
if [ $? -ne 0 ]; then
        echo "$0: Can’t create temp file, exiting..."
        exit 1
fi
```

SEE ALSO

mkdtemp(3), mkstemp(3), mktemp(3), environ(7)

HISTORY

The **mktemp** utility appeared in NetBSD 1.5. It has been imported from FreeBSD, the idea and the manual page were taken from OpenBSD.

NAME

mopchk — MOP Check Utility

SYNOPSIS

mopchk [**-a**] [**-v**] [*filename*]

DESCRIPTION

mopchk shows information about which devices are known, version of mopd suite or information about a MOP-image.

If *filename* is given, information about the MOP-image is read from the header of the file.

OPTIONS

-a Show all the Ethernets attached to the system.

-v Show version of the mopd suite.

SEE ALSO

mopcopy(1), mopprobe(1), moptrace(1), mopd(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

BUGS

In some implementations the same interface can occur more than once.

NAME

mopcopy — Create MOP image from another executable format

SYNOPSIS

mopcopy *infile outfile*

DESCRIPTION

mopcopy is used to convert a file from another executable format to a MOP image.

Elf32 and a.out VAX images are currently supported.

SEE ALSO

mopchk(1), mopprobe(1), moptrace(1), a.out(5), elf(5), mopd(8)

AUTHORS

Lloyd Parkes
Jason R. Thorpe

NAME

mopprobe — MOP Probe Utility

SYNOPSIS

mopprobe -a [**-3** | **-4**]
mopprobe [**-3** | **-4**] *interface*

DESCRIPTION

mopprobe prints the ethernet address and nodename of MOP SID message on the Ethernet connected to *interface* or all known interfaces if '**-a**' is given.

OPTIONS

-a Listen on all the Ethernets attached to the system. If '**-a**' is omitted, an interface must be specified.

-3 Ignore MOP V3 messages (Ethernet II).

-4 Ignore MOP V4 messages (Ethernet 802.3).

SEE ALSO

mopchk(1), mopcopy(1), moptrace(1), mopd(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

moptrace — MOP Trace Utility

SYNOPSIS

moptrace [**-a**] [**-d**] [**-3** | **-4**]
moptrace [**-d**] [**-3** | **-4**] [*interface*]

DESCRIPTION

moptrace prints the contents of MOP packages on the Ethernet connected to *interface* or all known interfaces if '**-a**' is given.

OPTIONS

-a Listen on all the Ethernets attached to the system. If '**-a**' is omitted, an interface must be specified.

-d Run in debug mode, with all the output to stderr.

-3 Ignore MOP V3 messages (Ethernet II).

-4 Ignore MOP V4 messages (Ethernet 802.3).

SEE ALSO

mopchk(1), mopcopy(1), mopprobe(1), mopd(8)

DECnet Digital Network Architecture Phase IV, Maintenance Operations Functional Specification V3.0.0, AA-X436A-TK.

DECnet Digital Network Architecture, Maintenance Operations Protocol Functional Specification V4.0.0, EK-DNA11-FS-001.

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

more — file perusal filter for CRT viewing

SYNOPSIS

more [**-ceinus**] [**-t** *tag*] [**-x** *tabs*] [**-/** *pattern*] [**-#**] [*file . . .*]

DESCRIPTION

The **more** command is a filter for paging through text one screenful at a time. It uses termcap(3) so it can run on a variety of terminals. There is even limited support for hardcopy terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are prefixed with an up-arrow). *File* may be a single dash (‘‘-’’), implying stdin.

OPTIONS

Command line options are described below. Options are also taken from the environment variable MORE (make sure to precede them with a dash (‘‘-’’)) but command line options will override them.

-c Normally, **more** will repaint the screen by scrolling from the bottom of the screen. If the **-c** option is set, when **more** needs to change the entire display, it will paint from the top line down.

-e Normally, if displaying a single file, **more** exits as soon as it reaches end-of-file. The **-e** option tells more to exit if it reaches end-of-file twice without an intervening operation. If the file is shorter than a single screen **more** will exit at end-of-file regardless.

-i The **-i** option causes searches to ignore case; that is, uppercase and lowercase are considered identical.

-n The **-n** flag suppresses line numbers. The default (to use line numbers) may cause **more** to run more slowly in some cases, especially with a very large input file. Suppressing line numbers with the **-n** flag will avoid this problem. Using line numbers means: the line number will be displayed in the **=** command, and the **v** command will pass the current line number to the editor.

-s The **-s** option causes consecutive blank lines to be squeezed into a single blank line.

-t The **-t** option, followed immediately by a tag, will edit the file containing that tag. For more information, see the ctags(1) command.

-u By default, **more** treats backspaces and CR-LF sequences specially. Backspaces which appear adjacent to an underscore character are displayed as underlined text. Backspaces which appear between two identical characters are displayed as emboldened text. CR-LF sequences are compressed to a single linefeed character. The **-u** option causes backspaces to always be displayed as control characters, i.e. as the two character sequence ‘‘^H’’, and CR-LF to be left alone.

-x The **-x** option sets tab stops every *N* positions. The default for *N* is 8.

-/ The **-/** option specifies a string that will be searched for before each file is displayed.

COMMANDS

Interactive commands for **more** are based on vi(1). Some commands may be preceded by a decimal number, called N in the descriptions below. In the following descriptions, ^X means control-X.

h Help: display a summary of these commands. If you forget all the other commands, remember this one.

SPACE or **f** or **^F**
Scroll forward N lines, default one window. If N is more than the screen size, only the final screenful is displayed.

b or **^B** Scroll backward N lines, default one window (see option -z below). If N is more than the screen size, only the final screenful is displayed.

j or **RETURN**
Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

k Scroll backward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

d or **^D** Scroll forward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

u or **^U** Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

g Go to line N in the file, default 1 (beginning of file).

G Go to line N in the file, default the end of the file.

p or **%** Go to a position N percent into the file. N should be between 0 and 100. (This works if standard input is being read, but only if **more** has already read to the end of the file. It is always fast, but not always useful.)

r or **^L** Repaint the screen.

R Repaint the screen, discarding any buffered input. Useful if the file is changing while it is being viewed.

m Followed by any lowercase letter, marks the current position with that letter.

' (Single quote.) Followed by any lowercase letter, returns to the position which was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed, or the beginning of the file if no such movements have occurred. All marks are lost when a new file is examined.

/pattern Search forward in the file for the N-th line containing the pattern. N defaults to 1. The pattern is a regular expression, as recognized by ed(1). The search starts at the second line displayed.

?*pattern* Search backward in the file for the N-th line containing the pattern. The search starts at the line immediately before the top line displayed.

/!*pattern*
Like /, but the search is for the N-th line which does NOT contain the pattern.

?!*pattern*
Like ?, but the search is for the N-th line which does NOT contain the pattern.

n Repeat previous search, for N-th line containing the last pattern (or NOT containing the last pattern, if the previous search was /! or ?!).

E[*filename*]
Examine a new file. If the filename is missing, the "current" file (see the N and P commands below) from the list of files in the command line is re-examined. If the filename is a pound sign (#), the previously examined file is re-examined.

N or **:n** Examine the next file (from the list of files given in the command line). If a number N is specified (not to be confused with the command N), the N-th next file is examined.

P or **:p** Examine the previous file. If a number N is specified, the N-th previous file is examined.

:t Go to supplied tag.

v Invokes an editor to edit the current file being viewed. The editor is taken from the environment variable EDITOR, or defaults to vi(1).

= or **^G** These options print out the number of the file currently being displayed relative to the total number of files there are to display, the current line number, the current byte number and the total bytes to display, and what percentage of the file has been displayed. If **more** is reading from stdin, or the file is shorter than a single screen, some of these items may not be available. Note, all of these items reference the first byte of the last line displayed on the screen.

q or **:q** or **ZZ**
Exits **more**.

ENVIRONMENT

The **more** command uses the following environment variables, if they exist:

MORE This variable may be set with favored options to **more**.

EDITOR Specify default editor.

SHELL Current shell in use (normally set by the shell at login time).

TERM Specifies terminal type, used by more to get the terminal characteristics necessary to manipulate the screen.

SEE ALSO

ctags(1), vi(1)

HISTORY

The **more** command appeared in 3.0BSD.

AUTHORS

This software is derived from software contributed to Berkeley by Mark Nudleman.

NAME

msconfig — Show or change the middle button emulation mode

SYNOPSIS

msconfig [**-f** *mouse-device*] [*on*|*off*]

DESCRIPTION

msconfig is used to modify or examine the state of the middle-button emulation mode of the NetBSD/Atari mouse driver. When **msconfig** is used without the *on*|*off* parameter, it prints the current state of the emulation mode. The default mouse device is /dev/mouse. This can be overridden by means of the **-f** option.

When the emulation is turned on, a middle-button event can be generated by simultaneously pressing the left and right buttons. When turned off, a middle-button event is generated by pushing the fire-button of joystick1. Note that you'll need a special type of mouse to use this.

HISTORY

The **msconfig** command appeared in NetBSD 1.3

NAME

mset — retrieve ASCII to IBM 3270 keyboard map

SYNOPSIS

mset [**-picky**] [**-shell**] [*keyboardname*]

DESCRIPTION

mset retrieves mapping information for the ASCII keyboard to IBM 3270 terminal special functions. Normally, these mappings are found in `/usr/share/misc/map3270` (see `map3270`(5)). This information is used by the **tn3270** command (see `tn3270`(1)).

The default **mset** output can be used to store the mapping information in the process environment in order to avoid scanning `map3270` each time **tn3270** is invoked. To do this, place the following command in your `.login` file:

```
set noglob; setenv MAP3270 "`mset`"; unset noglob
```

If the *keyboardname* argument is not supplied, **mset** attempts to determine the name of the keyboard the user is using, by checking the `KEYBD` environment variable. If the `KEYBD` environment variable is not set, then **mset** uses the user's terminal type from the environment variable `TERM` as the keyboard name. Normally, **mset** then uses the file `map3270`(5) to find the keyboard mapping for that terminal. However, if the environment variable `MAP3270` exists and contains the entry for the specified keyboard, then that definition is used. If the value of `MAP3270` begins with a slash ('/') then it is assumed to be the full pathname of an alternative mapping file and that file is searched first. In any case, if the mapping for the keyboard is not found in the environment, nor in an alternative map file, nor in the standard map file, then the same search is performed for an entry for a keyboard with the name *unknown*. If that search also fails, then a default mapping is used.

The arguments to **mset** are:

-picky When processing the various `map3270` entries (for the user's keyboard, and all those encountered before the one for the user's keyboard), **mset** normally will not complain about entries for unknown functions (like "PFX1"); the **-picky** argument causes **mset** to issue warning messages about these unknown entries.

-shell If the `map3270` entry is longer than the shell's 1024 environmental variable length limit, the default **mset** output cannot be used to store the mapping information in the process environment to avoid scanning `map3270` each time **tn3270** is invoked. The **-shell** argument causes **mset** to generate shell commands to set the environmental variables `MAP3270`, `MAP3270A`, and so on, breaking up the entry to fit within the shell environmental variable length limit. To set these variables, place the following command in your `.login` file:

```
mset -shell > tmp ; source tmp ; /bin/rm tmp
```

keyboardname

When searching for the `map3270` entry that matches the user's keyboard, **mset** will use *keyboardname* instead of determining the keyboard name from the `KEYBD` or `TERM` environmental variables.

FILES

`/usr/share/misc/map3270` keyboard mapping for known keyboards

SEE ALSO

`tn3270`(1), `map3270`(5)

HISTORY

The **mset** command appeared in 4.3BSD.

NAME

msgattrib − attribute matching and manipulation on message catalog

SYNOPSIS

msgattrib [*OPTION*] [*INPUTFILE*]

DESCRIPTION

Filters the messages of a translation catalog according to their attributes, and manipulates the attributes.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE
: input PO file

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If no input file is given or if it is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Message selection:

−−translated
: keep translated, remove untranslated messages

−−untranslated
: keep untranslated, remove translated messages

−−no−fuzzy
: remove ’fuzzy’ marked messages

−−only−fuzzy
: keep ’fuzzy’ marked messages

−−no−obsolete
: remove obsolete #˜ messages

−−only−obsolete
: keep obsolete #˜ messages

Attribute manipulation:

−−set−fuzzy
: set all messages ’fuzzy’

−−clear−fuzzy
: set all messages non-’fuzzy’

−−set−obsolete
: set all messages obsolete

−−clear−obsolete
: set all messages non-obsolete

−−only−file=*FILE*.po
: manipulate only entries listed in FILE.po

−−ignore−file=*FILE*.po
: manipulate only entries not listed in FILE.po

−−fuzzy
: synonym for **−−only−fuzzy −−clear−fuzzy**

−−obsolete
synonym for **−−only−obsolete −−clear−obsolete**

Input file syntax:

−P, **−−properties−input**
input file is in Java .properties syntax

−−stringtable−input
input file is in NeXTstep/GNUstep .strings syntax

Output details:

−e, **−−no−escape**
do not use C escapes in output (default)

−E, **−−escape**
use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−i, **−−indent**
write the .po file using indented style

−−no−location
do not write '#: filename:line' lines

−n, **−−add−location**
generate '#: filename:line' lines (default)

−−strict
write out strict Uniforum conforming .po file

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

The full documentation for **msgattrib** is maintained as a Texinfo manual. If the **info** and **msgattrib** programs are properly installed at your site, the command

info msgattrib

should give you access to the complete manual.

NAME

msgc, **msg_window**, **msg_string**, **msg_clear**, **msg_standout**, **msg_standend**, **msg_display**, **msg_display_add**, **msg_prompt**, **msg_prompt_add**, **msg_prompt_win**, **msg_prompt_noecho**, **msg_row**, **msg_table_add** — simple message list compiler

SYNOPSIS

msgc [**-o** *name*] *file*

```
#include "msg_defs.h"

void
msg_window(WINDOW *window);

const char *
msg_string(msg msg_no);

void
msg_clear(void);

void
msg_standout(void);

void
msg_standend(void);

void
msg_display(msg msg_no, ...);

void
msg_display_add(msg msg_no, ...);

void
msg_prompt(msg msg_no, const char *def, char *val, int max_chars, ...);

void
msg_prompt_add(msg msg_no, const char *def, char *val, int max_chars, ...);

void
msg_prompt_win(msg msg_no, WINDOW *win, const char *def, char *val,
    int max_chars, ...);

void
msg_prompt_noecho(msg msg_no, const char *def, char *val, int max_chars,
    ...);

int
msg_row(void);

void
msg_table_add(msg msg_no, ...);
```

DESCRIPTION

This implements a curses based message display system. A source file that lists messages with associated names is given to **msgc** and produces both a .c and a .h file that implement the menu system. The standard root name of the files is `msg_defs`. The **-o** *name* can be used to specify a different root name.

ENVIRONMENT

`MSGDEF` Can be set to point to a different set of definition files for **`msgc`**. The current location defaults to `/usr/share/misc`.

FILES

`/usr/share/misc/msg_sys.def`

SOURCE DESCRIPTION

The format is very simple. Each message is started with the word 'message' followed by the name of the message. The body of the message is next and is started by a { and closed by a }. The braces are not part of the message. Everything, including newlines between the braces are part of the message.

MESSAGE FUNCTIONS

The defined messages are used through calls routines that manipulate the messages. You first need to set the `curses`(3) environment up and then tell the message system which window to use for displaying message by calling the function **`msg_window`**().

All variable argument lists in the functions are used as are arguments to `sprintf`(3). The messages may have `sprintf`(3) conversions in them and the corresponding parameters should match. Messages are identified by name using the notation 'MENU_name' where "name" is the name in the message source file. (The definitions are accessed by including the generated .h file into a source file wanting to use the message routines.)

The function **`msg_string`**() just returns a pointer to the actual message string. The functions **`msg_clear`**(), **`msg_standout`**() and **`msg_standend`**() respectively clear the message window, set standout mode and clear standout mode.

The functions **`msg_display`**() and **`msg_display_add`**() cause a defined message to be displayed in the message window and does the requested conversions before printing. The difference is that **`msg_display`**() clears the window before displaying the message. These functions fill paragraphs for readability. The **`msg_table_add`**() function behaves like **`msg_display_add`**() but does not fill text.

The remaining functions deal with a prompt facility. A prompt message is either taken from the message directory or from a given string. The message is processed with `sprintf`(3) and then displayed. If the parameter *`def`* is non-`NULL` and not a string of zero length, a default value is printed in brackets. The user is allowed to type in a response. If the user types just the newline character, the default is returned in the value. The parameter *`max_chars`* is the length of the parameter *`val`*, where the results are stored. The parameters *`def`* and *`val`* may point to the same character array. If the default is chosen, the character array is not changed. The functions **`msg_echo`**() and **`msg_noecho`**() control whether the prompt routine echo or don't echo the input that is typed by the user.

`msg_prompt_win`() uses the specified curses window instead of the default one.

`msg_row`() return the current row - i.e.: getcury(msg_win) + getbegy(msg_win).

AUTHORS

Philip A. Nelson for Piermont Information Systems Inc.

NAME

msgcat − combines several message catalogs

SYNOPSIS

msgcat [*OPTION*] [*INPUTFILE*]...

DESCRIPTION

Concatenates and merges the specified PO files. Find messages which are common to two or more of the specified PO files. By using the **−−more−than** option, greater commonality may be requested before messages are printed. Conversely, the **−−less−than** option may be used to specify less commonality before messages are printed (i.e. **−−less−than**=*2* will only print the unique messages). Translations, comments and extract comments will be cumulated, except that if **−−use−first** is specified, they will be taken from the first PO file to define them. File positions from all PO files will be cumulated.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE ...
: input files

−f, **−−files−from**=*FILE*
: get list of input files from FILE

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If input file is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Message selection:

-<, **−−less−than**=*NUMBER*
: print messages with less than this many definitions, defaults to infinite if not set

->, **−−more−than**=*NUMBER*
: print messages with more than this many definitions, defaults to 0 if not set

−u, **−−unique**
: shorthand for **−−less−than**=*2*, requests that only unique messages be printed

Input file syntax:

−P, **−−properties−input**
: input files are in Java .properties syntax

−−stringtable−input
: input files are in NeXTstep/GNUstep .strings syntax

Output details:

−t, **−−to−code**=*NAME*
: encoding for output

−−use−first
: use first available translation for each message, don’t merge several translations

−e, **−−no−escape**
: do not use C escapes in output (default)

−E, **−−escape**
: use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−i, **−−indent**
write the .po file using indented style

−−no−location
do not write '#: filename:line' lines

−n, **−−add−location**
generate '#: filename:line' lines (default)

−−strict
write out strict Uniforum conforming .po file

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgcat** is maintained as a Texinfo manual. If the **info** and **msgcat** programs are properly installed at your site, the command

info msgcat

should give you access to the complete manual.

NAME

msgcmp − compare message catalog and template

SYNOPSIS

msgcmp [*OPTION*] *def.po ref.pot*

DESCRIPTION

Compare two Uniforum style .po files to check that both contain the same set of msgid strings. The def.po file is an existing PO file with the translations. The ref.pot file is the last created PO file, or a PO Template file (generally created by xgettext). This is useful for checking that you have translated each and every message in your program. Where an exact match cannot be found, fuzzy matching is used to produce better diagnostics.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

def.po translations

ref.pot references to the sources

−D, **−−directory**=*DIRECTORY*
add DIRECTORY to list for input files search

Operation modifiers:

−m, **−−multi−domain**
apply ref.pot to each of the domains in def.po

Input file syntax:

−P, **−−properties−input**
input files are in Java .properties syntax

−−stringtable−input
input files are in NeXTstep/GNUstep .strings syntax

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Peter Miller.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgcmp** is maintained as a Texinfo manual. If the **info** and **msgcmp** programs are properly installed at your site, the command

info msgcmp

should give you access to the complete manual.

NAME

msgcomm − match two message catalogs

SYNOPSIS

msgcomm [*OPTION*] [*INPUTFILE*]...

DESCRIPTION

Find messages which are common to two or more of the specified PO files. By using the **−−more−than** option, greater commonality may be requested before messages are printed. Conversely, the **−−less−than** option may be used to specify less commonality before messages are printed (i.e. **−−less−than**=*2* will only print the unique messages). Translations, comments and extract comments will be preserved, but only from the first PO file to define them. File positions from all PO files will be cumulated.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE ...
: input files

−f, **−−files−from**=*FILE*
: get list of input files from FILE

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If input file is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Message selection:

-<, **−−less−than**=*NUMBER*
: print messages with less than this many definitions, defaults to infinite if not set

->, **−−more−than**=*NUMBER*
: print messages with more than this many definitions, defaults to 1 if not set

−u, **−−unique**
: shorthand for **−−less−than**=*2*, requests that only unique messages be printed

Input file syntax:

−P, **−−properties−input**
: input files are in Java .properties syntax

−−stringtable−input
: input files are in NeXTstep/GNUstep .strings syntax

Output details:

−e, **−−no−escape**
: do not use C escapes in output (default)

−E, **−−escape**
: use C escapes in output, no extended chars

−−force−po
: write PO file even if empty

−i, **−−indent**
: write the .po file using indented style

−−no−location
: do not write '#: filename:line' lines

−n, **−−add−location**
generate ’#: filename:line’ lines (default)

−−strict
write out strict Uniforum conforming .po file

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

−−omit−header
don’t write header with ‘msgid ""’ entry

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Peter Miller.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgcomm** is maintained as a Texinfo manual. If the **info** and **msgcomm** programs are properly installed at your site, the command

info msgcomm

should give you access to the complete manual.

NAME

msgconv − character set conversion for message catalog

SYNOPSIS

msgconv [*OPTION*] [*INPUTFILE*]

DESCRIPTION

Converts a translation catalog to a different character encoding.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE
: input PO file

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If no input file is given or if it is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Conversion target:

−t, **−−to−code**=*NAME*
: encoding for output

The default encoding is the current locale’s encoding.

Input file syntax:

−P, **−−properties−input**
: input file is in Java .properties syntax

−−stringtable−input
: input file is in NeXTstep/GNUstep .strings syntax

Output details:

−e, **−−no−escape**
: do not use C escapes in output (default)

−E, **−−escape**
: use C escapes in output, no extended chars

−−force−po
: write PO file even if empty

−i, **−−indent**
: indented output style

−−no−location
: suppress ’#: filename:line’ lines

−−add−location
: preserve ’#: filename:line’ lines (default)

−−strict
: strict Uniforum output style

−p, **−−properties−output**
: write out a Java .properties file

−−stringtable−output
: write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgconv** is maintained as a Texinfo manual. If the **info** and **msgconv** programs are properly installed at your site, the command

info msgconv

should give you access to the complete manual.

NAME

msgen − create English message catalog

SYNOPSIS

msgen [*OPTION*] *INPUTFILE*

DESCRIPTION

Creates an English translation catalog. The input file is the last created English PO file, or a PO Template file (generally created by xgettext). Untranslated entries are assigned a translation that is identical to the msgid.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE
: input PO or POT file

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If input file is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Input file syntax:

−P, **−−properties−input**
: input file is in Java .properties syntax

−−stringtable−input
: input file is in NeXTstep/GNUstep .strings syntax

Output details:

−e, **−−no−escape**
: do not use C escapes in output (default)

−E, **−−escape**
: use C escapes in output, no extended chars

−−force−po
: write PO file even if empty

−i, **−−indent**
: indented output style

−−no−location
: suppress ’#: filename:line’ lines

−−add−location
: preserve ’#: filename:line’ lines (default)

−−strict
: strict Uniforum output style

−p, **−−properties−output**
: write out a Java .properties file

−−stringtable−output
: write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
: set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgen** is maintained as a Texinfo manual. If the **info** and **msgen** programs are properly installed at your site, the command

info msgen

should give you access to the complete manual.

NAME

msgexec − process translations of message catalog

SYNOPSIS

msgexec [*OPTION*] *COMMAND* [*COMMAND-OPTION*]

DESCRIPTION

Applies a command to all translations of a translation catalog. The COMMAND can be any program that reads a translation from standard input. It is invoked once for each translation. Its output becomes msgexec's output. msgexec's return code is the maximum return code across all invocations.

A special builtin command called '0' outputs the translation, followed by a null byte. The output of "msgexec 0" is suitable as input for "xargs **−0**".

Mandatory arguments to long options are mandatory for short options too.

Input file location:

−i, **−−input**=*INPUTFILE*
input PO file

−D, **−−directory**=*DIRECTORY*
add DIRECTORY to list for input files search

If no input file is given or if it is -, standard input is read.

Input file syntax:

−P, **−−properties−input**
input file is in Java .properties syntax

−−stringtable−input
input file is in NeXTstep/GNUstep .strings syntax

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgexec** is maintained as a Texinfo manual. If the **info** and **msgexec** programs are properly installed at your site, the command

info msgexec

should give you access to the complete manual.

NAME

msgfilter − edit translations of message catalog

SYNOPSIS

msgfilter [*OPTION*] *FILTER* [*FILTER-OPTION*]

DESCRIPTION

Applies a filter to all translations of a translation catalog.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

−i, **−−input**=*INPUTFILE*
input PO file

−D, **−−directory**=*DIRECTORY*
add DIRECTORY to list for input files search

If no input file is given or if it is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
write output to specified file

The results are written to standard output if no output file is specified or if it is -.

The FILTER can be any program that reads a translation from standard input and writes a modified translation to standard output.

Useful FILTER-OPTIONs when the FILTER is 'sed':

−e, **−−expression**=*SCRIPT*
add SCRIPT to the commands to be executed

−f, **−−file**=*SCRIPTFILE*
add the contents of SCRIPTFILE to the commands to be executed

−n, **−−quiet**, **−−silent**
suppress automatic printing of pattern space

Input file syntax:

−P, **−−properties−input**
input file is in Java .properties syntax

−−stringtable−input
input file is in NeXTstep/GNUstep .strings syntax

Output details:

−−no−escape
do not use C escapes in output (default)

−E, **−−escape**
use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−−indent
indented output style

−−keep−header
keep header entry unmodified, don't filter it

−−no−location
suppress '#: filename:line' lines

−−add−location
preserve '#: filename:line' lines (default)

−−strict
strict Uniforum output style

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgfilter** is maintained as a Texinfo manual. If the **info** and **msgfilter** programs are properly installed at your site, the command

info msgfilter

should give you access to the complete manual.

NAME

msgfmt − compile message catalog to binary format

SYNOPSIS

msgfmt [*OPTION*] *filename.po* ...

DESCRIPTION

Generate binary message catalog from textual translation description.

Mandatory arguments to long options are mandatory for short options too. Similarly for optional arguments.

Input file location:

filename.po ...
input files

−D, **−−directory**=*DIRECTORY*
add DIRECTORY to list for input files search

If input file is -, standard input is read.

Operation mode:

−j, **−−java**
Java mode: generate a Java ResourceBundle class

−−java2
like **−−java**, and assume Java2 (JDK 1.2 or higher)

−−csharp
C# mode: generate a .NET .dll file

−−csharp−resources
C# resources mode: generate a .NET .resources file

−−tcl Tcl mode: generate a tcl/msgcat .msg file

−−qt Qt mode: generate a Qt .qm file

Output file location:

−o, **−−output−file**=*FILE*
write output to specified file

−−strict
enable strict Uniforum mode

If output file is -, output is written to standard output.

Output file location in Java mode:

−r, **−−resource**=*RESOURCE*
resource name

−l, **−−locale**=*LOCALE*
locale name, either language or language_COUNTRY

−d DIRECTORY
base directory of classes directory hierarchy

The class name is determined by appending the locale name to the resource name, separated with an underscore. The **−d** option is mandatory. The class is written under the specified directory.

Output file location in C# mode:

−r, **−−resource**=*RESOURCE*
resource name

−l, **−−locale**=*LOCALE*
locale name, either language or language_COUNTRY

−d DIRECTORY
base directory for locale dependent .dll files

The **−l** and **−d** options are mandatory. The .dll file is written in a subdirectory of the specified directory whose name depends on the locale.

Output file location in Tcl mode:

−l, **−−locale**=*LOCALE*
locale name, either language or language_COUNTRY

−d DIRECTORY
base directory of .msg message catalogs

The **−l** and **−d** options are mandatory. The .msg file is written in the specified directory.

Input file syntax:

−P, **−−properties−input**
input files are in Java .properties syntax

−−stringtable−input
input files are in NeXTstep/GNUstep .strings syntax

Input file interpretation:

−c, **−−check**
perform all the checks implied by **−−check−format**, **−−check−header**, **−−check−domain**

−−check−format
check language dependent format strings

−−check−header
verify presence and contents of the header entry

−−check−domain
check for conflicts between domain directives and the **−−output−file** option

−C, **−−check−compatibility**
check that GNU msgfmt behaves like X/Open msgfmt

−−check−accelerators[*=CHAR*]
check presence of keyboard accelerators for menu items

−f, **−−use−fuzzy**
use fuzzy entries in output

Output details:

−a, **−−alignment**=*NUMBER*
align strings to NUMBER bytes (default: 1)

−−no−hash
binary file will not include the hash table

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

−−statistics
print statistics about translations

−v, **−−verbose**
increase verbosity level

AUTHOR

Written by Ulrich Drepper.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgfmt** is maintained as a Texinfo manual. If the **info** and **msgfmt** programs are properly installed at your site, the command

info msgfmt

should give you access to the complete manual.

NAME

msggrep − pattern matching on message catalog

SYNOPSIS

msggrep [*OPTION*] [*INPUTFILE*]

DESCRIPTION

Extracts all messages of a translation catalog that match a given pattern or belong to some given source files.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE
: input PO file

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If no input file is given or if it is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Message selection:

[-N SOURCEFILE]... [-M DOMAINNAME]... [-K MSGID-PATTERN] [-T MSGSTR-PATTERN] [-C COMMENT-PATTERN]

A message is selected if it comes from one of the specified source files, or if it comes from one of the specified domains, or if **−K** is given and its key (msgid or msgid_plural) matches MSGID-PATTERN, or if **−T** is given and its translation (msgstr) matches MSGSTR-PATTERN, or if **−C** is given and the translator's comment matches COMMENT-PATTERN.

When more than one selection criterion is specified, the set of selected messages is the union of the selected messages of each criterion.

MSGID-PATTERN or MSGSTR-PATTERN or COMMENT-PATTERN syntax:

[-E | **−F]** [-e PATTERN | **−f** FILE]...

PATTERNs are basic regular expressions by default, or extended regular expressions if **−E** is given, or fixed strings if **−F** is given.

−N, **−−location**=*SOURCEFILE*
: select messages extracted from SOURCEFILE

−M, **−−domain**=*DOMAINNAME*
: select messages belonging to domain DOMAINNAME

−K, **−−msgid**
: start of patterns for the msgid

−T, **−−msgstr**
: start of patterns for the msgstr

−C, **−−comment**
: start of patterns for the translator's comment

−E, **−−extended−regexp**
: PATTERN is an extended regular expression

−F, **−−fixed−strings**
: PATTERN is a set of newline-separated strings

−e, **−−regexp**=*PATTERN*
use PATTERN as a regular expression

−f, **−−file**=*FILE*
obtain PATTERN from FILE

−i, **−−ignore−case**
ignore case distinctions

Input file syntax:

−P, **−−properties−input**
input file is in Java .properties syntax

−−stringtable−input
input file is in NeXTstep/GNUstep .strings syntax

Output details:

−−no−escape
do not use C escapes in output (default)

−−escape
use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−−indent
indented output style

−−no−location
suppress ’#: filename:line’ lines

−−add−location
preserve ’#: filename:line’ lines (default)

−−strict
strict Uniforum output style

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−−sort−output
generate sorted output

−−sort−by−file
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msggrep** is maintained as a Texinfo manual. If the **info** and **msggrep** programs are properly installed at your site, the command

info msggrep

should give you access to the complete manual.

NAME

msginit − initialize a message catalog

SYNOPSIS

msginit [*OPTION*]

DESCRIPTION

Creates a new PO file, initializing the meta information with values from the user's environment.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

−i, **−−input**=*INPUTFILE*
input POT file

If no input file is given, the current directory is searched for the POT file. If it is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
write output to specified PO file

If no output file is given, it depends on the **−−locale** option or the user's locale setting. If it is -, the results are written to standard output.

Input file syntax:

−P, **−−properties−input**
input file is in Java .properties syntax

−−stringtable−input
input file is in NeXTstep/GNUstep .strings syntax

Output details:

−l, **−−locale**=*LL_CC*
set target locale

−−no−translator
assume the PO file is automatically generated

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

The full documentation for **msginit** is maintained as a Texinfo manual. If the **info** and **msginit** programs are properly installed at your site, the command

info msginit

should give you access to the complete manual.

NAME

msgmerge − merge message catalog and template

SYNOPSIS

msgmerge [*OPTION*] *def.po ref.pot*

DESCRIPTION

Merges two Uniforum style .po files together. The def.po file is an existing PO file with translations which will be taken over to the newly created file as long as they still match; comments will be preserved, but extracted comments and file positions will be discarded. The ref.pot file is the last created PO file with up-to-date source references but old translations, or a PO Template file (generally created by xgettext); any translations or comments in the file will be discarded, however dot comments and file positions will be preserved. Where an exact match cannot be found, fuzzy matching is used to produce better results.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

def.po translations referring to old sources

ref.pot references to new sources

−D, **−−directory**=*DIRECTORY*
add DIRECTORY to list for input files search

−C, **−−compendium**=*FILE*
additional library of message translations, may be specified more than once

Operation mode:

−U, **−−update**
update def.po, do nothing if def.po already up to date

Output file location:

−o, **−−output−file**=*FILE*
write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Output file location in update mode: The result is written back to def.po.

−−backup=*CONTROL*
make a backup of def.po

−−suffix=*SUFFIX*
override the usual backup suffix

The version control method may be selected via the **−−backup** option or through the VERSION_CONTROL environment variable. Here are the values:

none, off
never make backups (even if **−−backup** is given)

numbered, t
make numbered backups

existing, nil
numbered if numbered backups exist, simple otherwise

simple, never
always make simple backups

The backup suffix is ‘~’, unless set with **−−suffix** or the SIMPLE_BACKUP_SUFFIX environment variable.

Operation modifiers:

−m, **−−multi−domain**
apply ref.pot to each of the domains in def.po

−N, **−−no−fuzzy−matching**
do not use fuzzy matching

Input file syntax:

−P, **−−properties−input**
input files are in Java .properties syntax

−−stringtable−input
input files are in NeXTstep/GNUstep .strings syntax

Output details:

−e, **−−no−escape**
do not use C escapes in output (default)

−E, **−−escape**
use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−i, **−−indent**
indented output style

−−no−location
suppress '#: filename:line' lines

−−add−location
preserve '#: filename:line' lines (default)

−−strict
strict Uniforum output style

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

−v, **−−verbose**
increase verbosity level

−q, **−−quiet**, **−−silent**
suppress progress indicators

AUTHOR

Written by Peter Miller.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgmerge** is maintained as a Texinfo manual. If the **info** and **msgmerge** programs are properly installed at your site, the command

info msgmerge

should give you access to the complete manual.

NAME

msgs — system messages and junk mail program

SYNOPSIS

msgs [**-fhlpqr**] [*number*] [*-number*]
msgs [**-s**]
msgs [**-c** [*days*]]

DESCRIPTION

msgs is used to read system messages. These messages are sent by mailing to the login 'msgs' and should be short pieces of information which are suitable to be read once by most users of the system.

msgs is normally invoked each time you login, by placing it in the file `.login` (or `.profile` if you use `sh`(1)). It will then prompt you with the source and subject of each new message. If there is no subject line, the first few non-blank lines of the message will be displayed. If there is more to the message, you will be told how long it is and asked whether you wish to see the rest of the message. The possible responses are:

y Type the rest of the message.

RETURN Synonym for y.

n Skip this message and go on to the next message.

- Redisplay the last message.

q Drop out of **msgs**; the next time **msgs** will pick up where it last left off.

s Append the current message to the file ''Messages'' in the current directory; 's−' will save the previously displayed message. A 's' or 's−' may be followed by a space and a file name to receive the message replacing the default ''Messages''.

m A copy of the specified message is placed in a temporary mailbox and `mail`(1) is invoked on that mailbox.

p The specified message is piped through `PAGER`, or, if `PAGER` is null or not defined, `more`(1). The commands 'm', 'p', and 's' all accept a numeric argument in place of the '−'.

msgs keeps track of the next message you will see by a number in the file `.msgsrc` in your home directory. In the directory `/var/msgs` it keeps a set of files whose names are the (sequential) numbers of the messages they represent. The file `/var/msgs/bounds` shows the low and high number of the messages in the directory so that **msgs** can quickly determine if there are no messages for you. If the contents of `bounds` is incorrect it can be fixed by removing it; **msgs** will make a new `bounds` file the next time it is run.

The **-s** option is used for setting up the posting of messages. The line

```
msgs: "| /usr/bin/msgs -s"
```

should be included in `/etc/mail/aliases` (see `newaliases`(1)) to enable posting of messages.

The **-c** option is used for performing cleanup on `/var/msgs`. An entry with the **-c** option should be placed in `/etc/crontab` to run every night. This will remove all messages over 21 days old. A different expiration may be specified on the command line to override the default.

Options when reading messages include:

-f Do not print ''No new messages.''. This is useful in a `.login` file since this is often the case here.

-q Queries whether there are messages, printing ‘‘There are new messages.’’ if there are. The command ‘‘msgs −q’’ is often used in login scripts.

-h Print the first part of messages only.

-r Disables the ability to save messages or enter the mailer. It is assumed that PAGER is set to something secure.

-l Option causes only locally originated messages to be reported.

number A message number can be given on the command line, causing **msgs** to start at the specified message rather than at the next message indicated by your .msgsrc file. Thus

```
msgs -h 1
```

prints the first part of all messages.

−number Start *number* messages back from the one indicated in the .msgsrc file, useful for reviews of recent messages.

-p Pipe long messages through PAGER, or, if PAGER is null or not defined, more(1).

Within **msgs** you can also go to any specific message by typing its number when **msgs** requests input as to what to do.

ENVIRONMENT

msgs uses the HOME and TERM environment variables for the default home directory and terminal type.

FILES

/var/msgs/* database
~/.msgsrc number of next message to be presented

SEE ALSO

mail(1), more(1), aliases(5)

HISTORY

The **msgs** command appeared in 3.0BSD.

NAME

msgunfmt − uncompile message catalog from binary format

SYNOPSIS

msgunfmt [*OPTION*] [*FILE*]...

DESCRIPTION

Convert binary message catalog to Uniforum style .po file.

Mandatory arguments to long options are mandatory for short options too.

Operation mode:

−j, **−−java**
Java mode: input is a Java ResourceBundle class

−−csharp
C# mode: input is a .NET .dll file

−−csharp−resources
C# resources mode: input is a .NET .resources file

−−tcl Tcl mode: input is a tcl/msgcat .msg file

Input file location:

FILE ...
input .mo files

If no input file is given or if it is -, standard input is read.

Input file location in Java mode:

−r, **−−resource**=*RESOURCE*
resource name

−l, **−−locale**=*LOCALE*
locale name, either language or language_COUNTRY

The class name is determined by appending the locale name to the resource name, separated with an underscore. The class is located using the CLASSPATH.

Input file location in C# mode:

−r, **−−resource**=*RESOURCE*
resource name

−l, **−−locale**=*LOCALE*
locale name, either language or language_COUNTRY

−d DIRECTORY
base directory for locale dependent .dll files

The **−l** and **−d** options are mandatory. The .dll file is located in a subdirectory of the specified directory whose name depends on the locale.

Input file location in Tcl mode:

−l, **−−locale**=*LOCALE*
locale name, either language or language_COUNTRY

−d DIRECTORY
base directory of .msg message catalogs

The **−l** and **−d** options are mandatory. The .msg file is located in the specified directory.

Output file location:

−o, **−−output−file**=*FILE*
write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Output details:

−e, **−−no−escape**
do not use C escapes in output (default)

−E, **−−escape**
use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−i, **−−indent**
write indented output style

−−strict
write strict uniforum style

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

−v, **−−verbose**
increase verbosity level

AUTHOR

Written by Ulrich Drepper.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msgunfmt** is maintained as a Texinfo manual. If the **info** and **msgunfmt** programs are properly installed at your site, the command

info msgunfmt

should give you access to the complete manual.

NAME

msguniq − unify duplicate translations in message catalog

SYNOPSIS

msguniq [*OPTION*] [*INPUTFILE*]

DESCRIPTION

Unifies duplicate translations in a translation catalog. Finds duplicate translations of the same message ID. Such duplicates are invalid input for other programs like msgfmt, msgmerge or msgcat. By default, duplicates are merged together. When using the **−−repeated** option, only duplicates are output, and all other messages are discarded. Comments and extracted comments will be cumulated, except that if **−−use−first** is specified, they will be taken from the first translation. File positions will be cumulated. When using the **−−unique** option, duplicates are discarded.

Mandatory arguments to long options are mandatory for short options too.

Input file location:

INPUTFILE
: input PO file

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If no input file is given or if it is -, standard input is read.

Output file location:

−o, **−−output−file**=*FILE*
: write output to specified file

The results are written to standard output if no output file is specified or if it is -.

Message selection:

−d, **−−repeated**
: print only duplicates

−u, **−−unique**
: print only unique messages, discard duplicates

Input file syntax:

−P, **−−properties−input**
: input file is in Java .properties syntax

−−stringtable−input
: input file is in NeXTstep/GNUstep .strings syntax

Output details:

−t, **−−to−code**=*NAME*
: encoding for output

−−use−first
: use first available translation for each message, don’t merge several translations

−e, **−−no−escape**
: do not use C escapes in output (default)

−E, **−−escape**
: use C escapes in output, no extended chars

−−force−po
: write PO file even if empty

−i, **−−indent**
: write the .po file using indented style

−−no−location
do not write '#: filename:line' lines

−n, **−−add−location**
generate '#: filename:line' lines (default)

−−strict
write out strict Uniforum conforming .po file

−p, **−−properties−output**
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Bruno Haible.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **msguniq** is maintained as a Texinfo manual. If the **info** and **msguniq** programs are properly installed at your site, the command

info msguniq

should give you access to the complete manual.

NAME

mt — magnetic tape manipulation

SYNOPSIS

mt [**-f** *tapename*] *command* [*count*]

DESCRIPTION

The **mt** program is used to give commands to a magnetic tape drive. By default **mt** performs the requested operation once. Operations may be performed multiple times by specifying *count*.

Note that *tapename* must reference a raw (not block) tape device. If *tapename* is of the form "host:tapename", or "user@host:tapename", **mt** writes to the named tape device on the remote host using rmt(8).

The rmt(8) process on the remote host is typically initiated via rsh(1), although an alternate method such as ssh(1) can be specified via the RCMD_CMD environment variable.

The available commands are listed below. Only as many characters as are required to uniquely identify a command need be specified.

asf Move forward *count* files from the beginning of the tape. This is accomplished by a rewind followed by fsf *count*.

eof, **weof** Write *count* end-of-file marks at the current position on the tape.

fsf Forward space *count* files.

fsr Forward space *count* records.

bsf Back space *count* files.

bsr Back space *count* records.

rewind Rewind the tape. (The *count* is ignored.)

offline, **rewoffl**
Rewind the tape and place the tape unit off-line. Where supported, this ejects the tape. (The *count* is ignored.)

status Print status information about the tape unit. (The *count* is ignored.)

retension Retensions the tape. Not all tape drives support this feature. (The *count* is ignored.)

erase Erases the tape Not all tape drives support this feature. (The *count* is ignored.)

eew Enable or disable early warning EOM behaviour. Set *count* to nonzero to enable, zero to disable.

eom Forward space to the end of recorded media. (The *count* is ignored.)

blocksize, **setblk**
Set the tape blocksize to *count* bytes. A *count* of zero sets variable blocksize.

density, **setdensity**
Set the tape density code to *count* as specified in the SCSI-3 specification. See the **DENSITY CODES** section for a list of codes for commonly used media types.

rdspos Read the logical block position of the tape. Not all tape drives support this feature. (The *count* is ignored.)

rdhpos Read the hardware block position of the tape. Not all tape drives support this feature. (The *count* is ignored.)

setspos Set the logical block position of the tape to *count*. Not all tape drives support this feature.

sethpos Set the hardware block position of the tape to *count*. Not all tape drives support this feature.

compress If *count* is zero, disable compression. Otherwise enable compression. Not all tape drives support this feature.

If a tape name is not specified, and the environment variable TAPE is not set, then **mt** uses the device /dev/nrst0.

EXIT STATUS

mt returns a 0 exit status when the operation(s) were successful, 1 if the command was unrecognized, and 2 if an operation failed.

DENSITY CODES

The SCSI-3 specification defines a number of density codes for various tape media, some of which are listed here. Note that many tape drive vendors also define model-specific codes.

Code	*Format*
0	Device default
1	1/2" 800 bpi
2	1/2" 1600 bpi
3	1/2" 6250 bpi
4	QIC-11
5	QIC-24
15	QIC-120
16	QIC-150
17	QIC-320/525
18	QIC-1320/1350
19	DDS
28	QIC-385M
29	QIC-410M
30	QIC-1000C
31	QIC-2100C
32	QIC-6GB
33	QIC-20GB
34	QIC-2GB
35	QIC-875M
36	DDS-2
37	DDS-3
38	DDS-4

ENVIRONMENT

If the following environment variables exist, they are used by **mt**.

TAPE **mt** uses device filename given in the TAPE environment variable if the *tapename* argument is not given.

RCMD_CMD **mt** will use RCMD_CMD rather than /usr/bin/rsh to invoke rmt(8) on a remote machine. The full path name must be specified.

FILES

`/dev/rst*` Raw SCSI tape device
`/dev/rmt*` Raw magnetic tape device

SEE ALSO

`dd`(1), `ioctl`(2), `mtio`(4), `st`(4), `environ`(7)

HISTORY

The **`mt`** utility appeared in 4.3BSD.

NAME

mv — move files

SYNOPSIS

mv [**-fiv**] *source target*
mv [**-fiv**] *source ... directory*

DESCRIPTION

In its first form, the **mv** utility renames the file named by the *source* operand to the destination path named by the *target* operand. This form is assumed when the last operand does not name an already existing directory.

In its second form, **mv** moves each file named by a *source* operand to a destination file in the existing directory named by the *directory* operand. The destination path for each operand is the pathname produced by the concatenation of the last operand, a slash, and the final pathname component of the named file.

The following options are available:

-f Do not prompt for confirmation before overwriting the destination path.

-i Causes **mv** to write a prompt to standard error before moving a file that would overwrite an existing file. If the response from the standard input begins with the character ‘‘y’’, the move is attempted.

-v Cause **mv** to be verbose, showing files as they are processed.

The last of any **-f** or **-i** options is the one which affects **mv**’s behavior.

It is an error for any of the *source* operands to specify a nonexistent file or directory.

It is an error for the *source* operand to specify a directory if the *target* exists and is not a directory.

If the destination path does not have a mode which permits writing, **mv** prompts the user for confirmation as specified for the **-i** option.

Should the rename(2) call fail because *source* and *target* are on different file systems, **mv** will remove the destination file, copy the source file to the destination, and then remove the source. The effect is roughly equivalent to:

```
rm -f destination_path && \
cp -PRp source_file destination_path && \
rm -rf source_file
```

EXIT STATUS

The **mv** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

cp(1), rename(2), symlink(7)

STANDARDS

The **mv** utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

The **-v** option is an extension to IEEE Std 1003.2 (“POSIX.2”).

NAME

nbsvtool — create and verify detached signatures of files

SYNOPSIS

nbsvtool [**-a** *anchor-certificates*] [**-c** *certificate-chain*] [**-f** *certificate-chain*] [**-k** *keyfile*] [**-u** *required-keyusage*] *command* [*file ...*]

DESCRIPTION

nbsvtool is used to create and verify detached signatures of files. **nbsvtool** has two modes, signature creation and verification.

The signature mode requires the options **-k** and **-c**.

The command `verify-code` is an alias for `-u code verify`.

Supported options:

-a *trust anchor*	Trust anchor that will be used to verify the certificate signing the request.
-c *certificate chain file*	Additional certificates that will be added to the signature when creating one. For verification it is used to fill missing links in the trust chain.
-f *certificate file*	The certicate file to use when creating signatures. The certicate needs to match the key given by **-k**. This is enforced by the program.
-k *private key*	The private key file to use when creating signatures.
-u *key usage*	*key usage* is one of: “ssl-server”, “ssl-client”, “code”, or “smime”. This verifies the extended key-usage attribute in the signer certificate.
-v	Print verbose information of the signer.

EXIT STATUS

The **nbsvtool** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

```
nbsvtool -k key -c cert-chain sign hello hello.sp7
nbsvtool verify-code hello hello.sp7
nbsvtool -u code verify file file.sp7
nbsvtool -a anchor-file verify file.sp7
```

SEE ALSO

`openssl_smime`(1)

NAME

netgroup — list netgroup members

SYNOPSIS

netgroup [**-dhu**] *netgroup*

DESCRIPTION

netgroup lists the members of specified netgroups. The type of members is selected using the following options:

-d list domains

-h list hosts (the default)

-u list users

SEE ALSO

innetgr(1), netgroup(5)

HISTORY

A **netgroup** utility appeared in NetBSD 1.4 and was inspired from the **netgroup** utility that was written by Ken Lalonde (University of Toronto).

NAME

netstat — show network status

SYNOPSIS

netstat [**-Aan**] [**-f** *address_family*] [**-M** *core*] [**-N** *system*]
netstat [**-bdgiLmnqrsSv**] [**-f** *address_family*] [**-M** *core*] [**-N** *system*]
netstat [**-dn**] [**-I** *interface*] [**-M** *core*] [**-N** *system*] [**-w** *wait*]
netstat [**-p** *protocol*] [**-M** *core*] [**-N** *system*]
netstat [**-p** *protocol*] [**-M** *core*] [**-N** *system*] **-P** *pcbaddr*
netstat [**-p** *protocol*] [**-i**] [**-I** *Interface*]
netstat [**-s**] [**-f** *address_family*] [**-i**] [**-I** *Interface*]
netstat [**-s**] [**-I** *Interface*] **-B**

DESCRIPTION

The **netstat** command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented. The first form of the command displays a list of active sockets for each protocol. The second form presents the contents of one of the other network data structures according to the option selected. Using the third form, with a *wait* interval specified, **netstat** will continuously display the information regarding packet traffic on the configured network interfaces. The fourth form displays statistics about the named protocol. The fifth and sixth forms display per interface statistics for the specified protocol or address family.

The options have the following meaning:

-A With the default display, show the address of any protocol control blocks associated with sockets; used for debugging.

-a With the default display, show the state of all sockets; normally sockets used by server processes are not shown.

-B With the default display, show the current bpf(4) peers. To show only the peers listening to a specific interface, use the **-I** option. If the **-s** option is present, show the current bpf(4) statistics.

-b With the interface display (option **-i**), show bytes in and out, instead of packets in and out.

-d With either interface display (option **-i** or an interval, as described below), show the number of dropped packets.

-f *address_family*
Limit statistics or address control block reports to those of the specified *address_family*. The following address families are recognized: *inet*, for AF_INET; *inet6*, for AF_INET6; *arp*, for AF_ARP; *ns*, for AF_NS; *iso*, for AF_ISO; *atalk*, for AF_APPLETALK; and *local* or *unix*, for AF_LOCAL.

-g Show information related to multicast (group address) routing. By default, show the IP Multicast virtual-interface and routing tables. If the **-s** option is also present, show multicast routing statistics.

-I *interface*
Show information about the specified interface; used with a *wait* interval as described below. If the **-f** *address_family* option (with the **-s** option) or the **-p** *protocol* option is present, show per-interface statistics on the *interface* for the specified *address_family* or *protocol*, respectively.

-i Show the state of interfaces which have been auto-configured (interfaces statically configured into a system, but not located at boot time are not shown). If the **-a** options is also present, multicast addresses currently in use are shown for each Ethernet interface and for each IP interface address. Multicast addresses are shown on separate lines following the interface address with which they are

associated. If the **-f** *address_family* option (with the **-s** option) or the **-p** *protocol* option is present, show per-interface statistics on all interfaces for the specified *address_family* or *protocol*, respectively.

-L Don't show link-level routes (e.g., IPv4 ARP or IPv6 neighbour cache).

-M Extract values associated with the name list from the specified core instead of the default /dev/kmem.

-m Show statistics recorded by the memory management routines (the network manages a private pool of memory buffers).

-N Extract the name list from the specified system instead of the default /netbsd.

-n Show network addresses and ports as numbers (normally **netstat** interprets addresses and ports and attempts to display them symbolically). This option may be used with any of the display formats.

-S Show network addresses as numbers (as with **-n**, but show ports symbolically).

-P *pcbaddr*
Dump the contents of the protocol control block (PCB) located at kernel virtual address *pcbaddr*. This address may be obtained using the **-A** flag. The default protocol is TCP, but may be overridden using the **-p** flag.

-p *protocol*
Show statistics about *protocol*, which is either a well-known name for a protocol or an alias for it. Some protocol names and aliases are listed in the file /etc/protocols. A null response typically means that there are no interesting numbers to report. The program will complain if *protocol* is unknown or if there is no statistics routine for it.

-q Show software interrupt queue setting/statistics for all protocols.

-s Show per-protocol statistics. If this option is repeated, counters with a value of zero are suppressed.

-r Show the routing tables. When **-s** is also present, show routing statistics instead.

-v Show extra (verbose) detail for the routing tables (**-r**), or avoid truncation of long addresses.

-w *wait*
Show network interface statistics at intervals of *wait* seconds.

The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form ''host.port'' or ''network.port'' if a socket's address specifies a network but no specific host address. When known the host and network addresses are displayed symbolically according to the data bases /etc/hosts and /etc/networks, respectively. If a symbolic name for an address is unknown, or if the **-n** option is specified, the address is printed numerically, according to the address family. For more information regarding the Internet ''dot format,'' refer to inet(3)). Unspecified, or ''wildcard'', addresses and ports appear as ''*''. You can use the fstat(1) to find out which process or processes hold references to a socket.

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (''mtu'') are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows a collection of information about the route stored as binary choices. The individual flags are discussed in more detail in the route(8) and route(4) manual pages. The mapping between letters and flags is:

1 RTF_PROTO1 Protocol specific routing flag #1

2	RTF_PROTO2	Protocol specific routing flag #2
B	RTF_BLACKHOLE	Just discard pkts (during updates)
C	RTF_CLONING	Generate new routes on use
c	RTF_CLONED	Cloned routes (generated from RTF_CLONING)
D	RTF_DYNAMIC	Created dynamically (by redirect)
G	RTF_GATEWAY	Destination requires forwarding by intermediary
H	RTF_HOST	Host entry (net otherwise)
L	RTF_LLINFO	Valid protocol to link address translation.
M	RTF_MODIFIED	Modified dynamically (by redirect)
R	RTF_REJECT	Host or net unreachable
S	RTF_STATIC	Manually added
U	RTF_UP	Route usable
X	RTF_XRESOLVE	External daemon translates proto to link address

Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface. The refcnt field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The mtu entry shows the mtu associated with that route. This mtu value is used as the basis for the TCP maximum segment size. The 'L' flag appended to the mtu value indicates that the value is locked, and that path mtu discovery is turned off for that route. A '-' indicates that the mtu for this route has not been set, and a default TCP maximum segment size will be used. The interface entry indicates the network interface used for the route.

When **netstat** is invoked with the **-w** option and a *wait* interval argument, it displays a running count of statistics related to network interfaces. An obsolescent version of this option used a numeric parameter with no option, and is currently supported for backward compatibility. This display consists of a column for the primary interface (the first interface found during autoconfiguration) and a column summarizing information for all interfaces. The primary interface may be replaced with another interface with the **-I** option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

The first character of the flags column in the **-B** option shows the status of the bpf(4) descriptor which has three different values: Idle ('I'), Waiting ('W') and Timed Out ('T'). The second character indicates wheter the promisc flag is set. The third character indicates the status of the immediate mode. The fourth character indicates whether the peer will have the ability to see the packets sent. And the fifth character shows the header complete flag status.

SEE ALSO

fstat(1), nfsstat(1), ps(1), sockstat(1), vmstat(1), inet(3), bpf(4), hosts(5), networks(5), protocols(5), services(5), iostat(8), trpt(8),

HISTORY

The **netstat** command appeared in 4.2BSD. IPv6 support was added by WIDE/KAME project.

BUGS

The notion of errors is ill-defined.

NAME

newgrp — change to a new primary group

SYNOPSIS

newgrp [**-l**] [*group*]

DESCRIPTION

The **newgrp** command changes a user to a new primary group (real and effective group ID) by starting a new shell. The user remains logged in and the current directory and file creation mask remain unchanged. The user is always given a new shell even if the primary group change fails.

The **newgrp** command accepts the following options:

-l The environment is changed to what would be expected if the user actually logged in again. This simulates a full login.

The *group* is a group name or non-negative numeric group ID from the group database. The real and effective group IDs are set to *group* or the group ID associated with the group name.

If *group* is not specified, **newgrp** restores the user's real and effective group IDs to the user's primary group specified in the password database. The user's supplementary group IDs are restored to the set specified for the user in the group database.

If the user is not a member of the specified group, and the group requires a password, the user will be prompted for the group password.

EXIT STATUS

If a new shell is started the exit status is the exit status of the shell. Otherwise the exit status will be >0.

FILES

`/etc/group`	The group database
`/etc/master.passwd`	The user database
`/etc/passwd`	A Version 7 format password file

SEE ALSO

`csh`(1), `groups`(1), `login`(1), `sh`(1), `su`(1), `umask`(2), `group`(5), `passwd`(5), `environ`(7)

STANDARDS

The **newgrp** command conforms to IEEE Std 1003.1-2001 ("POSIX.1").

HISTORY

A **newgrp** command appeared in Version 6 AT&T UNIX. A **newgrp** command appeared in NetBSD 5.0.

BUGS

There is no convenient way to enter a password into `/etc/group`. The use of group passwords is strongly discouraged since they are inherently insecure. It is not possible to stop users from obtaining the encrypted password from the group database.

NAME

nfsstat — display NFS statistics

SYNOPSIS

nfsstat [**-cs**] [**-M** *core*] [**-N** *system*] [**-w** *wait*]

DESCRIPTION

nfsstat displays statistics kept about NFS client and server activity.

The options are as follows:

-c Only display values for NFS client side.

-s Only display values for NFS server side.

-M Extract values associated with the name list from the specified core instead of the default `/dev/kmem`.

-N Extract the name list from the specified system instead of the default `/netbsd`.

-w Display a shorter summary of NFS activity for both the client and server at *wait* second intervals.

FILES

`/netbsd` default kernel namelist
`/dev/kmem` default memory file

SEE ALSO

`fstat`(1), `netstat`(1), `ps`(1), `systat`(1), `vmstat`(1), `iostat`(8), `pstat`(8)

HISTORY

The **nfsstat** command appears in 4.4BSD.

NAME

nice — execute a utility with an altered scheduling priority

SYNOPSIS

nice [**-n** *increment*] *utility* [*argument ...*]

DESCRIPTION

nice runs *utility* at an altered scheduling priority. If an *increment* is given, it is used; otherwise an increment of 10 is assumed. The super-user can run utilities with priorities higher than normal by using a negative *increment*. The priority can be adjusted over a range of -20 (the highest) to 20 (the lowest). A priority of 19 or 20 will prevent a process from taking any cycles from others at nice 0 or better.

Available options:

-n *increment*
: A positive or negative decimal integer used to modify the system scheduling priority of *utility*.

EXIT STATUS

The **nice** utility exits with one of the following values:

1-125 An error occurred in the **nice** utility.

126 The *utility* was found but could not be invoked.

127 The *utility* could not be found.

Otherwise, the exit status of **nice** will be that of *utility*.

COMPATIBILITY

The historic **-***increment* option has been deprecated but is still supported in this implementation.

SEE ALSO

csh(1), getpriority(2), setpriority(2), renice(8)

STANDARDS

The **nice** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

HISTORY

A **nice** utility appeared in Version 6 AT&T UNIX.

BUGS

nice is built into csh(1) with a slightly different syntax than described here. The form `nice +10` nices to positive nice, and `nice -10` can be used by the super-user to give a process more of the processor.

NAME

nl — line numbering filter

SYNOPSIS

nl [**-p**] [**-b** *type*] [**-d** *delim*] [**-f** *type*] [**-h** *type*] [**-i** *incr*] [**-l** *num*] [**-n** *format*] [**-s** *sep*] [**-v** *startnum*] [**-w** *width*] [*file*]

DESCRIPTION

The **nl** utility reads lines from the named *file* or the standard input if the *file* argument is omitted, applies a configurable line numbering filter operation and writes the result to the standard output.

The **nl** utility treats the text it reads in terms of logical pages. Unless specified otherwise, line numbering is reset at the start of each logical page. A logical page consists of a header, a body and a footer section; empty sections are valid. Different line numbering options are independently available for header, body and footer sections.

The starts of logical page sections are signaled by input lines containing nothing but one of the following sequences of delimiter characters:

Line	*Start of*
\:\:\:	header
\:\:	body
\:	footer

If the input does not contain any logical page section signaling directives, the text being read is assumed to consist of a single logical page body.

The following options are available:

-b *type*

Specify the logical page body lines to be numbered. Recognized *type* arguments are:

a	Number all lines.
t	Number only non-empty lines.
n	No line numbering.
p*expr*	Number only those lines that contain the basic regular expression specified by *expr*.

The default *type* for logical page body lines is t.

-d *delim*

Specify the delimiter characters used to indicate the start of a logical page section in the input file. At most two characters may be specified; if only one character is specified, the first character is replaced and the second character remains unchanged. The default *delim* characters are ‘‘\:’’.

-f *type*

Specify the same as **-b** *type* except for logical page footer lines. The default *type* for logical page footer lines is n.

-h *type*

Specify the same as **-b** *type* except for logical page header lines. The default *type* for logical page header lines is n.

-i *incr*

Specify the increment value used to number logical page lines. The default *incr* value is 1.

-l *num*
If numbering of all lines is specified for the current logical section using the corresponding **-b** a, **-f** a or **-h** a option, specify the number of adjacent blank lines to be considered as one. For example, **-l** 2 results in only the second adjacent blank line being numbered. The default *num* value is 1.

-n *format*
Specify the line numbering output format. Recognized *format* arguments are:
ln Left justified.
rn Right justified, leading zeros suppressed.
rz Right justified, leading zeros kept.

The default *format* is rn.

-p Specify that line numbering should not be restarted at logical page delimiters.

-s *sep*
Specify the characters used in separating the line number and the corresponding text line. The default *sep* setting is a single tab character.

-v *startnum*
Specify the initial value used to number logical page lines; see also the description of the **-p** option. The default *startnum* value is 1.

-w *width*
Specify the number of characters to be occupied by the line number; in case the *width* is insufficient to hold the line number, it will be truncated to its *width* least significant digits. The default *width* is 6.

EXIT STATUS

The **nl** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

pr(1)

STANDARDS

The **nl** utility conforms to X/Open Portability Guide Issue 4, Version 2 ("XPG4.2") with the exception of not supporting the intermingling of the *file* operand with the options, which the standard considers an obsolescent feature to be removed from a further issue.

HISTORY

The **nl** utility first appeared in AT&T System V.2 UNIX.

NAME

nlmconv − converts object code into an NLM.

SYNOPSIS

nlmconv [**−I** *bfdname*|**−−input−target=***bfdname*]
[**−O** *bfdname*|**−−output−target=***bfdname*]
[**−T** *headerfile*|**−−header−file=***headerfile*]
[**−d**|**−−debug**] [**−l** *linker*|**−−linker=***linker*]
[**−h**|**−−help**] [**−V**|**−−version**]
infile outfile

DESCRIPTION

nlmconv converts the relocatable **i386** object file *infile* into the NetWare Loadable Module *outfile*, optionally reading *headerfile* for NLM header information. For instructions on writing the NLM command file language used in header files, see the **linkers** section, **NLMLINK** in particular, of the *NLM Development and Tools Overview*, which is part of the NLM Software Developer's Kit (''NLM SDK''), available from Novell, Inc. **nlmconv** uses the GNU Binary File Descriptor library to read *infile*;

nlmconv can perform a link step. In other words, you can list more than one object file for input if you list them in the definitions file (rather than simply specifying one input file on the command line). In this case, **nlmconv** calls the linker for you.

OPTIONS

−I *bfdname*
−−input−target=*bfdname*
Object format of the input file. **nlmconv** can usually determine the format of a given file (so no default is necessary).

−O *bfdname*
−−output−target=*bfdname*
Object format of the output file. **nlmconv** infers the output format based on the input format, e.g. for a **i386** input file the output format is **nlm32−i386**.

−T *headerfile*
−−header−file=*headerfile*
Reads *headerfile* for NLM header information. For instructions on writing the NLM command file language used in header files, see see the **linkers** section, of the *NLM Development and Tools Overview*, which is part of the NLM Software Developer's Kit, available from Novell, Inc.

−d
−−debug
Displays (on standard error) the linker command line used by **nlmconv**.

−l *linker*
−−linker=*linker*
Use *linker* for any linking. *linker* can be an absolute or a relative pathname.

−h
−−help
Prints a usage summary.

−V
−−version
Prints the version number for **nlmconv**.

SEE ALSO

the Info entries for *binutils*.

COPYRIGHT

NAME

nm − list symbols from object files

SYNOPSIS

nm [**−a**|**−−debug−syms**] [**−g**|**−−extern−only**]
[**−B**] [**−C**|**−−demangle**[*=style*]] [**−D**|**−−dynamic**]
[**−S**|**−−print−size**] [**−s**|**−−print−armap**]
[**−A**|**−o**|**−−print−file−name**][**−−special−syms**]
[**−n**|**−v**|**−−numeric−sort**] [**−p**|**−−no−sort**]
[**−r**|**−−reverse−sort**] [**−−size−sort**] [**−u**|**−−undefined−only**]
[**−t** *radix*|**−−radix=***radix*] [**−P**|**−−portability**]
[**−−target=***bfdname*] [**−f***format*|**−−format=***format*]
[**−−defined−only**] [**−l**|**−−line−numbers**] [**−−no−demangle**]
[**−V**|**−−version**] [**−X 32_64**] [**−−help**] [*objfile*...]

DESCRIPTION

GNU **nm** lists the symbols from object files *objfile*.... If no object files are listed as arguments, **nm** assumes the file *a.out*.

For each symbol, **nm** shows:

- The symbol value, in the radix selected by options (see below), or hexadecimal by default.
- The symbol type. At least the following types are used; others are, as well, depending on the object file format. If lowercase, the symbol is local; if uppercase, the symbol is global (external).

  `A` The symbol's value is absolute, and will not be changed by further linking.

  `B` The symbol is in the uninitialized data section (known as BSS).

  `C` The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the symbol is defined anywhere, the common symbols are treated as undefined references.

  `D` The symbol is in the initialized data section.

  `G` The symbol is in an initialized data section for small objects. Some object file formats permit more efficient access to small data objects, such as a global int variable as opposed to a large global array.

  `I` The symbol is an indirect reference to another symbol. This is a GNU extension to the a.out object file format which is rarely used.

  `N` The symbol is a debugging symbol.

  `R` The symbol is in a read only data section.

  `S` The symbol is in an uninitialized data section for small objects.

  `T` The symbol is in the text (code) section.

  `U` The symbol is undefined.

  `V` The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error.

  `W` The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the symbol is determined in a system-specific manner without error. On some systems, uppercase indicates that a default value has been specified.

  `−` The symbol is a stabs symbol in an a.out object file. In this case, the next values printed are the stabs other field, the stabs desc field, and the stab type. Stabs symbols are used to hold debugging information.

`?` The symbol type is unknown, or object file format specific.

- The symbol name.

OPTIONS

The long and short forms of options, shown here as alternatives, are equivalent.

−A
−o
−−print−file−name
Precede each symbol by the name of the input file (or archive member) in which it was found, rather than identifying the input file once only, before all of its symbols.

−a
−−debug−syms
Display all symbols, even debugger-only symbols; normally these are not listed.

−B The same as **−−format=bsd** (for compatibility with the MIPS **nm**).

−C
−−demangle[=*style*]
Decode (*demangle*) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

−−no−demangle
Do not demangle low-level symbol names. This is the default.

−D
−−dynamic
Display the dynamic symbols rather than the normal symbols. This is only meaningful for dynamic objects, such as certain types of shared libraries.

−f *format*
−−format=*format*
Use the output format *format*, which can be `bsd`, `sysv`, or `posix`. The default is `bsd`. Only the first character of *format* is significant; it can be either upper or lower case.

−g
−−extern−only
Display only external symbols.

−l
−−line−numbers
For each symbol, use debugging information to try to find a filename and line number. For a defined symbol, look for the line number of the address of the symbol. For an undefined symbol, look for the line number of a relocation entry which refers to the symbol. If line number information can be found, print it after the other symbol information.

−n
−v
−−numeric−sort
Sort symbols numerically by their addresses, rather than alphabetically by their names.

−p
−−no−sort
Do not bother to sort the symbols in any order; print them in the order encountered.

−P
−−portability
Use the POSIX.2 standard output format instead of the default format. Equivalent to **−f posix**.

−S
−−print−size
Print size, not the value, of defined symbols for the `bsd` output format.

−s
−−print−armap
When listing symbols from archive members, include the index: a mapping (stored in the archive by **ar** or **ranlib**) of which modules contain definitions for which names.

−r
−−reverse−sort
Reverse the order of the sort (whether numeric or alphabetic); let the last come first.

−−size−sort
Sort symbols by size. The size is computed as the difference between the value of the symbol and the value of the symbol with the next higher value. If the `bsd` output format is used the size of the symbol is printed, rather than the value, and **−S** must be used in order both size and value to be printed.

−−special−syms
Display symbols which have a target-specific special meaning. These symbols are usually used by the target for some special processing and are not normally helpful when included included in the normal symbol lists. For example for ARM targets this option would skip the mapping symbols used to mark transistions between ARM code, THUMB code and data.

−t *radix*
−−radix=*radix*
Use *radix* as the radix for printing the symbol values. It must be **d** for decimal, **o** for octal, or **x** for hexadecimal.

−−target=*bfdname*
Specify an object code format other than your system's default format.

−u
−−undefined−only
Display only undefined symbols (those external to each object file).

−−defined−only
Display only defined symbols for each object file.

−V
−−version
Show the version number of **nm** and exit.

−X This option is ignored for compatibility with the AIX version of **nm**. It takes one parameter which must be the string **32_64**. The default mode of AIX **nm** corresponds to **−X 32**, which is not supported by GNU **nm**.

−−help
Show a summary of the options to **nm** and exit.

SEE ALSO

ar (1), *objdump* (1), *ranlib* (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

nohup — invoke a command immune to hangups

SYNOPSIS

nohup *utility* [*arg ...*]

DESCRIPTION

The **nohup** command allows the specified utility to be protected from termination if the user should become logged out (for example, due to a modem line or TCP/IP connection being dropped). To do this, **nohup** sets the `SIGHUP` `signal`(3) ("terminal line hangup") to be ignored, then executes *utility* along with any arguments.

If the standard output is a terminal, the standard output is appended to the file `nohup.out` in the current directory. If standard error is a terminal, it is directed to the same place as the standard output. If the output file `nohup.out` cannot be created in the current directory, **nohup** attempts to create the file in the user's home directory. If the file `nohup.out` cannot be created, either in the current directory or the user's home directory, **nohup** will exit without invoking *utility*, with an exit value as described below.

EXIT STATUS

The **nohup** utility exits with one of the following values:

126 The *utility* was found but could not be invoked.

127 The *utility* could not be found or an error occurred in **nohup**.

Otherwise, the exit status of **nohup** will be that of *utility*.

ENVIRONMENT

The following variable is used by **nohup**.

`HOME` User's home directory.

SEE ALSO

`signal`(3)

STANDARDS

The **nohup** command is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

NAME

nslookup − query Internet name servers interactively

SYNOPSIS

nslookup [**−option**] [name | −] [server]

DESCRIPTION

Nslookup is a program to query Internet domain name servers. **Nslookup** has two modes: interactive and non−interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non−interactive mode is used to print just the name and requested information for a host or domain.

ARGUMENTS

Interactive mode is entered in the following cases:

1. when no arguments are given (the default name server will be used)
2. when the first argument is a hyphen (−) and the second argument is the host name or Internet address of a name server.

Non−interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Options can also be specified on the command line if they precede the arguments and are prefixed with a hyphen. For example, to change the default query type to host information, and the initial timeout to 10 seconds, type:

INTERACTIVE COMMANDS

host [server]

Look up information for host using the current default server or using server, if specified. If host is an Internet address and the query type is A or PTR, the name of the host is returned. If host is a name and does not have a trailing period, the search list is used to qualify the name.

To look up a host not in the current domain, append a period to the name.

server *domain*

lserver *domain*

Change the default server to *domain*; **lserver** uses the initial server to look up information about *domain*, while **server** uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

root

not implemented

finger

not implemented

ls

not implemented

view

not implemented

help

not implemented

?

not implemented

exit

Exits the program.

set *keyword[=value]*

This command is used to change state information that affects the lookups. Valid keywords are:

all
Prints the current values of the frequently used options to **set**. Information about the current default server and host is also printed.

class=*value*
Change the query class to one of:

IN
the Internet class

CH
the Chaos class

HS
the Hesiod class

ANY
wildcard

The class specifies the protocol group of the information.

(Default = IN; abbreviation = cl)

*[no]***debug**
Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer.

(Default = nodebug; abbreviation = [no]deb)

*[no]***d2**
Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer.

(Default = nod2)

domain=*name*
Sets the search list to *name*.

*[no]***search**
If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received.

(Default = search)

port=*value*
Change the default TCP/UDP name server port to *value*.

(Default = 53; abbreviation = po)

querytype=*value*

type=*value*
Change the type of the information query.

(Default = A; abbreviations = q, ty)

*[no]***recurse**
Tell the name server to query other servers if it does not have the information.

(Default = recurse; abbreviation = [no]rec)

retry=*number*

Set the number of retries to number.

timeout=*number*
Change the initial timeout interval for waiting for a reply to number seconds.

*[no]***vc**
Always use a virtual circuit when sending requests to the server.

(Default = novc)

*[no]***fail**
Try the next nameserver if a nameserver responds with SERVFAIL or a referral (nofail) or terminate query (fail) on such a response.

(Default = nofail)

FILES

/etc/resolv.conf

SEE ALSO

dig(1), **host**(1), **named**(8).

AUTHOR

Andrew Cherenson

COPYRIGHT

NAME

ntp-keygen − Create a NTP host key

SYNOPSIS

ntp-keygen [−*flag* [*value*]]... [−*-opt-name* [[=|]*value*]]...

All arguments must be options.

DESCRIPTION

This manual page documents, briefly, the **ntp-keygen** command. If there is no new host key, look for an existing one. If one is not found, create it.

OPTIONS

−c *scheme*, **−-certificate**=*scheme*
certificate scheme.

Just some descriptive text.

−d, **−-debug-level**
Increase output debug message level. This option may appear an unlimited number of times.

Increase the debugging message output level.

−D *string*, **−-set-debug-level**=*string*
Set the output debug message level. This option may appear an unlimited number of times.

Set the output debugging level. Can be supplied multiple times, but each overrides the previous value(s).

−e, **−-id-key**
Write identity keys.

Just some descriptive text.

−G, **−-gq-params**
Generate GQ parameters and keys.

Just some descriptive text.

−g, **−-gq-keys**
update GQ keys.

Just some descriptive text.

−H, **−-host-key**
generate RSA host key.

Just some descriptive text.

−I, **−-iffkey**
generate IFF parameters.

Just some descriptive text.

−i, **−-issuer-name**
set issuer name.

Just some descriptive text.

−M, **−-md5key**
generate MD5 keys.

Just some descriptive text.

−m *modulus*, **−-modulus**=*modulus*
modulus. This option takes an integer number as its argument. The value of *modulus* is constrained to being:
in the range 256 through 2048

Just some descriptive text.

−P, **−-pvt-cert**
generate PC private certificate.

Just some descriptive text.

−p *passwd*, **−-pvt-passwd**=*passwd*
output private password.

Just some descriptive text.

−q *passwd*, **−-get-pvt-passwd**=*passwd*
input private password.

Just some descriptive text.

−S *sign*, **−-sign-key**=*sign*
generate sign key (RSA or DSA).

Just some descriptive text.

−s *host*, **−-subject-name**=*host*
set subject name.

Just some descriptive text.

−T, **−-trusted-cert**
trusted certificate (TC scheme).

Just some descriptive text.

−V *num*, **−-mv-params**=*num*
generate <num> MV parameters. This option takes an integer number as its argument.

Just some descriptive text.

−v *num*, **−-mv-keys**=*num*
update <num> MV keys. This option takes an integer number as its argument.

Just some descriptive text.

−?, **−-help**
Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−> [*rcfile*], **−-save-opts**[=*rcfile*]
Save the option state to *rcfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below.

−< *rcfile*, **−-load-opts**=*rcfile*, **−-no-load-opts**
Load options from *rcfile*. The *no-load-opts* form will disable the loading of earlier RC/INI files. *--no-load-opts* is handled early, out of order.

−v [{*v|c|n*}], **−-version**[*={v|c|n}*]

Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:

NTP_KEYGEN_<option-name> or **NTP_KEYGEN**

The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those directories.

AUTHOR

ntp.org

Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

This manual page was *AutoGen*-erated from the **ntp-keygen** option definitions.

NAME

ntpd − NTP daemon program

SYNOPSIS

ntpd [−*flag* [*value*]]... [−**-***opt-name* [[=|]*value*]]...

All arguments must be options.

DESCRIPTION

This manual page documents, briefly, the **ntpd** command.

OPTIONS

−4, **−-ipv4**
Force IPv4 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, **−-ipv6**
Force IPv6 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−a, **−-authreq**
Require crypto authentication. This option must not appear in combination with any of the following options: authnoreq.

Require cryptographic authentication for broadcast client, multicast client and symmetric passive associations. This is the default.

−A, **−-authnoreq**
Do not require crypto authentication. This option must not appear in combination with any of the following options: authreq.

Do not require cryptographic authentication for broadcast client, multicast client and symmetric passive associations. This is almost never a good idea.

−b, **−-bcastsync**
Allow us to sync to broadcast servers.

−c *string*, **−-configfile**=*string*
configuration file name.

The name and path of the configuration file, /etc/ntp.conf by default.

−d, **−-debug-level**
Increase output debug message level. This option may appear an unlimited number of times.

Increase the debugging message output level.

−D *string*, **−-set-debug-level**=*string*
Set the output debug message level. This option may appear an unlimited number of times.

Set the output debugging level. Can be supplied multiple times, but each overrides the previous value(s).

−f *string*, **−-driftfile**=*string*
frequency drift file name.

The name and path of the frequency file, /etc/ntp.drift by default. This is the same operation as the

driftfile driftfile configuration specification in the /etc/ntp.conf file.

−g, **−−panicgate**
Allow the first adjustment to be Big.

Normally, ntpd exits with a message to the system log if the offset exceeds the panic threshold, which is 1000 s by default. This option allows the time to be set to any value without restriction; however, this can happen only once. If the threshold is exceeded after that, ntpd will exit with a message to the system log. This option can be used with the -q and -x options. See the tinker configuration file directive for other options.

−i *string*, **−−jaildir**=*string*
Jail directory.

Chroot the server to the directory jaildir This option also implies that the server attempts to drop root privileges at startup (otherwise, chroot gives very little additional security), and it is only available if the OS supports to run the server without full root privileges. You may need to also specify a -u option.

−I *iface*, **−−interface**=*iface*
Listen on interface. This option may appear an unlimited number of times.

−k *string*, **−−keyfile**=*string*
path to symmetric keys.

Specify the name and path of the symmetric key file. /etc/ntp.keys is the default. This is the same operation as the keys keyfile configuration file directive.

−l *string*, **−−logfile**=*string*
path to the log file.

Specify the name and path of the log file. The default is the system log file. This is the same operation as the logfile logfile configuration file directive.

−L, **−−novirtualips**
Do not listen to virtual IPs.

Do not listen to virtual IPs. The default is to listen.

−M, **−−modifymmtimer**
Modify Multimedia Timer (Windows only).

Set the Windows Multimedia Timer to highest resolution.

−n, **−−nofork**
Do not fork.

−N, **−−nice**
Run at high priority.

To the extent permitted by the operating system, run ntpd at the highest priority.

−p *string*, **−−pidfile**=*string*
path to the PID file.

Specify the name and path of the file used to record ntpd's process ID. This is the same operation

as the pidfile pidfile configuration file directive.

−P *number*, **−-priority**=*number*
Process priority. This option takes an integer number as its argument.

To the extent permitted by the operating system, run ntpd at the specified sched_setscheduler(SCHED_FIFO) priority.

−q, **−-quit**
Set the time and quit.

ntpd will exit just after the first time the clock is set. This behavior mimics that of the ntpdate program, which is to be retired. The -g and -x options can be used with this option. Note: The kernel time discipline is disabled with this option.

−r *string*, **−-propagationdelay**=*string*
Broadcast/propagation delay.

Specify the default propagation delay from the broadcast/multicast server to this client. This is necessary only if the delay cannot be computed automatically by the protocol.

−U *number*, **−-updateinterval**=*number*
interval in seconds between scans for new or dropped interfaces. This option takes an integer number as its argument.

Give the time in seconds between two scans for new or dropped interfaces. For systems with routing socket support the scans will be performed shortly after the interface change has been detected by the system. Use 0 to disable scanning. 60 seconds is the minimum time between scans.

−s *string*, **−-statsdir**=*string*
Statistics file location.

Specify the directory path for files created by the statistics facility. This is the same operation as the statsdir statsdir configuration file directive.

−t *tkey*, **−-trustedkey**=*tkey*
Trusted key number. This option may appear an unlimited number of times.

Add a key number to the trusted key list.

−u *string*, **−-user**=*string*
Run as userid (or userid:groupid).

Specify a user, and optionally a group, to switch to. This option is only available if the OS supports to run the server without full root privileges. Currently, this option is supported under NetBSD (configure with --enable-clockctl) and Linux (configure with --enable-linuxcaps).

−v *nvar*, **−-var**=*nvar*
make ARG an ntp variable (RW). This option may appear an unlimited number of times.

−V *ndvar*, **−-dvar**=*ndvar*
make ARG an ntp variable (RW|DEF). This option may appear an unlimited number of times.

−x, **−-slew**
Slew up to 600 seconds.

Normally, the time is slewed if the offset is less than the step threshold, which is 128 ms by default, and stepped if above the threshold. This option sets the threshold to 600 s, which is well within the accuracy window to set the clock manually. Note: Since the slew rate of typical Unix kernels is limited to 0.5 ms/s, each second of adjustment requires an amortization interval of 2000 s. Thus, an adjustment as much as 600 s will take almost 14 days to complete. This option can be used with the -g and -q options. See the tinker configuration file directive for other options. Note: The kernel time discipline is disabled with this option.

−?, **−-help**
Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−v [{*v*|*c*|*n*}], **−-version**[*={v|c|n}*]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from environment variables named:
NTPD_<option-name> or **NTPD**

AUTHOR

ntp.org
Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

This manual page was *AutoGen*-erated from the **ntpd** option definitions.

NAME

ntpdc − vendor-specific NTP query program

SYNOPSIS

ntpdc [−*flag* [*value*]]... [−**-***opt-name* [[=|]*value*]]...
[host ...]

DESCRIPTION

This manual page documents, briefly, the **ntpdc** command. The [= prog-name =] utility program is used to query an NTP daemon about its current state and to request changes in that state. It uses NTP mode 7 control message formats described in the source code. The program may be run either in interactive mode or controlled using command line arguments. Extensive state and statistics information is available through the [= prog-name =] interface. In addition, nearly all the configuration options which can be specified at startup using ntpd's configuration file may also be specified at run time using [= prog-name =] .

OPTIONS

−4, **−-ipv4**

Force IPv4 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, **−-ipv6**

Force IPv6 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−c *cmd*, **−-command**=*cmd*

run a command and exit. This option may appear an unlimited number of times.

The following argument is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s).

−l, **−-listpeers**

Print a list of the peers. This option must not appear in combination with any of the following options: command.

Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the 'listpeers' interactive command.

−p, **−-peers**

Print a list of the peers. This option must not appear in combination with any of the following options: command.

Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the 'peers' interactive command.

−s, **−-showpeers**

Show a list of the peers. This option must not appear in combination with any of the following options: command.

Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the 'dmpeers' interactive command.

−i, **−-interactive**

Force ntpq to operate in interactive mode. This option must not appear in combination with any of the following options: command, listpeers, peers, showpeers.

Force ntpq to operate in interactive mode. Prompts will be written to the standard output and

commands read from the standard input.

−d, **−-debug-level**
Increase output debug message level. This option may appear an unlimited number of times.

Increase the debugging message output level.

−D *string*, **−-set-debug-level**=*string*
Set the output debug message level. This option may appear an unlimited number of times.

Set the output debugging level. Can be supplied multiple times, but each overrides the previous value(s).

−n, **−-numeric**
numeric host addresses.

Output all host addresses in dotted-quad numeric format rather than converting to the canonical host names.

−?, **−-help**
Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−> [*rcfile*], **−-save-opts**[=*rcfile*]
Save the option state to *rcfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below.

−< *rcfile*, **−-load-opts**=*rcfile*, **−-no-load-opts**
Load options from *rcfile*. The *no-load-opts* form will disable the loading of earlier RC/INI files. *--no-load-opts* is handled early, out of order.

−v [{*v|c|n*}], **−-version**[*={v|c|n}*]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:
NTPDC_<option-name> or **NTPDC**
The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those directories.

AUTHOR

ntp.org
Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

This manual page was *AutoGen*-erated from the **ntpdc** option definitions.

NAME

ntpdsim − NTP daemon simulation program

SYNOPSIS

ntpdsim [−*flag* [*value*]]... [−**-***opt-name* [[=|]*value*]]...

All arguments must be options.

DESCRIPTION

This manual page documents, briefly, the **ntpdsim** command.

OPTIONS

−4, **−-ipv4**

Force IPv4 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, **−-ipv6**

Force IPv6 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−a, **−-authreq**

Require crypto authentication. This option must not appear in combination with any of the following options: authnoreq.

Require cryptographic authentication for broadcast client, multicast client and symmetric passive associations. This is the default.

−A, **−-authnoreq**

Do not require crypto authentication. This option must not appear in combination with any of the following options: authreq.

Do not require cryptographic authentication for broadcast client, multicast client and symmetric passive associations. This is almost never a good idea.

−b, **−-bcastsync**

Allow us to sync to broadcast servers.

−B *string*, **−-simbroadcastdelay**=*string*

Simulator broadcast delay.

−c *string*, **−-configfile**=*string*

configuration file name.

The name and path of the configuration file, /etc/ntp.conf by default.

−C *string*, **−-phasenoise**=*string*

Phase noise level.

−d, **−-debug-level**

Increase output debug message level. This option may appear an unlimited number of times.

Increase the debugging message output level.

−D *string*, **−-set-debug-level**=*string*
Set the output debug message level. This option may appear an unlimited number of times.

Set the output debugging level. Can be supplied multiple times, but each overrides the previous value(s).

−f *string*, **−-driftfile**=*string*
frequency drift file name.

The name and path of the frequency file, /etc/ntp.drift by default. This is the same operation as the driftfile driftfile configuration specification in the /etc/ntp.conf file.

−g, **−-panicgate**
Allow the first adjustment to be Big.

Normally, ntpd exits with a message to the system log if the offset exceeds the panic threshold, which is 1000 s by default. This option allows the time to be set to any value without restriction; however, this can happen only once. If the threshold is exceeded after that, ntpd will exit with a message to the system log. This option can be used with the -q and -x options. See the tinker configuration file directive for other options.

−H *string*, **−-simslew**=*string*
Simuator slew.

−i *string*, **−-jaildir**=*string*
Jail directory.

Chroot the server to the directory jaildir This option also implies that the server attempts to drop root privileges at startup (otherwise, chroot gives very little additional security), and it is only available if the OS supports to run the server without full root privileges. You may need to also specify a -u option.

−I *iface*, **−-interface**=*iface*
Listen on interface. This option may appear an unlimited number of times.

−k *string*, **−-keyfile**=*string*
path to symmetric keys.

Specify the name and path of the symmetric key file. /etc/ntp.keys is the default. This is the same operation as the keys keyfile configuration file directive.

−l *string*, **−-logfile**=*string*
path to the log file.

Specify the name and path of the log file. The default is the system log file. This is the same operation as the logfile logfile configuration file directive.

−L, **−-novirtualips**
Do not listen to virtual IPs.

Do not listen to virtual IPs. The default is to listen.

−M, **−-modifymmtimer**
Modify Multimedia Timer (Windows only).

Set the Windows Multimedia Timer to highest resolution.

−n, **−-nofork**
Do not fork.

−N, **−-nice**
Run at high priority.

To the extent permitted by the operating system, run ntpd at the highest priority.

−O *string*, **−-servertime**=*string*
Server time.

−p *string*, **−-pidfile**=*string*
path to the PID file.

Specify the name and path of the file used to record ntpd's process ID. This is the same operation as the pidfile pidfile configuration file directive.

−P *number*, **−-priority**=*number*
Process priority. This option takes an integer number as its argument.

To the extent permitted by the operating system, run ntpd at the specified sched_setscheduler(SCHED_FIFO) priority.

−q, **−-quit**
Set the time and quit.

ntpd will exit just after the first time the clock is set. This behavior mimics that of the ntpdate program, which is to be retired. The -g and -x options can be used with this option. Note: The kernel time discipline is disabled with this option.

−r *string*, **−-propagationdelay**=*string*
Broadcast/propagation delay.

Specify the default propagation delay from the broadcast/multicast server to this client. This is necessary only if the delay cannot be computed automatically by the protocol.

−U *number*, **−-updateinterval**=*number*
interval in seconds between scans for new or dropped interfaces. This option takes an integer number as its argument.

Give the time in seconds between two scans for new or dropped interfaces. For systems with routing socket support the scans will be performed shortly after the interface change has been detected by the system. Use 0 to disable scanning. 60 seconds is the minimum time between scans.

−s *string*, **−-statsdir**=*string*
Statistics file location.

Specify the directory path for files created by the statistics facility. This is the same operation as the statsdir statsdir configuration file directive.

−S *string*, **−-endsimtime**=*string*
Simulation end time.

−t *tkey*, **−-trustedkey**=*tkey*
Trusted key number. This option may appear an unlimited number of times.

Add a key number to the trusted key list.

−T *string*, **−-freqerr**=*string*
Simulation frequency error.

−W *string*, **−-walknoise**=*string*
Simulation random walk noise.

−u *string*, **−-user**=*string*
Run as userid (or userid:groupid).

Specify a user, and optionally a group, to switch to. This option is only available if the OS supports to run the server without full root privileges. Currently, this option is supported under NetBSD (configure with --enable-clockctl) and Linux (configure with --enable-linuxcaps).

−v *nvar*, **−-var**=*nvar*
make ARG an ntp variable (RW). This option may appear an unlimited number of times.

−V *ndvar*, **−-dvar**=*ndvar*
make ARG an ntp variable (RW|DEF). This option may appear an unlimited number of times.

−x, **−-slew**
Slew up to 600 seconds.

Normally, the time is slewed if the offset is less than the step threshold, which is 128 ms by default, and stepped if above the threshold. This option sets the threshold to 600 s, which is well within the accuracy window to set the clock manually. Note: Since the slew rate of typical Unix kernels is limited to 0.5 ms/s, each second of adjustment requires an amortization interval of 2000 s. Thus, an adjustment as much as 600 s will take almost 14 days to complete. This option can be used with the -g and -q options. See the tinker configuration file directive for other options. Note: The kernel time discipline is disabled with this option.

−Y *string*, **−-ndelay**=*string*
Simulation network delay.

−Z *string*, **−-pdelay**=*string*
Simulation processing delay.

−?, **−-help**
Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−> [*rcfile*], −**-save-opts**[=*rcfile*]
Save the option state to *rcfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below.

−< *rcfile*, −**-load-opts**=*rcfile*, −**-no-load-opts**
Load options from *rcfile*. The *no-load-opts* form will disable the loading of earlier RC/INI files. --*no-load-opts* is handled early, out of order.

−**v** [{*v*|*c*|*n*}], −**-version**[={*v*|*c*|*n*}]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:
NTPDSIM_<option-name> or **NTPDSIM**
The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those directories.

AUTHOR

ntp.org
Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

This manual page was *AutoGen*-erated from the **ntpdsim** option definitions.

NAME

ntpq − standard NTP query program

SYNOPSIS

ntpq [−*flag* [*value*]]... [−*-opt-name* [[=|]*value*]]...
[host ...]

DESCRIPTION

This manual page documents, briefly, the **ntpq** command. The [= prog-name =] utility program is used to query NTP servers which implement the standard NTP mode 6 control message formats defined in Appendix B of the NTPv3 specification RFC1305, requesting information about current state and/or changes in that state. The same formats are used in NTPv4, although some of the variables have changed and new ones added. The description on this page is for the NTPv4 variables. The program may be run either in interactive mode or controlled using command line arguments. Requests to read and write arbitrary variables can be assembled, with raw and pretty-printed output options being available. The [= prog-name =] utility can also obtain and print a list of peers in a common format by sending multiple queries to the server.

If one or more request options is included on the command line when [= prog-name =] is executed, each of the requests will be sent to the NTP servers running on each of the hosts given as command line arguments, or on localhost by default. If no request options are given, [= prog-name =] will attempt to read commands from the standard input and execute these on the NTP server running on the first host given on the command line, again defaulting to localhost when no other host is specified. The [= prog-name =] utility will prompt for commands if the standard input is a terminal device.

The [= prog-name =] utility uses NTP mode 6 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network which permits it. Note that since NTP is a UDP protocol this communication will be somewhat unreliable, especially over large distances in terms of network topology. The [= prog-name =] utility makes one attempt to retransmit requests, and will time requests out if the remote host is not heard from within a suitable timeout time.

Specifying a command line option other than or will cause the specified query (queries) to be sent to the indicated host(s) immediately. Otherwise, [= prog-name =] will attempt to read interactive format commands from the standard input. Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed.

A number of interactive format commands are executed entirely within the [= prog-name =] utility itself and do not result in NTP mode 6 requests being sent to a server. These are described following.

? [command_keyword]

A by itself will print a list of all the command keywords known to this incarnation of [= prog-name =] . A followed by a command keyword will print function and usage information about the command. This command is probably a better source of information about [= prog-name =] than this manual page.

addvars

rmvars variable_name ...

clearvars The data carried by NTP mode 6 messages consists of a list of items of the form where the is ignored, and can be omitted, in requests to the server to read variables. The [= prog-name =] utility maintains an internal list in which data to be included in control messages can be assembled, and sent using the and commands described below. The command allows variables and their optional values to be added to the list. If more than one variable is to be added, the list should be comma-separated and not contain white space. The command can be used to remove individual variables from the list, while the command removes all variables from the list.

authenticate [yes | no] Normally [= prog-name =] does not authenticate requests unless they are write requests. The command causes [= prog-name =] to send authentication with all requests it makes. Authenticated requests causes some servers to handle requests slightly differently, and can occasionally melt the CPU in fuzzballs if you turn authentication on before doing a display. The command causes [= prog-name =] to display whether or not [= prog-name =] is currently autheinticating requests.

cooked Causes output from query commands to be "cooked", so that variables which are recognized by [= prog-name =] will have their values reformatted for human consumption. Variables which [= prog-name =] thinks should have a decodable value but didn't are marked with a trailing] With no argument, displays the current debug level. Otherwise, the debug level is changed to the indicated level.

delay milliseconds Specify a time interval to be added to timestamps included in requests which require authentication. This is used to enable (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. Actually the server does not now require timestamps in authenticated requests, so this command may be obsolete.

host hostname Set the host to which future queries will be sent. Hostname may be either a host name or a numeric address.

hostnames Cm yes | Cm no If is specified, host names are printed in information displays. If is specified, numeric addresses are printed instead. The default is unless modified using the command line switch.

keyid keyid This command allows the specification of a key number to be used to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose.

ntpversion [ ] Sets the NTP version number which [= prog-name =] claims in packets. Defaults to 3, Note that mode 6 control messages (and modes, for that matter) didn't exist in NTP version 1. There appear to be no servers left which demand version 1. With no argument, displays the current NTP version that will be used when communicating with servers.

quit Exit [= prog-name =] .

passwd This command prompts you to type in a password (which will not be echoed) which will be used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose if such requests are to be successful.

raw Causes all output from query commands is printed as received from the remote server. The only formating/interpretation done on the data is to transform nonascii data into a printable (but barely understandable) form.

timeout Ar milliseconds Specify a timeout period for responses to server queries. The default is about 5000 milliseconds. Note that since [= prog-name =] retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

OPTIONS

−4, **−-ipv4**

Force IPv4 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, **−-ipv6**

Force IPv6 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−c *cmd*, **−-command**=*cmd*
run a command and exit. This option may appear an unlimited number of times.

The following argument is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s).

−d, **−-debug-level**
Increase output debug message level. This option may appear an unlimited number of times.

Increase the debugging message output level.

−D *string*, **−-set-debug-level**=*string*
Set the output debug message level. This option may appear an unlimited number of times.

Set the output debugging level. Can be supplied multiple times, but each overrides the previous value(s).

−p, **−-peers**
Print a list of the peers. This option must not appear in combination with any of the following options: interactive.

Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the 'peers' interactive command.

−i, **−-interactive**
Force ntpq to operate in interactive mode. This option must not appear in combination with any of the following options: command, peers.

Force ntpq to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

−n, **−-numeric**
numeric host addresses.

Output all host addresses in dotted-quad numeric format rather than converting to the canonical host names.

−?, **−-help**
Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−> [*rcfile*], **−-save-opts**[=*rcfile*]
Save the option state to *rcfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below.

−< *rcfile*, **−-load-opts**=*rcfile*, **−-no-load-opts**
Load options from *rcfile*. The *no-load-opts* form will disable the loading of earlier RC/INI files. *--no-load-opts* is handled early, out of order.

−v [{*v*|*c*|*n*}], **−-version**[={*v*|*c*|*n*}]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:

NTPQ_<option-name> or **NTPQ**

The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those

directories.

AUTHOR

ntp.org
Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

This manual page was *AutoGen*-erated from the **ntpq** option definitions.

NAME

objcopy − copy and translate object files

SYNOPSIS

objcopy [**−F** *bfdname*|**−−target=***bfdname*]
[**−I** *bfdname*|**−−input−target=***bfdname*]
[**−O** *bfdname*|**−−output−target=***bfdname*]
[**−B** *bfdarch*|**−−binary−architecture=***bfdarch*]
[**−S**|**−−strip−all**]
[**−g**|**−−strip−debug**]
[**−K** *symbolname*|**−−keep−symbol=***symbolname*]
[**−N** *symbolname*|**−−strip−symbol=***symbolname*]
[**−−strip−unneeded−symbol=***symbolname*]
[**−G** *symbolname*|**−−keep−global−symbol=***symbolname*]
[**−L** *symbolname*|**−−localize−symbol=***symbolname*]
[**−W** *symbolname*|**−−weaken−symbol=***symbolname*]
[**−w**|**−−wildcard**]
[**−x**|**−−discard−all**]
[**−X**|**−−discard−locals**]
[**−b** *byte*|**−−byte=***byte*]
[**−i** *interleave*|**−−interleave=***interleave*]
[**−j** *sectionname*|**−−only−section=***sectionname*]
[**−R** *sectionname*|**−−remove−section=***sectionname*]
[**−p**|**−−preserve−dates**]
[**−−debugging**]
[**−−gap−fill=***val*]
[**−−pad−to=***address*]
[**−−set−start=***val*]
[**−−adjust−start=***incr*]
[**−−change−addresses=***incr*]
[**−−change−section−address** *section*{=,+,−}*val*]
[**−−change−section−lma** *section*{=,+,−}*val*]
[**−−change−section−vma** *section*{=,+,−}*val*]
[**−−change−warnings**] [**−−no−change−warnings**]
[**−−set−section−flags** *section=flags*]
[**−−add−section** *sectionname=filename*]
[**−−rename−section** *oldname=newname*[,*flags*]]
[**−−change−leading−char**] [**−−remove−leading−char**]
[**−−srec−len=***ival*] [**−−srec−forceS3**]
[**−−redefine−sym** *old=new*]
[**−−redefine−syms=***filename*]
[**−−weaken**]
[**−−keep−symbols=***filename*]
[**−−strip−symbols=***filename*]
[**−−strip−unneeded−symbols=***filename*]
[**−−keep−global−symbols=***filename*]
[**−−localize−symbols=***filename*]
[**−−weaken−symbols=***filename*]
[**−−alt−machine−code=***index*]
[**−−prefix−symbols=***string*]
[**−−prefix−sections=***string*]
[**−−prefix−alloc−sections=***string*]
[**−−add−gnu−debuglink=***path-to-file*]
[**−−only−keep−debug**]
[**−−writable−text**]

[**−−readonly−text**]
[**−−pure**]
[**−−impure**]
[**−v**|**−−verbose**]
[**−V**|**−−version**]
[**−−help**] [**−−info**]
infile [*outfile*]

DESCRIPTION

The GNU **objcopy** utility copies the contents of an object file to another. **objcopy** uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file. The exact behavior of **objcopy** is controlled by command-line options. Note that **objcopy** should be able to copy a fully linked file between any two formats. However, copying a relocatable object file between any two formats may not work as expected.

objcopy creates temporary files to do its translations and deletes them afterward. **objcopy** uses BFD to do all its translation work; it has access to all the formats described in BFD and thus is able to recognize most formats without being told explicitly.

objcopy can be used to generate S−records by using an output target of **srec** (e.g., use **−O srec**).

objcopy can be used to generate a raw binary file by using an output target of **binary** (e.g., use **−O binary**). When **objcopy** generates a raw binary file, it will essentially produce a memory dump of the contents of the input object file. All symbols and relocation information will be discarded. The memory dump will start at the load address of the lowest section copied into the output file.

When generating an S−record or a raw binary file, it may be helpful to use **−S** to remove sections containing debugging information. In some cases **−R** will be useful to remove sections which contain information that is not needed by the binary file.

Note−−−**objcopy** is not able to change the endianness of its input files. If the input format has an endianness (some formats do not), **objcopy** can only copy the inputs into file formats that have the same endianness or which have no endianness (e.g., **srec**).

OPTIONS

infile
outfile
The input and output files, respectively. If you do not specify *outfile*, **objcopy** creates a temporary file and destructively renames the result with the name of *infile*.

−I *bfdname*
−−input−target=*bfdname*
Consider the source file's object format to be *bfdname*, rather than attempting to deduce it.

−O *bfdname*
−−output−target=*bfdname*
Write the output file using the object format *bfdname*.

−F *bfdname*
−−target=*bfdname*
Use *bfdname* as the object format for both the input and the output file; i.e., simply transfer data from source to destination with no translation.

−B *bfdarch*
−−binary−architecture=*bfdarch*
Useful when transforming a raw binary input file into an object file. In this case the output architecture can be set to *bfdarch*. This option will be ignored if the input file has a known *bfdarch*. You can access this binary data inside a program by referencing the special symbols that are created by the conversion process. These symbols are called _binary_*objfile*_start, _binary_*objfile*_end and _binary_*objfile*_size. e.g. you can transform a picture file into an object file and then access it in your code using these symbols.

−j *sectionname*
−−only−section=*sectionname*
Copy only the named section from the input file to the output file. This option may be given more than once. Note that using this option inappropriately may make the output file unusable.

−R *sectionname*
−−remove−section=*sectionname*
Remove any section named *sectionname* from the output file. This option may be given more than once. Note that using this option inappropriately may make the output file unusable.

−S
−−strip−all
Do not copy relocation and symbol information from the source file.

−g
−−strip−debug
Do not copy debugging symbols or sections from the source file.

−−strip−unneeded
Strip all symbols that are not needed for relocation processing.

−K *symbolname*
−−keep−symbol=*symbolname*
Copy only symbol *symbolname* from the source file. This option may be given more than once.

−N *symbolname*
−−strip−symbol=*symbolname*
Do not copy symbol *symbolname* from the source file. This option may be given more than once.

−−strip−unneeded−symbol=*symbolname*
Do not copy symbol *symbolname* from the source file unless it is needed by a relocation. This option may be given more than once.

−G *symbolname*
−−keep−global−symbol=*symbolname*
Keep only symbol *symbolname* global. Make all other symbols local to the file, so that they are not visible externally. This option may be given more than once.

−L *symbolname*
−−localize−symbol=*symbolname*
Make symbol *symbolname* local to the file, so that it is not visible externally. This option may be given more than once.

−W *symbolname*
−−weaken−symbol=*symbolname*
Make symbol *symbolname* weak. This option may be given more than once.

−w
−−wildcard
Permit regular expressions in *symbolname*s used in other command line options. The question mark (?), asterisk (*), backslash (\) and square brackets ([]) operators can be used anywhere in the symbol name. If the first character of the symbol name is the exclamation point (!) then the sense of the switch is reversed for that symbol. For example:

```
−w −W !foo −W fo*
```

would cause objcopy to weaken all symbols that start with ‘‘fo’’ except for the symbol ‘‘foo’’.

−x
−−discard−all
Do not copy non-global symbols from the source file.

−X
−−discard−locals
Do not copy compiler-generated local symbols. (These usually start with **L** or **..**)

−b *byte*
−−byte=*byte*
Keep only every *byte*th byte of the input file (header data is not affected). *byte* can be in the range from 0 to *interleave*−1, where *interleave* is given by the **−i** or **−−interleave** option, or the default of 4. This option is useful for creating files to program ROM. It is typically used with an `srec` output target.

−i *interleave*
−−interleave=*interleave*
Only copy one out of every *interleave* bytes. Select which byte to copy with the **−b** or **−−byte** option. The default is 4. **objcopy** ignores this option if you do not specify either **−b** or **−−byte**.

−p
−−preserve−dates
Set the access and modification dates of the output file to be the same as those of the input file.

−−debugging
Convert debugging information, if possible. This is not the default because only certain debugging formats are supported, and the conversion process can be time consuming.

−−gap−fill *val*
Fill gaps between sections with *val*. This operation applies to the *load address* (LMA) of the sections. It is done by increasing the size of the section with the lower address, and filling in the extra space created with *val*.

−−pad−to *address*
Pad the output file up to the load address *address*. This is done by increasing the size of the last section. The extra space is filled in with the value specified by **−−gap−fill** (default zero).

−−set−start *val*
Set the start address of the new file to *val*. Not all object file formats support setting the start address.

−−change−start *incr*
−−adjust−start *incr*
Change the start address by adding *incr*. Not all object file formats support setting the start address.

−−change−addresses *incr*
−−adjust−vma *incr*
Change the VMA and LMA addresses of all sections, as well as the start address, by adding *incr*. Some object file formats do not permit section addresses to be changed arbitrarily. Note that this does not relocate the sections; if the program expects sections to be loaded at a certain address, and this option is used to change the sections such that they are loaded at a different address, the program may fail.

−−change−section−address *section***{=,+,−}***val*
−−adjust−section−vma *section***{=,+,−}***val*
Set or change both the VMA address and the LMA address of the named *section*. If **=** is used, the section address is set to *val*. Otherwise, *val* is added to or subtracted from the section address. See the comments under **−−change−addresses**, above. If *section* does not exist in the input file, a warning will be issued, unless **−−no−change−warnings** is used.

−−change−section−lma *section***{=,+,−}***val*
Set or change the LMA address of the named *section*. The LMA address is the address where the section will be loaded into memory at program load time. Normally this is the same as the VMA address, which is the address of the section at program run time, but on some systems, especially those where a program is held in ROM, the two can be different. If **=** is used, the section address is set to *val*. Otherwise, *val* is added to or subtracted from the section address. See the comments under **−−change−addresses**, above. If *section* does not exist in the input file, a warning will be issued,

unless **--no-change-warnings** is used.

--change-section-vma *section***{=,+,-}***val*
Set or change the VMA address of the named *section*. The VMA address is the address where the section will be located once the program has started executing. Normally this is the same as the LMA address, which is the address where the section will be loaded into memory, but on some systems, especially those where a program is held in ROM, the two can be different. If = is used, the section address is set to *val*. Otherwise, *val* is added to or subtracted from the section address. See the comments under **--change-addresses**, above. If *section* does not exist in the input file, a warning will be issued, unless **--no-change-warnings** is used.

--change-warnings
--adjust-warnings
If **--change-section-address** or **--change-section-lma** or **--change-section-vma** is used, and the named section does not exist, issue a warning. This is the default.

--no-change-warnings
--no-adjust-warnings
Do not issue a warning if **--change-section-address** or **--adjust-section-lma** or **--adjust-section-vma** is used, even if the named section does not exist.

--set-section-flags *section=flags*
Set the flags for the named section. The *flags* argument is a comma separated string of flag names. The recognized names are **alloc**, **contents**, **load**, **noload**, **readonly**, **code**, **data**, **rom**, **share**, and **debug**. You can set the **contents** flag for a section which does not have contents, but it is not meaningful to clear the **contents** flag of a section which does have contents — just remove the section instead. Not all flags are meaningful for all object file formats.

--add-section *sectionname=filename*
Add a new section named *sectionname* while copying the file. The contents of the new section are taken from the file *filename*. The size of the section will be the size of the file. This option only works on file formats which can support sections with arbitrary names.

--rename-section *oldname=newname***[,***flags***]**
Rename a section from *oldname* to *newname*, optionally changing the section's flags to *flags* in the process. This has the advantage over usng a linker script to perform the rename in that the output stays as an object file and does not become a linked executable.

This option is particularly helpful when the input format is binary, since this will always create a section called .data. If for example, you wanted instead to create a section called .rodata containing binary data you could use the following command line to achieve it:

```
objcopy -I binary -O <output_format> -B <architecture> \
 --rename-section .data=.rodata,alloc,load,readonly,data,contents \
 <input_binary_file> <output_object_file>
```

--change-leading-char
Some object file formats use special characters at the start of symbols. The most common such character is underscore, which compilers often add before every symbol. This option tells **objcopy** to change the leading character of every symbol when it converts between object file formats. If the object file formats use the same leading character, this option has no effect. Otherwise, it will add a character, or remove a character, or change a character, as appropriate.

--remove-leading-char
If the first character of a global symbol is a special symbol leading character used by the object file format, remove the character. The most common symbol leading character is underscore. This option will remove a leading underscore from all global symbols. This can be useful if you want to link together objects of different file formats with different conventions for symbol names. This is different from **--change-leading-char** because it always changes the symbol name when appropriate, regardless of the object file format of the output file.

−−srec−len=*ival*
Meaningful only for srec output. Set the maximum length of the Srecords being produced to *ival*. This length covers both address, data and crc fields.

−−srec−forceS3
Meaningful only for srec output. Avoid generation of S1/S2 records, creating S3−only record format.

−−redefine−sym *old=new*
Change the name of a symbol *old*, to *new*. This can be useful when one is trying link two things together for which you have no source, and there are name collisions.

−−redefine−syms=*filename*
Apply **−−redefine−sym** to each symbol pair "*old new*" listed in the file *filename*. *filename* is simply a flat file, with one symbol pair per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−weaken
Change all global symbols in the file to be weak. This can be useful when building an object which will be linked against other objects using the **−R** option to the linker. This option is only effective when using an object file format which supports weak symbols.

−−keep−symbols=*filename*
Apply **−−keep−symbol** option to each symbol listed in the file *filename*. *filename* is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−strip−symbols=*filename*
Apply **−−strip−symbol** option to each symbol listed in the file *filename*. *filename* is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−strip−unneeded−symbols=*filename*
Apply **−−strip−unneeded−symbol** option to each symbol listed in the file *filename*. *filename* is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−keep−global−symbols=*filename*
Apply **−−keep−global−symbol** option to each symbol listed in the file *filename*. *filename* is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−localize−symbols=*filename*
Apply **−−localize−symbol** option to each symbol listed in the file *filename*. *filename* is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−weaken−symbols=*filename*
Apply **−−weaken−symbol** option to each symbol listed in the file *filename*. *filename* is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

−−alt−machine−code=*index*
If the output architecture has alternate machine codes, use the *index*th code instead of the default one. This is useful in case a machine is assigned an official code and the tool-chain adopts the new code, but other applications still depend on the original code being used.

−−writable−text
Mark the output text as writable. This option isn't meaningful for all object file formats.

−−readonly−text
Make the output text write protected. This option isn't meaningful for all object file formats.

--pure
Mark the output file as demand paged. This option isn't meaningful for all object file formats.

--impure
Mark the output file as impure. This option isn't meaningful for all object file formats.

--prefix-symbols=*string*
Prefix all symbols in the output file with *string*.

--prefix-sections=*string*
Prefix all section names in the output file with *string*.

--prefix-alloc-sections=*string*
Prefix all the names of all allocated sections in the output file with *string*.

--add-gnu-debuglink=*path-to-file*
Creates a .gnu_debuglink section which contains a reference to *path-to-file* and adds it to the output file.

--only-keep-debug
Strip a file, removing any sections that would be stripped by **--strip-debug** and leaving the debugging sections.

The intention is that this option will be used in conjunction with **--add-gnu-debuglink** to create a two part executable. One a stripped binary which will occupy less space in RAM and in a distribution and the second a debugging information file which is only needed if debugging abilities are required. The suggested procedure to create these files is as follows:

1.<Link the executable as normal. Assuming that is is called>
`foo` then...

1.<Run `objcopy --only-keep-debug foo foo.dbg` to>
create a file containing the debugging info.

1.<Run `objcopy --strip-debug foo` to create a>
stripped executable.

1.<Run `objcopy --add-gnu-debuglink=foo.dbg foo`>
to add a link to the debugging info into the stripped executable.

Note - the choice of `.dbg` as an extension for the debug info file is arbitrary. Also the `--only-keep-debug` step is optional. You could instead do this:

1.<Link the executable as normal.>
1.<Copy `foo` to `foo.full`>
1.<Run `objcopy --strip-debug foo`>
1.<Run `objcopy --add-gnu-debuglink=foo.full foo`>

ie the file pointed to by the **--add-gnu-debuglink** can be the full executable. It does not have to be a file created by the **--only-keep-debug** switch.

-V
--version
Show the version number of **objcopy**.

-v
--verbose
Verbose output: list all object files modified. In the case of archives, **objcopy -V** lists all members of the archive.

--help
Show a summary of the options to **objcopy**.

−−info
Display a list showing all architectures and object formats available.

SEE ALSO

ld (1), *objdump* (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

objdump − display information from object files.

SYNOPSIS

objdump [**−a**|**−−archive−headers**]
[**−b** *bfdname*|**−−target=***bfdname*]
[**−C**|**−−demangle**[*=style*]]
[**−d**|**−−disassemble**]
[**−D**|**−−disassemble−all**]
[**−z**|**−−disassemble−zeroes**]
[**−EB**|**−EL**|**−−endian=**{big | little }]
[**−f**|**−−file−headers**]
[**−−file−start−context**]
[**−g**|**−−debugging**]
[**−e**|**−−debugging−tags**]
[**−h**|**−−section−headers**|**−−headers**]
[**−i**|**−−info**]
[**−j** *section*|**−−section=***section*]
[**−l**|**−−line−numbers**]
[**−S**|**−−source**]
[**−m** *machine*|**−−architecture=***machine*]
[**−M** *options*|**−−disassembler−options=***options*]
[**−p**|**−−private−headers**]
[**−r**|**−−reloc**]
[**−R**|**−−dynamic−reloc**]
[**−s**|**−−full−contents**]
[**−G**|**−−stabs**]
[**−t**|**−−syms**]
[**−T**|**−−dynamic−syms**]
[**−x**|**−−all−headers**]
[**−w**|**−−wide**]
[**−−start−address=***address*]
[**−−stop−address=***address*]
[**−−prefix−addresses**]
[**−−[no−]show−raw−insn**]
[**−−adjust−vma=***offset*]
[**−−special−syms**]
[**−V**|**−−version**]
[**−H**|**−−help**]
objfile...

DESCRIPTION

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

objfile... are the object files to be examined. When you specify archives, **objdump** shows information on each of the member object files.

OPTIONS

The long and short forms of options, shown here as alternatives, are equivalent. At least one option from the list **−a,−d,−D,−e,−f,−g,−G,−h,−H,−p,−r,−R,−s,−S,−t,−T,−V,−x** must be given.

−a
−−archive−header
If any of the *objfile* files are archives, display the archive header information (in a format similar to **ls −l**). Besides the information you could list with **ar tv**, **objdump −a** shows the object file format of

each archive member.

−−adjust−vma=*offset*
When dumping information, first add *offset* to all the section addresses. This is useful if the section addresses do not correspond to the symbol table, which can happen when putting sections at particular addresses when using a format which can not represent section addresses, such as a.out.

−b *bfdname*
−−target=*bfdname*
Specify that the object-code format for the object files is *bfdname*. This option may not be necessary; *objdump* can automatically recognize many formats.

For example,

```
objdump −b oasys −m vax −h fu.o
```

displays summary information from the section headers (**−h**) of *fu.o*, which is explicitly identified (**−m**) as a VAX object file in the format produced by Oasys compilers. You can list the formats available with the **−i** option.

−C
−−demangle[=*style***]**
Decode (*demangle*) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

−g
−−debugging
Display debugging information. This attempts to parse debugging information stored in the file and print it out using a C like syntax. Only certain types of debugging information have been implemented. Some other types are supported by **readelf −w**.

−e
−−debugging−tags
Like **−g**, but the information is generated in a format compatible with ctags tool.

−d
−−disassemble
Display the assembler mnemonics for the machine instructions from *objfile*. This option only disassembles those sections which are expected to contain instructions.

−D
−−disassemble−all
Like **−d**, but disassemble the contents of all sections, not just those expected to contain instructions.

−−prefix−addresses
When disassembling, print the complete address on each line. This is the older disassembly format.

−EB
−EL
−−endian={big|little}
Specify the endianness of the object files. This only affects disassembly. This can be useful when disassembling a file format which does not describe endianness information, such as S−records.

−f
−−file−headers
Display summary information from the overall header of each of the *objfile* files.

−−file−start−context
Specify that when displaying interlisted source code/disassembly (assumes **−S**) from a file that has not yet been displayed, extend the context to the start of the file.

−h
−−section−headers
−−headers
Display summary information from the section headers of the object file.

File segments may be relocated to nonstandard addresses, for example by using the **−Ttext**, **−Tdata**, or **−Tbss** options to **ld**. However, some object file formats, such as a.out, do not store the starting address of the file segments. In those situations, although **ld** relocates the sections correctly, using **objdump −h** to list the file section headers cannot show the correct addresses. Instead, it shows the usual addresses, which are implicit for the target.

−H
−−help
Print a summary of the options to **objdump** and exit.

−i
−−info
Display a list showing all architectures and object formats available for specification with **−b** or **−m**.

−j *name*
−−section=*name*
Display information only for section *name*.

−l
−−line−numbers
Label the display (using debugging information) with the filename and source line numbers corresponding to the object code or relocs shown. Only useful with **−d**, **−D**, or **−r**.

−m *machine*
−−architecture=*machine*
Specify the architecture to use when disassembling object files. This can be useful when disassembling object files which do not describe architecture information, such as S−records. You can list the available architectures with the **−i** option.

−M *options*
−−disassembler−options=*options*
Pass target specific information to the disassembler. Only supported on some targets. If it is necessary to specify more than one disassembler option then multiple **−M** options can be used or can be placed together into a comma separated list.

If the target is an ARM architecture then this switch can be used to select which register name set is used during disassembler. Specifying **−M reg-name-std** (the default) will select the register names as used in ARM's instruction set documentation, but with register 13 called 'sp', register 14 called 'lr' and register 15 called 'pc'. Specifying **−M reg-names-apcs** will select the name set used by the ARM Procedure Call Standard, whilst specifying **−M reg-names-raw** will just use **r** followed by the register number.

There are also two variants on the APCS register naming scheme enabled by **−M reg-names-atpcs** and **−M reg-names-special-atpcs** which use the ARM/Thumb Procedure Call Standard naming conventions. (Either with the normal register names or the special register names).

This option can also be used for ARM architectures to force the disassembler to interpret all instructions as Thumb instructions by using the switch **−−disassembler−options=force−thumb**. This can be useful when attempting to disassemble thumb code produced by other compilers.

For the x86, some of the options duplicate functions of the **−m** switch, but allow finer grained control. Multiple selections from the following may be specified as a comma separated string. **x86−64**, **i386** and **i8086** select disassembly for the given architecture. **intel** and **att** select between intel syntax mode and AT&T syntax mode. **addr32**, **addr16**, **data32** and **data16** specify the default address size and operand size. These four options will be overridden if **x86−64**, **i386** or **i8086** appear later in the option string. Lastly, **suffix**, when in AT&T mode, instructs the disassembler to print a mnemonic suffix even

when the suffix could be inferred by the operands.

For PPC, **booke**, **booke32** and **booke64** select disassembly of BookE instructions. **32** and **64** select PowerPC and PowerPC64 disassembly, respectively.

For MIPS, this option controls the printing of instruction mneumonic names and register names in disassembled instructions. Multiple selections from the following may be specified as a comma separated string, and invalid options are ignored:

`no-aliases`
Print the 'raw' instruction mneumonic instead of some pseudo instruction mneumonic. I.E. print 'daddu' or 'or' instead of 'move', 'sll' instead of 'nop', etc.

`gpr-names=`*`ABI`*
Print GPR (general-purpose register) names as appropriate for the specified ABI. By default, GPR names are selected according to the ABI of the binary being disassembled.

`fpr-names=`*`ABI`*
Print FPR (floating-point register) names as appropriate for the specified ABI. By default, FPR numbers are printed rather than names.

`cp0-names=`*`ARCH`*
Print CP0 (system control coprocessor; coprocessor 0) register names as appropriate for the CPU or architecture specified by *ARCH*. By default, CP0 register names are selected according to the architecture and CPU of the binary being disassembled.

`hwr-names=`*`ARCH`*
Print HWR (hardware register, used by the `rdhwr` instruction) names as appropriate for the CPU or architecture specified by *ARCH*. By default, HWR names are selected according to the architecture and CPU of the binary being disassembled.

`reg-names=`*`ABI`*
Print GPR and FPR names as appropriate for the selected ABI.

`reg-names=`*`ARCH`*
Print CPU-specific register names (CP0 register and HWR names) as appropriate for the selected CPU or architecture.

For any of the options listed above, *ABI* or *ARCH* may be specified as **numeric** to have numbers printed rather than names, for the selected types of registers. You can list the available values of *ABI* and *ARCH* using the `--`**help** option.

-p
--private-headers
Print information that is specific to the object file format. The exact information printed depends upon the object file format. For some object file formats, no additional information is printed.

-r
--reloc
Print the relocation entries of the file. If used with **-d** or **-D**, the relocations are printed interspersed with the disassembly.

-R
--dynamic-reloc
Print the dynamic relocation entries of the file. This is only meaningful for dynamic objects, such as certain types of shared libraries.

-s
--full-contents
Display the full contents of any sections requested. By default all non-empty sections are displayed.

-S

−−source
Display source code intermixed with disassembly, if possible. Implies **−d**.

−−show−raw−insn
When disassembling instructions, print the instruction in hex as well as in symbolic form. This is the default except when **−−prefix−addresses** is used.

−−no−show−raw−insn
When disassembling instructions, do not print the instruction bytes. This is the default when **−−prefix−addresses** is used.

−G
−−stabs
Display the full contents of any sections requested. Display the contents of the .stab and .stab.index and .stab.excl sections from an ELF file. This is only useful on systems (such as Solaris 2.0) in which `.stab` debugging symbol-table entries are carried in an ELF section. In most other file formats, debugging symbol-table entries are interleaved with linkage symbols, and are visible in the **−−syms** output.

−−start−address=*address*
Start displaying data at the specified address. This affects the output of the **−d**, **−r** and **−s** options.

−−stop−address=*address*
Stop displaying data at the specified address. This affects the output of the **−d**, **−r** and **−s** options.

−t
−−syms
Print the symbol table entries of the file. This is similar to the information provided by the **nm** program.

−T
−−dynamic−syms
Print the dynamic symbol table entries of the file. This is only meaningful for dynamic objects, such as certain types of shared libraries. This is similar to the information provided by the **nm** program when given the **−D** (**−−dynamic**) option.

−−special−syms
When displaying symbols include those which the target considers to be special in some way and which would not normally be of interest to the user.

−V
−−version
Print the version number of **objdump** and exit.

−x
−−all−headers
Display all available header information, including the symbol table and relocation entries. Using **−x** is equivalent to specifying all of **−a −f −h −p −r −t**.

−w
−−wide
Format some lines for output devices that have more than 80 columns. Also do not truncate symbol names when they are displayed.

−z
−−disassemble−zeroes
Normally the disassembly output will skip blocks of zeroes. This option directs the disassembler to disassemble those blocks, just like any other data.

SEE ALSO

nm (1), *readelf* (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

od — octal, decimal, hex, ascii dump

SYNOPSIS

od [**-aBbcDdeFfHhIiLlOovXx**] [**-j** *skip*] [**-N** *length*] [**-t** *type_string*]
[[**+**]offset[**.**][**Bb**]] *file ...*

DESCRIPTION

od has been deprecated in favor of hexdump(1).

hexdump(1), if called as **od**, provides compatibility for the options described below. It does not provide compatibility for the **-s** option (see strings(1)) or the **-P**, **-p**, or **-w** options, nor is compatibility provided for the ''label'' component of the offset syntax.

The options are as follows:

-a *One-byte character display*. Display the input offset in octal, followed by sixteen space-separated, three column, space-filled, characters of input data per line. Control characters are printed as their names instead of as C-style escapes.

-B Same as **-o**.

-b *One-byte octal display*. Display the input offset in octal, followed by sixteen space-separated, three column, zero-filled, bytes of input data, in octal, per line. This is the default output style if no other is selected.

-c *One-byte character display*. Display the input offset in octal, followed by sixteen space-separated, three column, space-filled, characters of input data per line. Control characters are printed at C-style escapes, or as three octal digits, if no C escape exists for the character.

-d *Two-byte decimal display*. Display the input offset in octal, followed by eight space-separated, five column, zero-filled, two-byte units of input data, in unsigned decimal, per line.

-e *Eight-byte floating point display*. Display the input offset in octal, followed by two space-separated, twenty-one column, space filled, eight byte units of input data, in floating point, per line.

-F Same as **-e**.

-f *Four-byte floating point display*. Display the input offset in octal, followed by four space-separated, 14 column, space filled, four byte units of input data, in floating point, per line.

-H *Four-byte hex display*. Display the input offset in octal, followed by four space-separated, eight column, zero filled, four byte units of input data, in hex, per line.

-h *Two-byte hex display*. Display the input offset in octal, followed by eight space-separated, four column, zero filled, two byte units of input data, in hex, per line.

-I *Four-byte decimal display*. Display the input offset in octal, followed by four space-separated, eleven column, space filled, four byte units of input data, in decimal, per line.

-i *Two-byte decimal display*. Display the input offset in octal, followed by eight space-separated, six column, space filled, two-byte units of input data, in decimal, per line.

-j *offset*
Skip *offset* bytes from the beginning of the input. By default, *offset* is interpreted as a decimal number. With a leading **0x** or **0X**, *offset* is interpreted as a hexadecimal number, otherwise, with a leading **0**, *offset* is interpreted as an octal number. Appending the character **b**, **k**, or **m** to *offset* causes it to be interpreted as a multiple of 512, 1024, or 1048576, respectively.

-L Same as **-I**.

-l Same as **-I**.

-N *length*
Interpret only *length* bytes of input.

-O *Four-byte octal display*. Display the input offset in octal, followed by four space-separated, eleven column, zero-filled, four-byte units of input data, in octal, per line.

-o *Two-byte octal display*. Display the input offset in octal, followed by eight space-separated, six column, zero-filled, two-byte units of input data, in octal, per line.

-t *type_string*
Specify one or more output types. The *type_string* option-argument must be a string specifying the types to be used when writing the input data. The string must consist of the type specification characters:

a selects US-ASCII output, with control characters replaced with their names instead of as C escape sequences. See also the **_u** conversion provided by hexdump(1).

c selects a standard character based conversion. See also the **_c** conversion provided by hexdump(1).

f selects the floating point output format. This type character can be optionally followed by the characters **4** or **F** to specify four byte floating point output, or **8** or **L** to specify eight byte floating point output. The default output format is eight byte floats. See also the **e** conversion provided by hexdump(1).

d, **o**, **u**, or **x** select decimal, octal, unsigned decimal, or hex output respectively. These types can optionally be followed by **C** to specify *char*-sized output, **S** to specify *short*-sized output, **I** to specify *int*-sized output, **L** to specify *long*-sized output, **1** to specify one-byte output, **2** to specify two-byte output, **4** to specify four-byte output, or **8** to specify eight-byte output. The default output format is in four-byte quantities. See also the **d**, **o**, **u**, and **x** conversions provided by hexdump(1).

-v The **-v** option causes **od** to display all input data. Without the **-v** option, any number of groups of output lines, which would be identical to the immediately preceding group of output lines (except for the input offsets), are replaced with a line comprised of a single asterisk.

-X Same as **-H**.

-x Same as **-h**.

For each input file, **od** sequentially copies the input to standard output, transforming the data according to the options given. If no options are specified, the default display is equivalent to specifying the **-o** option.

od exits 0 on success and >0 if an error occurred.

SEE ALSO

hexdump(1), strings(1)

HISTORY

A **od** command appears in Version 1 AT&T UNIX.

This man page was written in February 2001 by Andrew Brown, shortly after he augmented the deprecated od syntax to include things he felt had been missing for a long time.

NAME

omshell - OMAPI Command Shell

SYNOPSIS

omshell

DESCRIPTION

The OMAPI Command Shell, omshell, provides an interactive way to connect to, query, and possibly change, the ISC DHCP Server's state via OMAPI, the Object Management API. By using OMAPI and omshell, you do not have to stop, make changes, and then restart the DHCP server, but can make the changes while the server is running. Omshell provides a way of accessing OMAPI.

OMAPI is simply a communications mechanism that allows you to manipulate objects. In order to actually *use* omshell, you *must* understand what objects are available and how to use them. Documentation for OMAPI objects can be found in the documentation for the server that provides them - for example, in the **dhcpd(1)** manual page and the **dhclient(1)** manual page.

CONTRIBUTIONS

This software is free software. At various times its development has been underwritten by various organizations, including the ISC and Vixie Enterprises. The development of 3.0 has been funded almost entirely by Nominum, Inc.

At this point development is being shepherded by Ted Lemon, and hosted by the ISC, but the future of this project depends on you. If you have features you want, please consider implementing them.

LOCAL AND REMOTE OBJECTS

Throughout this document, there are references to local and remote objects. Local objects are ones created in omshell with the **new** command. Remote objects are ones on the server: leases, hosts, and groups that the DHCP server knows about. Local and remote objects are associated together to enable viewing and modification of object attributes. Also, new remote objects can be created to match local objects.

OPENING A CONNECTION

omshell is started from the command line. Once omshell is started, there are several commands that can be issued:

server *address*

where address is the IP address of the DHCP server to connect to. If this is not specified, the default server is 127.0.0.1 (localhost).

port *number*

where number is the port that OMAPI listens on. By default, this is 7911.

key *name secret*

This specifies the TSIG key to use to authenticate the OMAPI transactions. *name* is the name of a key defined in *dhcpd.conf* with the **omapi-key** statement. The *secret* is the secret generated from **dnssec-keygen** or another key generation program.

connect

This starts the OMAPI connection to the server as specified by the *server* statement.

CREATING LOCAL OBJECTS

Any object defined in OMAPI can be created, queried, and/or modified. The object types available to OMAPI are defined in **dhcpd(8)** and **dhclient**. When using omshell, objects are first defined locally, manipulated as desired, and then associated with an object on the server. Only one object can be manipulated at a time. To create a local object, use

new *object-type*

object-type is one of group, host, or lease.

At this point, you now have an object that you can set properties on. For example, if a new lease object was created with *new lease*, any of a lease's attributes can be set as follows:

set *attribute-name = value*

Attribute names are defined in dhcpd(8) and **dhclient(8)**. Values should be quoted if they are strings. So, to set a lease's IP address, you would do the following:
set ip-address = 192.168.4.50

ASSOCIATING LOCAL AND REMOTE OBJECTS

At this point, you can query the server for information about this lease, by

open

Now, the local lease object you created and set the IP address for is associated with the corresponding lease object on the DHCP server. All of the lease attributes from the DHCP server are now also the attributes on the local object, and will be shown in omshell.

VIEWING A REMOTE OBJECT

To query a lease of address 192.168.4.50, and find out its attributes, after connecting to the server, take the following steps:

new lease

This creates a new local lease object.

set ip-address = 192.168.4.50

This sets the *local* object's IP address to be 192.168.4.50

open

Now, if a lease with that IP address exists, you will see all the information the DHCP server has about that particular lease. Any data that isn't readily printable text will show up in colon-separated hexadecimal values. In this example, output back from the server for the entire transaction might look like this:

```
> new "lease"
obj: lease
> set ip-address = 192.168.4.50
obj: lease
ip-address = c0:a8:04:32
> open
obj: lease
ip-address = c0:a8:04:32
state = 00:00:00:02
dhcp-client-identifier = 01:00:10:a4:b2:36:2c
client-hostname = "wendelina"
subnet = 00:00:00:06
pool = 00:00:00:07
hardware-address = 00:10:a4:b2:36:2c
hardware-type = 00:00:00:01
ends = dc:d9:0d:3b
starts = 5c:9f:04:3b
tstp = 00:00:00:00
tsfp = 00:00:00:00
cltt = 00:00:00:00
```

As you can see here, the IP address is represented in hexadecimal, as are the starting and ending times of the lease.

MODIFYING A REMOTE OBJECT

Attributes of remote objects are updated by using the **set** command as before, and then issuing an **update** command. The **set** command sets the attributes on the current local object, and the **update** command pushes those changes out to the server.

Continuing with the previous example, if a **set client-hostname = "something-else"** was issued, followed by an **update** command, the output would look about like this:

```
> set client-hostname = "something-else"
obj: lease
ip-address = c0:a8:04:32
state = 00:00:00:02
dhcp-client-identifier = 01:00:10:a4:b2:36:2c
client-hostname = "something-else"
subnet = 00:00:00:06
pool = 00:00:00:07
hardware-address = 00:10:a4:b2:36:2c
hardware-type = 00:00:00:01
ends = dc:d9:0d:3b
starts = 5c:9f:04:3b
tstp = 00:00:00:00
tsfp = 00:00:00:00
cltt = 00:00:00:00
> update
obj: lease
ip-address = c0:a8:04:32
state = 00:00:00:02
dhcp-client-identifier = 01:00:10:a4:b2:36:2c
client-hostname = "something-else"
subnet = 00:00:00:06
pool = 00:00:00:07
hardware-address = 00:10:a4:b2:36:2c
hardware-type = 00:00:00:01
ends = dc:d9:0d:3b
starts = 5c:9f:04:3b
tstp = 00:00:00:00
tsfp = 00:00:00:00
cltt = 00:00:00:00
```

NEW REMOTE OBJECTS

New remote objects are created much in the same way that existing server objects are modified. Create a local object using **new**, set the attributes as you'd wish them to be, and then create the remote object with the same properties by using

create

Now a new object exists on the DHCP server which matches the properties that you gave your local object. Objects created via OMAPI are saved into the dhcpd.leases file.

For example, if a new host with the IP address of 192.168.4.40 needs to be created it would be done as follows:

```
> new host
obj: host
> set name = "some-host"
obj: host
name = "some-host"
> set hardware-address = 00:80:c7:84:b1:94
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
> set hardware-type = 1
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
```

```
hardware-type = 1
> set ip-address = 192.168.4.40
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 1
ip-address = c0:a8:04:28
> create
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = c0:a8:04:28
>
```

Your dhcpd.leases file would then have an entry like this in it:

```
host some-host {
  dynamic;
  hardware ethernet 00:80:c7:84:b1:94;
  fixed-address 192.168.4.40;
}
```

The *dynamic;* line is to denote that this host entry did not come from dhcpd.conf, but was created dynamically via OMAPI.

RESETTING ATTRIBUTES

If you want to remove an attribute from an object, you can do this with the **unset** command. Once you have unset an attribute, you must use the **update** command to update the remote object. So, if the host "some-host" from the previous example will not have a static IP address anymore, the commands in omshell would look like this:

```
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = c0:a8:04:28
> unset ip-address
obj: host
name = "some-host"
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = <null>
>
```

REFRESHING OBJECTS

A local object may be refreshed with the current remote object properties using the **refresh** command. This is useful for object that change periodically, like leases, to see if they have been updated. This isn't particularly useful for hosts.

DELETING OBJECTS

Any remote object that can be created can also be destroyed. This is done by creating a new local object, setting attributes, associating the local and remote object using **open**, *and then using the* **remove** command. If the host "some-host" from before was created in error, this could be corrected as follows:

```
obj: host
name = "some-host"
```

```
hardware-address = 00:80:c7:84:b1:94
hardware-type = 00:00:00:01
ip-address = c0:a8:04:28
> remove
obj: <null>
>
```

HELP

The **help** command will print out all of the commands available in omshell, with some syntax pointers.

SEE ALSO

dhcpctl(3), omapi(3), dhcpd(8), dhclient(8), dhcpd.conf(5), dhclient.conf(5).

AUTHOR

omshell was written by Ted Lemon of Nominum, Inc. Information about Nominum and support contracts for DHCP and BIND can be found at **http://www.nominum.com. This preliminary documentation was** written by Wendy Verschoor of Nominum, Inc., while she was testing omshell.

NAME

openssl − OpenSSL command line tool

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl *command* [*command_opts*] [*command_args*]

openssl [**list-standard-commands** | **list-message-digest-commands** | **list-cipher-commands** | **list-cipher-algorithms** | **list-message-digest-algorithms** | **list-public-key-algorithms**]

openssl no−*XXX* [*arbitrary options*]

DESCRIPTION

OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and related cryptography standards required by them.

The **openssl** program is a command line tool for using the various cryptography functions of OpenSSL's **crypto** library from the shell. It can be used for

- Creation and management of private keys, public keys and parameters
- Public key cryptographic operations
- Creation of X.509 certificates, CSRs and CRLs
- Calculation of Message Digests
- Encryption and Decryption with Ciphers
- SSL/TLS Client and Server Tests
- Handling of S/MIME signed or encrypted mail
- Time Stamp requests, generation and verification

COMMAND SUMMARY

The **openssl** program provides a rich variety of commands (*command* in the SYNOPSIS above), each of which often has a wealth of options and arguments (*command_opts* and *command_args* in the SYNOPSIS).

The pseudo-commands **list-standard-commands**, **list-message-digest-commands**, and **list-cipher-commands** output a list (one entry per line) of the names of all standard commands, message digest commands, or cipher commands, respectively, that are available in the present **openssl** utility.

The pseudo-commands **list-cipher-algorithms** and **list-message-digest-algorithms** list all cipher and message digest names, one entry per line. Aliases are listed as:

```
from => to
```

The pseudo-command **list-public-key-algorithms** lists all supported public key algorithms.

The pseudo-command **no−***XXX* tests whether a command of the specified name is available. If no command named *XXX* exists, it returns 0 (success) and prints **no−***XXX*; otherwise it returns 1 and prints *XXX*. In both cases, the output goes to **stdout** and nothing is printed to **stderr**. Additional command line arguments are always ignored. Since for each cipher there is a command of the same name, this provides an easy way for shell scripts to test for the availability of ciphers in the **openssl** program. (**no−***XXX* is not able to detect pseudo-commands such as **quit**, **list−...−commands**, or **no−***XXX* itself.)

STANDARD COMMANDS

asn1parse Parse an ASN.1 sequence.

ca Certificate Authority (CA) Management.

ciphers Cipher Suite Description Determination.

crl Certificate Revocation List (CRL) Management.

crl2pkcs7 CRL to PKCS#7 Conversion.

dgst Message Digest Calculation.

dh Diffie-Hellman Parameter Management. Obsoleted by **dhparam**.

dsa DSA Data Management.

dsaparam DSA Parameter Generation and Management. Superseded by **genpkey** and **pkeyparam**

enc Encoding with Ciphers.

errstr Error Number to Error String Conversion.

dhparam Generation and Management of Diffie-Hellman Parameters. Superseded by **genpkey** and **pkeyparam**

gendh Generation of Diffie-Hellman Parameters. Obsoleted by **dhparam**.

gendsa Generation of DSA Private Key from Parameters. Superseded by **genpkey** and **pkey**

genpkey Generation of Private Key or Parameters.

genrsa Generation of RSA Private Key. Superceded by **genpkey**.

ocsp Online Certificate Status Protocol utility.

passwd Generation of hashed passwords.

pkcs12 PKCS#12 Data Management.

pkcs7 PKCS#7 Data Management.

pkey Public and private key management.

pkeyutl Public key algorithm cryptographic operation utility.

pkeyparam Public key algorithm parameter management.

rand Generate pseudo-random bytes.

req PKCS#10 X.509 Certificate Signing Request (CSR) Management.

rsa RSA key management.

rsautl RSA utility for signing, verification, encryption, and decryption. Superseded by **pkeyutl**

s_client This implements a generic SSL/TLS client which can establish a transparent connection to a remote server speaking SSL/TLS. It's intended for testing purposes only and provides only rudimentary interface functionality but internally uses mostly all functionality of the OpenSSL **ssl** library.

s_server This implements a generic SSL/TLS server which accepts connections from remote clients speaking SSL/TLS. It's intended for testing purposes only and provides only rudimentary interface functionality but internally uses mostly all functionality of the OpenSSL **ssl** library. It provides both an own command line oriented protocol for testing SSL functions and a simple HTTP response facility to emulate an SSL/TLS−aware webserver.

s_time SSL Connection Timer.

sess_id SSL Session Data Management.

smime S/MIME mail processing.

speed Algorithm Speed Measurement.

ts Time Stamping Authority tool (client/server)

verify X.509 Certificate Verification.

version OpenSSL Version Information.

x509 X.509 Certificate Data Management.

MESSAGE DIGEST COMMANDS

md2 MD2 Digest

md5 MD5 Digest

mdc2 MDC2 Digest

rmd160 RMD−160 Digest

sha SHA Digest

sha1 SHA−1 Digest

sha224
SHA−224 Digest

sha256
SHA−256 Digest

sha384
SHA−384 Digest

sha512
SHA−512 Digest

ENCODING AND CIPHER COMMANDS

base64 Base64 Encoding

bf bf-cbc bf-cfb bf-ecb bf-ofb
Blowfish Cipher

cast cast-cbc
CAST Cipher

cast5−cbc cast5−cfb cast5−ecb cast5−ofb
CAST5 Cipher

des des-cbc des-cfb des-ecb des-ede des-ede-cbc des-ede-cfb des-ede-ofb des-ofb
DES Cipher

des3 desx des−ede3 des−ede3−cbc des−ede3−cfb des−ede3−ofb
Triple-DES Cipher

idea idea-cbc idea-cfb idea-ecb idea-ofb
IDEA Cipher

rc2 rc2−cbc rc2−cfb rc2−ecb rc2−ofb
RC2 Cipher

rc4 RC4 Cipher

rc5 rc5−cbc rc5−cfb rc5−ecb rc5−ofb
RC5 Cipher

PASS PHRASE ARGUMENTS

Several commands accept password arguments, typically using **−passin** and **−passout** for input and output passwords respectively. These allow the password to be obtained from a variety of sources. Both of these options take a single argument whose format is described below. If no password argument is given and a password is required then the user is prompted to enter one: this will typically be read from the current terminal with echoing turned off.

pass:password

the actual password is **password**. Since the password is visible to utilities (like 'ps' under Unix) this form should only be used where security is not important.

env:var obtain the password from the environment variable **var**. Since the environment of other processes is visible on certain platforms (e.g. ps under certain Unix OSes) this option should be used with caution.

file:pathname
the first line of **pathname** is the password. If the same **pathname** argument is supplied to **−passin** and **−passout** arguments then the first line will be used for the input password and the next line for the output password. **pathname** need not refer to a regular file: it could for example refer to a device or named pipe.

fd:number
read the password from the file descriptor **number**. This can be used to send the data via a pipe for example.

stdin read the password from standard input.

SEE ALSO

openssl_asn1parse (1), *openssl_ca* (1), *openssl.cnf* (5), *openssl_crl* (1), *openssl_crl2pkcs7* (1), *openssl_dgst* (1), *openssl_dhparam* (1), *openssl_dsa* (1), *openssl_dsaparam* (1), *openssl_enc* (1), *openssl_gendsa* (1), *genpkey* (1), *openssl_genrsa* (1), *openssl_nseq* (1), *openssl* (1), *openssl_passwd* (1), *openssl_pkcs12* (1), *openssl_pkcs7* (1), *openssl_pkcs8* (1), *openssl_rand* (1), *openssl_req* (1), *openssl_rsa* (1), *openssl_rsautl* (1), *openssl_s_client* (1), *openssl_s_server* (1), *s_time* (1), *openssl_smime* (1), *openssl_spkac* (1), *openssl_verify* (1), *openssl_version* (1), *openssl_x509* (1), *crypto* (3), *ssl* (3), *x509v3_config* (5)

HISTORY

The *openssl* (1) document appeared in OpenSSL 0.9.2. The **list−***XXX***−commands** pseudo-commands were added in OpenSSL 0.9.3; The **list−***XXX***−algorithms** pseudo-commands were added in OpenSSL 0.9.9; the **no−***XXX* pseudo-commands were added in OpenSSL 0.9.5a. For notes on the availability of other commands, see their individual manual pages.

NAME

CA.pl − friendlier interface for OpenSSL certificate programs

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

CA.pl [**−?**] [**−h**] [**−help**] [**−newcert**] [**−newreq**] [**−newreq−nodes**] [**−newca**] [**−xsign**] [**−sign**] [**−signreq**] [**−signcert**] [**−verify**] [**files**]

DESCRIPTION

The **CA.pl** script is a perl script that supplies the relevant command line arguments to the **openssl** command for some common certificate operations. It is intended to simplify the process of certificate creation and management by the use of some simple options.

COMMAND OPTIONS

?, **−h**, **−help**

prints a usage message.

−newcert

creates a new self signed certificate. The private key and certificate are written to the file ‘‘newreq.pem’’.

−newreq

creates a new certificate request. The private key and request are written to the file ‘‘newreq.pem’’.

−newreq−nodes

is like **−newreq** except that the private key will not be encrypted.

−newca

creates a new CA hierarchy for use with the **ca** program (or the **−signcert** and **−xsign** options). The user is prompted to enter the filename of the CA certificates (which should also contain the private key) or by hitting ENTER details of the CA will be prompted for. The relevant files and directories are created in a directory called ‘‘demoCA’’ in the current directory.

−pkcs12

create a PKCS#12 file containing the user certificate, private key and CA certificate. It expects the user certificate and private key to be in the file ‘‘newcert.pem’’ and the CA certificate to be in the file demoCA/cacert.pem, it creates a file ‘‘newcert.p12’’. This command can thus be called after the **−sign** option. The PKCS#12 file can be imported directly into a browser. If there is an additional argument on the command line it will be used as the ‘‘friendly name’’ for the certificate (which is typically displayed in the browser list box), otherwise the name ‘‘My Certificate’’ is used.

−sign, **−signreq**, **−xsign**

calls the **ca** program to sign a certificate request. It expects the request to be in the file ‘‘newreq.pem’’. The new certificate is written to the file ‘‘newcert.pem’’ except in the case of the **−xsign** option when it is written to standard output.

−signCA

this option is the same as the **−signreq** option except it uses the configuration file section **v3_ca** and so makes the signed request a valid CA certificate. This is useful when creating intermediate CA from a root CA.

−signcert

this option is the same as **−sign** except it expects a self signed certificate to be present in the file ‘‘newreq.pem’’.

−verify

verifies certificates against the CA certificate for ‘‘demoCA’’. If no certificates are specified on the command line it tries to verify the file ‘‘newcert.pem’’.

files

one or more optional certificate file names for use with the **−verify** command.

EXAMPLES

Create a CA hierarchy:

```
CA.pl -newca
```

Complete certificate creation example: create a CA, create a request, sign the request and finally create a PKCS#12 file containing it.

```
CA.pl -newca
CA.pl -newreq
CA.pl -signreq
CA.pl -pkcs12 "My Test Certificate"
```

DSA CERTIFICATES

Although the **CA.pl** creates RSA CAs and requests it is still possible to use it with DSA certificates and requests using the *openssl_req* (1) command directly. The following example shows the steps that would typically be taken.

Create some DSA parameters:

```
openssl dsaparam -out dsap.pem 1024
```

Create a DSA CA certificate and private key:

```
openssl req -x509 -newkey dsa:dsap.pem -keyout cacert.pem -out cacert.pem
```

Create the CA directories and files:

```
CA.pl -newca
```

enter cacert.pem when prompted for the CA file name.

Create a DSA certificate request and private key (a different set of parameters can optionally be created first):

```
openssl req -out newreq.pem -newkey dsa:dsap.pem
```

Sign the request:

```
CA.pl -signreq
```

NOTES

Most of the filenames mentioned can be modified by editing the **CA.pl** script.

If the demoCA directory already exists then the **−newca** command will not overwrite it and will do nothing. This can happen if a previous call using the **−newca** option terminated abnormally. To get the correct behaviour delete the demoCA directory if it already exists.

Under some environments it may not be possible to run the **CA.pl** script directly (for example Win32) and the default configuration file location may be wrong. In this case the command:

```
perl -S CA.pl
```

can be used and the **OPENSSL_CONF** environment variable changed to point to the correct path of the configuration file ‘‘openssl.cnf’’.

The script is intended as a simple front end for the **openssl** program for use by a beginner. Its behaviour isn’t always what is wanted. For more control over the behaviour of the certificate commands call the **openssl** command directly.

ENVIRONMENT VARIABLES

The variable **OPENSSL_CONF** if defined allows an alternative configuration file location to be specified, it should contain the full path to the configuration file, not just its directory.

SEE ALSO

openssl_x509 (1), *openssl_ca* (1), *openssl_req* (1), *openssl_pkcs12* (1), *openssl.cnf* (5)

NAME

asn1parse − ASN.1 parsing tool

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl asn1parse [**−inform PEM|DER**] [**−in filename**] [**−out filename**] [**−noout**] [**−offset number**] [**−length number**] [**−i**] [**−oid filename**] [**−strparse offset**] [**−genstr string**] [**−genconf file**]

DESCRIPTION

The **asn1parse** command is a diagnostic utility that can parse ASN.1 structures. It can also be used to extract data from ASN.1 formatted data.

OPTIONS

−inform DER|PEM
: the input format. **DER** is binary format and **PEM** (the default) is base64 encoded.

−in filename
: the input file, default is standard input

−out filename
: output file to place the DER encoded data into. If this option is not present then no data will be output. This is most useful when combined with the **−strparse** option.

−noout
: don't output the parsed version of the input file.

−offset number
: starting offset to begin parsing, default is start of file.

−length number
: number of bytes to parse, default is until end of file.

−i indents the output according to the ''depth'' of the structures.

−oid filename
: a file containing additional OBJECT IDENTIFIERs (OIDs). The format of this file is described in the NOTES section below.

−strparse offset
: parse the contents octets of the ASN.1 object starting at **offset**. This option can be used multiple times to ''drill down'' into a nested structure.

−genstr string, **−genconf file**
: generate encoded data based on **string**, **file** or both using *ASN1_generate_nconf* (3) format. If **file** only is present then the string is obtained from the default section using the name **asn1**. The encoded data is passed through the ASN1 parser and printed out as though it came from a file, the contents can thus be examined and written to a file using the **out** option.

OUTPUT

The output will typically contain lines like this:

```
  0:d=0  hl=4 l= 681 cons: SEQUENCE
```

.....

```
  229:d=3  hl=3 l= 141 prim: BIT STRING
  373:d=2  hl=3 l= 162 cons: cont [ 3 ]
  376:d=3  hl=3 l= 159 cons: SEQUENCE
  379:d=4  hl=2 l=  29 cons: SEQUENCE
  381:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Identifier
  386:d=5  hl=2 l=  22 prim: OCTET STRING
  410:d=4  hl=2 l= 112 cons: SEQUENCE
  412:d=5  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Identifier
  417:d=5  hl=2 l= 105 prim: OCTET STRING
  524:d=4  hl=2 l=  12 cons: SEQUENCE
```

.....

This example is part of a self signed certificate. Each line starts with the offset in decimal. **d=XX** specifies the current depth. The depth is increased within the scope of any SET or SEQUENCE. **hl=XX** gives the header length (tag and length octets) of the current type. **l=XX** gives the length of the contents octets.

The **−i** option can be used to make the output more readable.

Some knowledge of the ASN.1 structure is needed to interpret the output.

In this example the BIT STRING at offset 229 is the certificate public key. The contents octets of this will contain the public key information. This can be examined using the option **−strparse 229** to yield:

```
    0:d=0  hl=3 l= 137 cons: SEQUENCE
    3:d=1  hl=3 l= 129 prim: INTEGER           :E5D21E1F5C8D208EA7A2166C7FAF9F6BDF20
  135:d=1  hl=2 l=   3 prim: INTEGER           :010001
```

NOTES

If an OID is not part of OpenSSL's internal table it will be represented in numerical form (for example 1.2.3.4). The file passed to the **−oid** option allows additional OIDs to be included. Each line consists of three columns, the first column is the OID in numerical format and should be followed by white space. The second column is the ''short name'' which is a single word followed by white space. The final column is the rest of the line and is the ''long name''. **asn1parse** displays the long name. Example:

```
1.2.3.4     shortName   A long name
```

EXAMPLES

Parse a file:

```
openssl asn1parse -in file.pem
```

Parse a DER file:

```
openssl asn1parse -inform DER -in file.der
```

Generate a simple UTF8String:

```
openssl asn1parse -genstr 'UTF8:Hello World'
```

Generate and write out a UTF8String, don't print parsed output:

```
openssl asn1parse -genstr 'UTF8:Hello World' -noout -out utf8.der
```

Generate using a config file:

```
openssl asn1parse -genconf asn1.cnf -noout -out asn1.der
```

Example config file:

```
asn1=SEQUENCE:seq_sect

[seq_sect]

field1=BOOL:TRUE
field2=EXP:0, UTF8:some random string
```

BUGS

There should be options to change the format of output lines. The output of some ASN.1 types is not well handled (if at all).

SEE ALSO

ASN1_generate_nconf (3)

NAME

ca − sample minimal CA application

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl ca [**−verbose**] [**−config filename**] [**−name section**] [**−gencrl**] [**−revoke file**] [**−crl_reason reason**] [**−crl_hold instruction**] [**−crl_compromise time**] [**−crl_CA_compromise time**] [**−crldays days**] [**−crlhours hours**] [**−crlexts section**] [**−startdate date**] [**−enddate date**] [**−days arg**] [**−md arg**] [**−policy arg**] [**−keyfile arg**] [**−key arg**] [**−passin arg**] [**−cert file**] [**−selfsign**] [**−in file**] [**−out file**] [**−notext**] [**−outdir dir**] [**−infiles**] [**−spkac file**] [**−ss_cert file**] [**−preserveDN**] [**−noemailDN**] [**−batch**] [**−msie_hack**] [**−extensions section**] [**−extfile section**] [**−engine id**] [**−subj arg**] [**−utf8**] [**−multivalue−rdn**]

DESCRIPTION

The **ca** command is a minimal CA application. It can be used to sign certificate requests in a variety of forms and generate CRLs it also maintains a text database of issued certificates and their status.

The options descriptions will be divided into each purpose.

CA OPTIONS

−config filename
specifies the configuration file to use.

−name section
specifies the configuration file section to use (overrides **default_ca** in the **ca** section).

−in filename
an input filename containing a single certificate request to be signed by the CA.

−ss_cert filename
a single self signed certificate to be signed by the CA.

−spkac filename
a file containing a single Netscape signed public key and challenge and additional field values to be signed by the CA. See the **SPKAC FORMAT** section for information on the required format.

−infiles
if present this should be the last option, all subsequent arguments are assumed to the the names of files containing certificate requests.

−out filename
the output file to output certificates to. The default is standard output. The certificate details will also be printed out to this file.

−outdir directory
the directory to output certificates to. The certificate will be written to a filename consisting of the serial number in hex with ‘‘.pem’’ appended.

−cert
the CA certificate file.

−keyfile filename
the private key to sign requests with.

−key password
the password used to encrypt the private key. Since on some systems the command line arguments are visible (e.g. Unix with the ’ps’ utility) this option should be used with caution.

−selfsign
indicates the issued certificates are to be signed with the key the certificate requests were signed with (given with **−keyfile**). Cerificate requests signed with a different key are ignored. If **−spkac**, **−ss_cert** or **−gencrl** are given, **−selfsign** is ignored.

A consequence of using **−selfsign** is that the self-signed certificate appears among the entries in the

certificate database (see the configuration option **database**), and uses the same serial number counter as all other certificates sign with the self-signed certificate.

−passin arg
the key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−verbose
this prints extra details about the operations being performed.

−notext
don't output the text form of a certificate to the output file.

−startdate date
this allows the start date to be explicitly set. The format of the date is YYMMDDHHMMSSZ (the same as an ASN1 UTCTime structure).

−enddate date
this allows the expiry date to be explicitly set. The format of the date is YYMMDDHHMMSSZ (the same as an ASN1 UTCTime structure).

−days arg
the number of days to certify the certificate for.

−md alg
the message digest to use. Possible values include md5, sha1 and mdc2. This option also applies to CRLs.

−policy arg
this option defines the CA ''policy'' to use. This is a section in the configuration file which decides which fields should be mandatory or match the CA certificate. Check out the **POLICY FORMAT** section for more information.

−msie_hack
this is a legacy option to make **ca** work with very old versions of the IE certificate enrollment control ''certenr3''. It used UniversalStrings for almost everything. Since the old control has various security bugs its use is strongly discouraged. The newer control ''Xenroll'' does not need this option.

−preserveDN
Normally the DN order of a certificate is the same as the order of the fields in the relevant policy section. When this option is set the order is the same as the request. This is largely for compatibility with the older IE enrollment control which would only accept certificates if their DNs match the order of the request. This is not needed for Xenroll.

−noemailDN
The DN of a certificate can contain the EMAIL field if present in the request DN, however it is good policy just having the e−mail set into the altName extension of the certificate. When this option is set the EMAIL field is removed from the certificate' subject and set only in the, eventually present, extensions. The **email_in_dn** keyword can be used in the configuration file to enable this behaviour.

−batch
this sets the batch mode. In this mode no questions will be asked and all certificates will be certified automatically.

−extensions section
the section of the configuration file containing certificate extensions to be added when a certificate is issued (defaults to **x509_extensions** unless the **−extfile** option is used). If no extension section is present then, a V1 certificate is created. If the extension section is present (even if it is empty), then a V3 certificate is created. See the:w *x509v3_config* (5) manual page for details of the extension section format.

−extfile file
an additional configuration file to read certificate extensions from (using the default section unless the **−extensions** option is also used).

−engine id
specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

−subj arg
supersedes subject name given in the request. The arg must be formatted as */type0=value0/type1=value1/type2=...*, characters may be escaped by \ (backslash), no spaces are skipped.

−utf8
this option causes field values to be interpreted as UTF8 strings, by default they are interpreted as ASCII. This means that the field values, whether prompted from a terminal or obtained from a configuration file, must be valid UTF8 strings.

−multivalue−rdn
this option causes the −subj argument to be interpretedt with full support for multivalued RDNs. Example:

/DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe

If −multi−rdn is not used then the UID value is *123456+CN=John Doe*.

CRL OPTIONS

−gencrl
this option generates a CRL based on information in the index file.

−crldays num
the number of days before the next CRL is due. That is the days from now to place in the CRL nextUpdate field.

−crlhours num
the number of hours before the next CRL is due.

−revoke filename
a filename containing a certificate to revoke.

−crl_reason reason
revocation reason, where **reason** is one of: **unspecified**, **keyCompromise**, **CACompromise**, **affiliationChanged**, **superseded**, **cessationOfOperation**, **certificateHold** or **removeFromCRL**. The matching of **reason** is case insensitive. Setting any revocation reason will make the CRL v2.

In practive **removeFromCRL** is not particularly useful because it is only used in delta CRLs which are not currently implemented.

−crl_hold instruction
This sets the CRL revocation reason code to **certificateHold** and the hold instruction to **instruction** which must be an OID. Although any OID can be used only **holdInstructionNone** (the use of which is discouraged by RFC2459) **holdInstructionCallIssuer** or **holdInstructionReject** will normally be used.

−crl_compromise time
This sets the revocation reason to **keyCompromise** and the compromise time to **time**. **time** should be in GeneralizedTime format that is **YYYYMMDDHHMMSSZ**.

−crl_CA_compromise time
This is the same as **crl_compromise** except the revocation reason is set to **CACompromise**.

−crlexts section

the section of the configuration file containing CRL extensions to include. If no CRL extension section is present then a V1 CRL is created, if the CRL extension section is present (even if it is empty) then a V2 CRL is created. The CRL extensions specified are CRL extensions and **not** CRL entry extensions. It should be noted that some software (for example Netscape) can't handle V2 CRLs. See *x509v3_config* (5) manual page for details of the extension section format.

CONFIGURATION FILE OPTIONS

The section of the configuration file containing options for **ca** is found as follows: If the **−name** command line option is used, then it names the section to be used. Otherwise the section to be used must be named in the **default_ca** option of the **ca** section of the configuration file (or in the default section of the configuration file). Besides **default_ca**, the following options are read directly from the **ca** section:

RANDFILE
preserve
msie_hack With the exception of **RANDFILE**, this is probably a bug and may change in future releases.

Many of the configuration file options are identical to command line options. Where the option is present in the configuration file and the command line the command line value is used. Where an option is described as mandatory then it must be present in the configuration file or the command line equivalent (if any) used.

oid_file

This specifies a file containing additional **OBJECT IDENTIFIERS**. Each line of the file should consist of the numerical form of the object identifier followed by white space then the short name followed by white space and finally the long name.

oid_section

This specifies a section in the configuration file containing extra object identifiers. Each line should consist of the short name of the object identifier followed by = and the numerical form. The short and long names are the same when this option is used.

new_certs_dir

the same as the **−outdir** command line option. It specifies the directory where new certificates will be placed. Mandatory.

certificate

the same as **−cert**. It gives the file containing the CA certificate. Mandatory.

private_key

same as the **−keyfile** option. The file containing the CA private key. Mandatory.

RANDFILE

a file used to read and write random number seed information, or an EGD socket (see *RAND_egd* (3)).

default_days

the same as the **−days** option. The number of days to certify a certificate for.

default_startdate

the same as the **−startdate** option. The start date to certify a certificate for. If not set the current time is used.

default_enddate

the same as the **−enddate** option. Either this option or **default_days** (or the command line equivalents) must be present.

default_crl_hours default_crl_days

the same as the **−crlhours** and the **−crldays** options. These will only be used if neither command line option is present. At least one of these must be present to generate a CRL.

default_md

the same as the **−md** option. The message digest to use. Mandatory.

database
the text database file to use. Mandatory. This file must be present though initially it will be empty.

unique_subject
if the value **yes** is given, the valid certificate entries in the database must have unique subjects. if the value **no** is given, several valid certificate entries may have the exact same subject. The default value is **yes**, to be compatible with older (pre 0.9.8) versions of OpenSSL. However, to make CA certificate roll-over easier, it's recommended to use the value **no**, especially if combined with the **−selfsign** command line option.

serial
a text file containing the next serial number to use in hex. Mandatory. This file must be present and contain a valid serial number.

crlnumber
a text file containing the next CRL number to use in hex. The crl number will be inserted in the CRLs only if this file exists. If this file is present, it must contain a valid CRL number.

x509_extensions
the same as **−extensions**.

crl_extensions
the same as **−crlexts**.

preserve
the same as **−preserveDN**

email_in_dn
the same as **−noemailDN**. If you want the EMAIL field to be removed from the DN of the certificate simply set this to 'no'. If not present the default is to allow for the EMAIL filed in the certificate's DN.

msie_hack
the same as **−msie_hack**

policy
the same as **−policy**. Mandatory. See the **POLICY FORMAT** section for more information.

name_opt, **cert_opt**
these options allow the format used to display the certificate details when asking the user to confirm signing. All the options supported by the **x509** utilities **−nameopt** and **−certopt** switches can be used here, except the **no_signame** and **no_sigdump** are permanently set and cannot be disabled (this is because the certificate signature cannot be displayed because the certificate has not been signed at this point).

For convenience the values **ca_default** are accepted by both to produce a reasonable output.

If neither option is present the format used in earlier versions of OpenSSL is used. Use of the old format is **strongly** discouraged because it only displays fields mentioned in the **policy** section, mishandles multicharacter string types and does not display extensions.

copy_extensions
determines how extensions in certificate requests should be handled. If set to **none** or this option is not present then extensions are ignored and not copied to the certificate. If set to **copy** then any extensions present in the request that are not already present are copied to the certificate. If set to **copyall** then all extensions in the request are copied to the certificate: if the extension is already present in the certificate it is deleted first. See the **WARNINGS** section before using this option.

The main use of this option is to allow a certificate request to supply values for certain extensions such as subjectAltName.

POLICY FORMAT

The policy section consists of a set of variables corresponding to certificate DN fields. If the value is "match" then the field value must match the same field in the CA certificate. If the value is "supplied" then

it must be present. If the value is ''optional'' then it may be present. Any fields not mentioned in the policy section are silently deleted, unless the **−preserveDN** option is set but this can be regarded more of a quirk than intended behaviour.

SPKAC FORMAT

The input to the **−spkac** command line option is a Netscape signed public key and challenge. This will usually come from the **KEYGEN** tag in an HTML form to create a new private key. It is however possible to create SPKACs using the **spkac** utility.

The file should contain the variable SPKAC set to the value of the SPKAC and also the required DN components as name value pairs. If you need to include the same component twice then it can be preceded by a number and a '.'.

EXAMPLES

Note: these examples assume that the **ca** directory structure is already set up and the relevant files already exist. This usually involves creating a CA certificate and private key with **req**, a serial number file and an empty index file and placing them in the relevant directories.

To use the sample configuration file below the directories demoCA, demoCA/private and demoCA/newcerts would be created. The CA certificate would be copied to demoCA/cacert.pem and its private key to demoCA/private/cakey.pem. A file demoCA/serial would be created containing for example ''01'' and the empty index file demoCA/index.txt.

Sign a certificate request:

```
openssl ca -in req.pem -out newcert.pem
```

Sign a certificate request, using CA extensions:

```
openssl ca -in req.pem -extensions v3_ca -out newcert.pem
```

Generate a CRL

```
openssl ca -gencrl -out crl.pem
```

Sign several requests:

```
openssl ca -infiles req1.pem req2.pem req3.pem
```

Certify a Netscape SPKAC:

```
openssl ca -spkac spkac.txt
```

A sample SPKAC file (the SPKAC line has been truncated for clarity):

```
SPKAC=MIG0MGAwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAn7PDhCeV/xIxUg8V70YRxK2A5
CN=Steve Test
emailAddress=steve@openssl.org
0.OU=OpenSSL Group
1.OU=Another Group
```

A sample configuration file with the relevant sections for **ca**:

```
[ ca ]
default_ca      = CA_default            # The default ca section

[ CA_default ]

dir             = ./demoCA              # top dir
database        = $dir/index.txt        # index file.
new_certs_dir   = $dir/newcerts         # new certs dir

certificate     = $dir/cacert.pem       # The CA cert
serial          = $dir/serial           # serial no file
private_key     = $dir/private/cakey.pem# CA private key
RANDFILE        = $dir/private/.rand    # random number file
```

```
default_days   = 365                    # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md     = md5                    # md to use

policy         = policy_any             # default policy
email_in_dn    = no                     # Don't add the email into cert DN

name_opt       = ca_default             # Subject name display option
cert_opt       = ca_default             # Certificate display option
copy_extensions = none                  # Don't copy extensions from request

[ policy_any ]
countryName            = supplied
stateOrProvinceName    = optional
organizationName       = optional
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional
```

FILES

Note: the location of all files can change either by compile time options, configuration file entries, environment variables or command line options. The values below reflect the default values.

```
/etc/openssh/openssh.cnf        - master configuration file
./demoCA                        - main CA directory
./demoCA/cacert.pem             - CA certificate
./demoCA/private/cakey.pem      - CA private key
./demoCA/serial                 - CA serial number file
./demoCA/serial.old             - CA serial number backup file
./demoCA/index.txt              - CA text database file
./demoCA/index.txt.old          - CA text database backup file
./demoCA/certs                  - certificate output file
./demoCA/.rnd                   - CA random seed information
```

ENVIRONMENT VARIABLES

OPENSSL_CONF reflects the location of master configuration file it can be overridden by the **−config** command line option.

RESTRICTIONS

The text database index file is a critical part of the process and if corrupted it can be difficult to fix. It is theoretically possible to rebuild the index file from all the issued certificates and a current CRL: however there is no option to do this.

V2 CRL features like delta CRLs are not currently supported.

Although several requests can be input and handled at once it is only possible to include one SPKAC or self signed certificate.

BUGS

The use of an in memory text database can cause problems when large numbers of certificates are present because, as the name implies the database has to be kept in memory.

The **ca** command really needs rewriting or the required functionality exposed at either a command or interface level so a more friendly utility (perl script or GUI) can handle things properly. The scripts **CA.sh** and **CA.pl** help a little but not very much.

Any fields in a request that are not present in a policy are silently deleted. This does not happen if the **−preserveDN** option is used. To enforce the absence of the EMAIL field within the DN, as suggested by RFCs, regardless the contents of the request' subject the **−noemailDN** option can be used. The behaviour should be more friendly and configurable.

Cancelling some commands by refusing to certify a certificate can create an empty file.

WARNINGS

The **ca** command is quirky and at times downright unfriendly.

The **ca** utility was originally meant as an example of how to do things in a CA. It was not supposed to be used as a full blown CA itself: nevertheless some people are using it for this purpose.

The **ca** command is effectively a single user command: no locking is done on the various files and attempts to run more than one **ca** command on the same database can have unpredictable results.

The **copy_extensions** option should be used with caution. If care is not taken then it can be a security risk. For example if a certificate request contains a basicConstraints extension with CA:TRUE and the **copy_extensions** value is set to **copyall** and the user does not spot this when the certificate is displayed then this will hand the requestor a valid CA certificate.

This situation can be avoided by setting **copy_extensions** to **copy** and including basicConstraints with CA:FALSE in the configuration file. Then if the request contains a basicConstraints extension it will be ignored.

It is advisable to also include values for other extensions such as **keyUsage** to prevent a request supplying its own values.

Additional restrictions can be placed on the CA certificate itself. For example if the CA certificate has:

```
basicConstraints = CA:TRUE, pathlen:0
```

then even if a certificate is issued with CA:TRUE it will not be valid.

SEE ALSO

openssl_req (1), *openssl_spkac* (1), *openssl_x509* (1), *CA.pl* (1), *openssl.cnf* (5), *x509v3_config* (5)

NAME

ciphers − SSL cipher display and cipher list tool.

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl ciphers [**−v**] [**−V**] [**−ssl2**] [**−ssl3**] [**−tls1**] [**cipherlist**]

DESCRIPTION

The **ciphers** command converts textual OpenSSL cipher lists into ordered SSL cipher preference lists. It can be used as a test tool to determine the appropriate cipherlist.

COMMAND OPTIONS

−v Verbose option. List ciphers with a complete description of protocol version (SSLv2 or SSLv3; the latter includes TLS), key exchange, authentication, encryption and mac algorithms used along with any key size restrictions and whether the algorithm is classed as an ‘‘export’’ cipher. Note that without the **−v** option, ciphers may seem to appear twice in a cipher list; this is when similar ciphers are available for SSL v2 and for SSL v3/TLS v1.

−V Like **−V**, but include cipher suite codes in output (hex format).

−ssl3
only include SSL v3 ciphers.

−ssl2
only include SSL v2 ciphers.

−tls1
only include TLS v1 ciphers.

−h, **−?**
print a brief usage message.

cipherlist
a cipher list to convert to a cipher preference list. If it is not included then the default cipher list will be used. The format is described below.

CIPHER LIST FORMAT

The cipher list consists of one or more *cipher strings* separated by colons. Commas or spaces are also acceptable separators but colons are normally used.

The actual cipher string can take several different forms.

It can consist of a single cipher suite such as **RC4−SHA**.

It can represent a list of cipher suites containing a certain algorithm, or cipher suites of a certain type. For example **SHA1** represents all ciphers suites using the digest algorithm SHA1 and **SSLv3** represents all SSL v3 algorithms.

Lists of cipher suites can be combined in a single cipher string using the **+** character. This is used as a logical **and** operation. For example **SHA1+DES** represents all cipher suites containing the SHA1 **and** the DES algorithms.

Each cipher string can be optionally preceded by the characters **!**, **−** or **+**.

If **!** is used then the ciphers are permanently deleted from the list. The ciphers deleted can never reappear in the list even if they are explicitly stated.

If **−** is used then the ciphers are deleted from the list, but some or all of the ciphers can be added again by later options.

If **+** is used then the ciphers are moved to the end of the list. This option doesn’t add any new ciphers it just moves matching existing ones.

If none of these characters is present then the string is just interpreted as a list of ciphers to be appended to

the current preference list. If the list includes any ciphers already present they will be ignored: that is they will not moved to the end of the list.

Additionally the cipher string **@STRENGTH** can be used at any point to sort the current cipher list in order of encryption algorithm key length.

CIPHER STRINGS

The following is a list of all permitted cipher strings and their meanings.

DEFAULT

the default cipher list. This is determined at compile time and, as of OpenSSL 0.9.9, is normally **ALL:!aNULL:!eNULL**. This must be the first cipher string specified.

COMPLEMENTOFDEFAULT

the ciphers included in **ALL**, but not enabled by default. Currently this is **ADH**. Note that this rule does not cover **eNULL**, which is not included by **ALL** (use **COMPLEMENTOFALL** if necessary).

ALL

all cipher suites except the **eNULL** ciphers which must be explicitly enabled; as of OpenSSL, the **ALL** cipher suites are reasonably ordered by default

COMPLEMENTOFALL

the cipher suites not enabled by **ALL**, currently being **eNULL**.

HIGH

''high'' encryption cipher suites. This currently means those with key lengths larger than 128 bits, and some cipher suites with 128−bit keys.

MEDIUM

''medium'' encryption cipher suites, currently some of those using 128 bit encryption.

LOW

''low'' encryption cipher suites, currently those using 64 or 56 bit encryption algorithms but excluding export cipher suites.

EXP, **EXPORT**

export encryption algorithms. Including 40 and 56 bits algorithms.

EXPORT40

40 bit export encryption algorithms

EXPORT56

56 bit export encryption algorithms. In OpenSSL 0.9.8c and later the set of 56 bit export ciphers is empty unless OpenSSL has been explicitly configured with support for experimental ciphers.

eNULL, **NULL**

the ''NULL'' ciphers that is those offering no encryption. Because these offer no encryption at all and are a security risk they are disabled unless explicitly included.

aNULL

the cipher suites offering no authentication. This is currently the anonymous DH algorithms. These cipher suites are vulnerable to a ''man in the middle'' attack and so their use is normally discouraged.

kRSA, **RSA**

cipher suites using RSA key exchange.

kEDH

cipher suites using ephemeral DH key agreement.

kDHr, **kDHd**

cipher suites using DH key agreement and DH certificates signed by CAs with RSA and DSS keys respectively. Not implemented.

aRSA
cipher suites using RSA authentication, i.e. the certificates carry RSA keys.

aDSS, **DSS**
cipher suites using DSS authentication, i.e. the certificates carry DSS keys.

aDH
cipher suites effectively using DH authentication, i.e. the certificates carry DH keys. Not implemented.

kFZA, **aFZA**, **eFZA**, **FZA**
ciphers suites using FORTEZZA key exchange, authentication, encryption or all FORTEZZA algorithms. Not implemented.

TLSv1, **SSLv3**, **SSLv2**
TLS v1.0, SSL v3.0 or SSL v2.0 cipher suites respectively.

DH cipher suites using DH, including anonymous DH.

ADH
anonymous DH cipher suites.

AES
cipher suites using AES.

CAMELLIA
cipher suites using Camellia.

3DES
cipher suites using triple DES.

DES
cipher suites using DES (not triple DES).

RC4
cipher suites using RC4.

RC2
cipher suites using RC2.

IDEA
cipher suites using IDEA.

SEED
cipher suites using SEED.

MD5
cipher suites using MD5.

SHA1, **SHA**
cipher suites using SHA1.

CIPHER SUITE NAMES

The following lists give the SSL or TLS cipher suites names from the relevant specification and their OpenSSL equivalents. It should be noted, that several cipher suite names do not include the authentication used, e.g. DES-CBC3-SHA. In these cases, RSA authentication is used.

SSL v3.0 cipher suites.

SSL_RSA_WITH_NULL_MD5	NULL-MD5
SSL_RSA_WITH_NULL_SHA	NULL-SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
SSL_RSA_WITH_RC4_128_MD5	RC4-MD5
SSL_RSA_WITH_RC4_128_SHA	RC4-SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	EXP-RC2-CBC-MD5
SSL_RSA_WITH_IDEA_CBC_SHA	IDEA-CBC-SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DES-CBC-SHA
SSL_RSA_WITH_DES_CBC_SHA	DES-CBC-SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA
SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	Not implemented.
SSL_DH_DSS_WITH_DES_CBC_SHA	Not implemented.
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	Not implemented.
SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	Not implemented.
SSL_DH_RSA_WITH_DES_CBC_SHA	Not implemented.
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	Not implemented.
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-DSS-DES-CBC-SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA	EDH-DSS-CBC-SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	EDH-DSS-DES-CBC3-SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-RSA-DES-CBC-SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA	EDH-RSA-DES-CBC-SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	EDH-RSA-DES-CBC3-SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	EXP-ADH-RC4-MD5
SSL_DH_anon_WITH_RC4_128_MD5	ADH-RC4-MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	EXP-ADH-DES-CBC-SHA
SSL_DH_anon_WITH_DES_CBC_SHA	ADH-DES-CBC-SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	ADH-DES-CBC3-SHA
SSL_FORTEZZA_KEA_WITH_NULL_SHA	Not implemented.
SSL_FORTEZZA_KEA_WITH_FORTEZZA_CBC_SHA	Not implemented.
SSL_FORTEZZA_KEA_WITH_RC4_128_SHA	Not implemented.

TLS v1.0 cipher suites.

TLS_RSA_WITH_NULL_MD5	NULL-MD5
TLS_RSA_WITH_NULL_SHA	NULL-SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
TLS_RSA_WITH_RC4_128_MD5	RC4-MD5
TLS_RSA_WITH_RC4_128_SHA	RC4-SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5	EXP-RC2-CBC-MD5
TLS_RSA_WITH_IDEA_CBC_SHA	IDEA-CBC-SHA
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DES-CBC-SHA
TLS_RSA_WITH_DES_CBC_SHA	DES-CBC-SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	DES-CBC3-SHA

TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA	Not implemented.
TLS_DH_DSS_WITH_DES_CBC_SHA	Not implemented.
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	Not implemented.
TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_DES_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	Not implemented.
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-DSS-DES-CBC-SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA	EDH-DSS-CBC-SHA
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	EDH-DSS-DES-CBC3-SHA
TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-RSA-DES-CBC-SHA
TLS_DHE_RSA_WITH_DES_CBC_SHA	EDH-RSA-DES-CBC-SHA
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	EDH-RSA-DES-CBC3-SHA
TLS_DH_anon_EXPORT_WITH_RC4_40_MD5	EXP-ADH-RC4-MD5
TLS_DH_anon_WITH_RC4_128_MD5	ADH-RC4-MD5
TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA	EXP-ADH-DES-CBC-SHA
TLS_DH_anon_WITH_DES_CBC_SHA	ADH-DES-CBC-SHA
TLS_DH_anon_WITH_3DES_EDE_CBC_SHA	ADH-DES-CBC3-SHA

AES ciphersuites from RFC3268, extending TLS v1.0

TLS_RSA_WITH_AES_128_CBC_SHA	AES128-SHA
TLS_RSA_WITH_AES_256_CBC_SHA	AES256-SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA	Not implemented.
TLS_DH_DSS_WITH_AES_256_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_AES_128_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_AES_256_CBC_SHA	Not implemented.
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	DHE-DSS-AES128-SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	DHE-DSS-AES256-SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	DHE-RSA-AES128-SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	DHE-RSA-AES256-SHA
TLS_DH_anon_WITH_AES_128_CBC_SHA	ADH-AES128-SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA	ADH-AES256-SHA

Camellia ciphersuites from RFC4132, extending TLS v1.0

TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	CAMELLIA128-SHA
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	CAMELLIA256-SHA
TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA	Not implemented.
TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA	Not implemented.
TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA	Not implemented.
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA	DHE-DSS-CAMELLIA128-SHA
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA	DHE-DSS-CAMELLIA256-SHA
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	DHE-RSA-CAMELLIA128-SHA
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	DHE-RSA-CAMELLIA256-SHA
TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA	ADH-CAMELLIA128-SHA
TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA	ADH-CAMELLIA256-SHA

SEED ciphersuites from RFC4162, extending TLS v1.0

TLS_RSA_WITH_SEED_CBC_SHA	SEED-SHA

```
TLS_DH_DSS_WITH_SEED_CBC_SHA          Not implemented.
TLS_DH_RSA_WITH_SEED_CBC_SHA          Not implemented.

TLS_DHE_DSS_WITH_SEED_CBC_SHA         DHE-DSS-SEED-SHA
TLS_DHE_RSA_WITH_SEED_CBC_SHA         DHE-RSA-SEED-SHA

TLS_DH_anon_WITH_SEED_CBC_SHA         ADH-SEED-SHA
```

Additional Export 1024 and other cipher suites

Note: these ciphers can also be used in SSL v3.

```
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA     EXP1024-DES-CBC-SHA
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA      EXP1024-RC4-SHA
TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA EXP1024-DHE-DSS-DES-CBC-SHA
TLS_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA  EXP1024-DHE-DSS-RC4-SHA
TLS_DHE_DSS_WITH_RC4_128_SHA            DHE-DSS-RC4-SHA
```

SSL v2.0 cipher suites.

```
SSL_CK_RC4_128_WITH_MD5                 RC4-MD5
SSL_CK_RC4_128_EXPORT40_WITH_MD5        EXP-RC4-MD5
SSL_CK_RC2_128_CBC_WITH_MD5             RC2-MD5
SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5    EXP-RC2-MD5
SSL_CK_IDEA_128_CBC_WITH_MD5            IDEA-CBC-MD5
SSL_CK_DES_64_CBC_WITH_MD5              DES-CBC-MD5
SSL_CK_DES_192_EDE3_CBC_WITH_MD5        DES-CBC3-MD5
```

NOTES

The non-ephemeral DH modes are currently unimplemented in OpenSSL because there is no support for DH certificates.

Some compiled versions of OpenSSL may not include all the ciphers listed here because some ciphers were excluded at compile time.

EXAMPLES

Verbose listing of all OpenSSL ciphers including NULL ciphers:

```
openssl ciphers -v 'ALL:eNULL'
```

Include all ciphers except NULL and anonymous DH then sort by strength:

```
openssl ciphers -v 'ALL:!ADH:@STRENGTH'
```

Include only 3DES ciphers and then place RSA ciphers last:

```
openssl ciphers -v '3DES:+RSA'
```

Include all RC4 ciphers but leave out those without authentication:

```
openssl ciphers -v 'RC4:!COMPLEMENTOFDEFAULT'
```

Include all chiphers with RSA authentication but leave out ciphers without encryption.

```
openssl ciphers -v 'RSA:!COMPLEMENTOFALL'
```

SEE ALSO

openssl_s_client (1), *openssl_s_server* (1), *ssl* (3)

HISTORY

The **COMPLENTOFALL** and **COMPLEMENTOFDEFAULT** selection options for cipherlist strings were added in OpenSSL 0.9.7. The **−V** option for the **ciphers** command was added in OpenSSL 0.9.9.

NAME

crl − CRL utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl crl [**−inform PEM|DER**] [**−outform PEM|DER**] [**−text**] [**−in filename**] [**−out filename**] [**−noout**] [**−hash**] [**−issuer**] [**−lastupdate**] [**−nextupdate**] [**−CAfile file**] [**−CApath dir**]

DESCRIPTION

The **crl** command processes CRL files in DER or PEM format.

COMMAND OPTIONS

−inform DER|PEM

This specifies the input format. **DER** format is DER encoded CRL structure. **PEM** (the default) is a base64 encoded version of the DER form with header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read from or standard input if this option is not specified.

−out filename

specifies the output filename to write to or standard output by default.

−text

print out the CRL in text form.

−noout

don’t output the encoded version of the CRL.

−hash

output a hash of the issuer name. This can be use to lookup CRLs in a directory by issuer name.

−issuer

output the issuer name.

−lastupdate

output the lastUpdate field.

−nextupdate

output the nextUpdate field.

−CAfile file

verify the signature on a CRL by looking up the issuing certificate in **file**

−CApath dir

verify the signature on a CRL by looking up the issuing certificate in **dir**. This directory must be a standard certificate directory: that is a hash of each subject name (using **x509 −hash**) should be linked to each certificate.

NOTES

The PEM CRL format uses the header and footer lines:

```
-----BEGIN X509 CRL-----
-----END X509 CRL-----
```

EXAMPLES

Convert a CRL file from PEM to DER:

```
openssl crl -in crl.pem -outform DER -out crl.der
```

Output the text form of a DER encoded certificate:

```
openssl crl -in crl.der -text -noout
```

BUGS

Ideally it should be possible to create a CRL using appropriate options and files too.

SEE ALSO

openssl_crl2pkcs7 (1), *openssl_ca* (1), *openssl_x509* (1)

NAME

crl2pkcs7 − Create a PKCS#7 structure from a CRL and certificates.

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl crl2pkcs7 [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−out filename**] [**−certfile filename**] [**−nocrl**]

DESCRIPTION

The **crl2pkcs7** command takes an optional CRL and one or more certificates and converts them into a PKCS#7 degenerate ''certificates only'' structure.

COMMAND OPTIONS

−inform DER|PEM

This specifies the CRL input format. **DER** format is DER encoded CRL structure.**PEM** (the default) is a base64 encoded version of the DER form with header and footer lines.

−outform DER|PEM

This specifies the PKCS#7 structure output format. **DER** format is DER encoded PKCS#7 structure.**PEM** (the default) is a base64 encoded version of the DER form with header and footer lines.

−in filename

This specifies the input filename to read a CRL from or standard input if this option is not specified.

−out filename

specifies the output filename to write the PKCS#7 structure to or standard output by default.

−certfile filename

specifies a filename containing one or more certificates in **PEM** format. All certificates in the file will be added to the PKCS#7 structure. This option can be used more than once to read certificates form multiple files.

−nocrl

normally a CRL is included in the output file. With this option no CRL is included in the output file and a CRL is not read from the input file.

EXAMPLES

Create a PKCS#7 structure from a certificate and CRL:

```
openssl crl2pkcs7 -in crl.pem -certfile cert.pem -out p7.pem
```

Creates a PKCS#7 structure in DER format with no CRL from several different certificates:

```
openssl crl2pkcs7 -nocrl -certfile newcert.pem
       -certfile demoCA/cacert.pem -outform DER -out p7.der
```

NOTES

The output file is a PKCS#7 signed data structure containing no signers and just certificates and an optional CRL.

This utility can be used to send certificates and CAs to Netscape as part of the certificate enrollment process. This involves sending the DER encoded output as MIME type application/x−x509−user−cert.

The **PEM** encoded form with the header and footer lines removed can be used to install user certificates and CAs in MSIE using the Xenroll control.

SEE ALSO

openssl_pkcs7 (1)

NAME

dgst, md5, md4, md2, sha1, sha, mdc2, ripemd160 − message digests

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl dgst [**−md5**|**−md4**|**−md2**|**−sha1**|**−sha**|**−mdc2**|**−ripemd160**|**−dss1**] [**−c**] [**−d**] [**−hex**] [**−binary**] [**−out filename**] [**−sign filename**] [**−passin arg**] [**−verify filename**] [**−prverify filename**] [**−signature filename**] [**file...**]

[**md5**|**md4**|**md2**|**sha1**|**sha**|**mdc2**|**ripemd160**] [**−c**] [**−d**] [**file...**]

DESCRIPTION

The digest functions output the message digest of a supplied file or files in hexadecimal form. They can also be used for digital signing and verification.

OPTIONS

−c print out the digest in two digit groups separated by colons, only relevant if **hex** format output is used.

−d print out BIO debugging information.

−hex
digest is to be output as a hex dump. This is the default case for a ‘‘normal’’ digest as opposed to a digital signature.

−binary
output the digest or signature in binary form.

−out filename
filename to output to, or standard output by default.

−sign filename
digitally sign the digest using the private key in ‘‘filename’’.

−passin arg
the private key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−verify filename
verify the signature using the the public key in ‘‘filename’’. The output is either ‘‘Verification OK’’ or ‘‘Verification Failure’’.

−prverify filename
verify the signature using the the private key in ‘‘filename’’.

−signature filename
the actual signature to verify.

−rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd* (3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

file...
file or files to digest. If no files are specified then standard input is used.

NOTES

The digest of choice for all new applications is SHA1. Other digests are however still widely used.

If you wish to sign or verify data using the DSA algorithm then the dss1 digest must be used.

A source of random numbers is required for certain signing algorithms, in particular DSA.

The signing and verify options should only be used if a single file is being signed or verified.

NAME

dhparam − DH parameter manipulation and generation

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl dhparam [**−inform DER|PEM**] [**−outform DER|PEM**] [**−in** *filename*] [**−out** *filename*] [**−dsaparam**] [**−noout**] [**−text**] [**−C**] [**−2**] [**−5**] [**−rand** *file(s)*] [**−engine id**] [*numbits*]

DESCRIPTION

This command is used to manipulate DH parameter files.

OPTIONS

−inform DER|PEM

This specifies the input format. The **DER** option uses an ASN1 DER encoded form compatible with the PKCS#3 DHparameter structure. The PEM form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in *filename*

This specifies the input filename to read parameters from or standard input if this option is not specified.

−out *filename*

This specifies the output filename parameters to. Standard output is used if this option is not present. The output filename should **not** be the same as the input filename.

−dsaparam

If this option is used, DSA rather than DH parameters are read or created; they are converted to DH format. Otherwise, ''strong'' primes (such that (p−1)/2 is also prime) will be used for DH parameter generation.

DH parameter generation with the **−dsaparam** option is much faster, and the recommended exponent length is shorter, which makes DH key exchange more efficient. Beware that with such DSA-style DH parameters, a fresh DH key should be created for each use to avoid small-subgroup attacks that may be possible otherwise.

−2, **−5**

The generator to use, either 2 or 5. 2 is the default. If present then the input file is ignored and parameters are generated instead.

−rand *file(s)*

a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd*(3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

numbits

this option specifies that a parameter set should be generated of size *numbits*. It must be the last option. If not present then a value of 512 is used. If this option is present then the input file is ignored and parameters are generated instead.

−noout

this option inhibits the output of the encoded version of the parameters.

−text

this option prints out the DH parameters in human readable form.

−C this option converts the parameters into C code. The parameters can then be loaded by calling the **get_dh***numbits***()** function.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

WARNINGS

The program **dhparam** combines the functionality of the programs **dh** and **gendh** in previous versions of OpenSSL and SSLeay. The **dh** and **gendh** programs are retained for now but may have different purposes in future versions of OpenSSL.

NOTES

PEM format DH parameters use the header and footer lines:

```
-----BEGIN DH PARAMETERS-----
-----END DH PARAMETERS-----
```

OpenSSL currently only supports the older PKCS#3 DH, not the newer X9.42 DH.

This program manipulates DH parameters not keys.

BUGS

There should be a way to generate and manipulate DH keys.

SEE ALSO

openssl_dsaparam (1)

HISTORY

The **dhparam** command was added in OpenSSL 0.9.5. The **−dsaparam** option was added in OpenSSL 0.9.6.

NAME

dsa − DSA key processing

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl dsa [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−passin arg**] [**−out filename**] [**−passout arg**] [**−des**] [**−des3**] [**−idea**] [**−text**] [**−noout**] [**−modulus**] [**−pubin**] [**−pubout**] [**−engine id**]

DESCRIPTION

The **dsa** command processes DSA keys. They can be converted between various forms and their components printed out. **Note** This command uses the traditional SSLeay compatible format for private key encryption: newer applications should use the more secure PKCS#8 format using the **pkcs8**

COMMAND OPTIONS

−inform DER|PEM

This specifies the input format. The **DER** option with a private key uses an ASN1 DER encoded form of an ASN.1 SEQUENCE consisting of the values of version (currently zero), p, q, g, the public and private key components respectively as ASN.1 INTEGERs. When used with a public key it uses a SubjectPublicKeyInfo structure: it is an error if the key is not DSA.

The **PEM** form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines. In the case of a private key PKCS#8 format is also accepted.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read a key from or standard input if this option is not specified. If the key is encrypted a pass phrase will be prompted for.

−passin arg

the input file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−out filename

This specifies the output filename to write a key to or standard output by is not specified. If any encryption options are set then a pass phrase will be prompted for. The output filename should **not** be the same as the input filename.

−passout arg

the output file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−des|−des3|−idea

These options encrypt the private key with the DES, triple DES, or the IDEA ciphers respectively before outputting it. A pass phrase is prompted for. If none of these options is specified the key is written in plain text. This means that using the **dsa** utility to read in an encrypted key with no encryption option can be used to remove the pass phrase from a key, or by setting the encryption options it can be use to add or change the pass phrase. These options can only be used with PEM format output files.

−text

prints out the public, private key components and parameters.

−noout

this option prevents output of the encoded version of the key.

−modulus

this option prints out the value of the public key component of the key.

−pubin
by default a private key is read from the input file: with this option a public key is read instead.

−pubout
by default a private key is output. With this option a public key will be output instead. This option is automatically set if the input is a public key.

−engine id
specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

NOTES

The PEM private key format uses the header and footer lines:

```
-----BEGIN DSA PRIVATE KEY-----
-----END DSA PRIVATE KEY-----
```

The PEM public key format uses the header and footer lines:

```
-----BEGIN PUBLIC KEY-----
-----END PUBLIC KEY-----
```

EXAMPLES

To remove the pass phrase on a DSA private key:

```
openssl dsa -in key.pem -out keyout.pem
```

To encrypt a private key using triple DES:

```
openssl dsa -in key.pem -des3 -out keyout.pem
```

To convert a private key from PEM to DER format:

```
openssl dsa -in key.pem -outform DER -out keyout.der
```

To print out the components of a private key to standard output:

```
openssl dsa -in key.pem -text -noout
```

To just output the public part of a private key:

```
openssl dsa -in key.pem -pubout -out pubkey.pem
```

SEE ALSO

openssl_dsaparam (1), *openssl_gendsa* (1), *openssl_rsa* (1), *openssl_genrsa* (1)

NAME

dsaparam − DSA parameter manipulation and generation

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl dsaparam [**−inform DER|PEM**] [**−outform DER|PEM**] [**−in filename**] [**−out filename**] [**−noout**] [**−text**] [**−C**] [**−rand file(s)**] [**−genkey**] [**−engine id**] [**numbits**]

DESCRIPTION

This command is used to manipulate or generate DSA parameter files.

OPTIONS

−inform DER|PEM

This specifies the input format. The **DER** option uses an ASN1 DER encoded form compatible with RFC2459 (PKIX) DSS-Parms that is a SEQUENCE consisting of p, q and g respectively. The PEM form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read parameters from or standard input if this option is not specified. If the **numbits** parameter is included then this option will be ignored.

−out filename

This specifies the output filename parameters to. Standard output is used if this option is not present. The output filename should **not** be the same as the input filename.

−noout

this option inhibits the output of the encoded version of the parameters.

−text

this option prints out the DSA parameters in human readable form.

−C this option converts the parameters into C code. The parameters can then be loaded by calling the ***get_dsaXXX()*** function.

−genkey

this option will generate a DSA either using the specified or generated parameters.

−rand file(s)

a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd*(3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

numbits

this option specifies that a parameter set should be generated of size **numbits**. It must be the last option. If this option is included then the input file (if any) is ignored.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

NOTES

PEM format DSA parameters use the header and footer lines:

```
-----BEGIN DSA PARAMETERS-----
-----END DSA PARAMETERS-----
```

DSA parameter generation is a slow process and as a result the same set of DSA parameters is often used to generate several distinct keys.

SEE ALSO

openssl_gendsa (1), *openssl_dsa* (1), *openssl_genrsa* (1), *openssl_rsa* (1)

NAME

ec − EC key processing

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl ec [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−passin arg**] [**−out filename**] [**−passout arg**] [**−des**] [**−des3**] [**−idea**] [**−text**] [**−noout**] [**−param_out**] [**−pubin**] [**−pubout**] [**−conv_form arg**] [**−param_enc arg**] [**−engine id**]

DESCRIPTION

The **ec** command processes EC keys. They can be converted between various forms and their components printed out. **Note** OpenSSL uses the private key format specified in 'SEC 1: Elliptic Curve Cryptography' (http://www.secg.org/). To convert a OpenSSL EC private key into the PKCS#8 private key format use the **pkcs8** command.

COMMAND OPTIONS

−inform DER|PEM

This specifies the input format. The **DER** option with a private key uses an ASN.1 DER encoded SEC1 private key. When used with a public key it uses the SubjectPublicKeyInfo structur as specified in RFC 3280. The **PEM** form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines. In the case of a private key PKCS#8 format is also accepted.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read a key from or standard input if this option is not specified. If the key is encrypted a pass phrase will be prompted for.

−passin arg

the input file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−out filename

This specifies the output filename to write a key to or standard output by is not specified. If any encryption options are set then a pass phrase will be prompted for. The output filename should **not** be the same as the input filename.

−passout arg

the output file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−des|−des3|−idea

These options encrypt the private key with the DES, triple DES, IDEA or any other cipher supported by OpenSSL before outputting it. A pass phrase is prompted for. If none of these options is specified the key is written in plain text. This means that using the **ec** utility to read in an encrypted key with no encryption option can be used to remove the pass phrase from a key, or by setting the encryption options it can be use to add or change the pass phrase. These options can only be used with PEM format output files.

−text

prints out the public, private key components and parameters.

−noout

this option prevents output of the encoded version of the key.

−modulus

this option prints out the value of the public key component of the key.

−pubin
by default a private key is read from the input file: with this option a public key is read instead.

−pubout
by default a private key is output. With this option a public key will be output instead. This option is automatically set if the input is a public key.

−conv_form
This specifies how the points on the elliptic curve are converted into octet strings. Possible values are: **compressed** (the default value), **uncompressed** and **hybrid**. For more information regarding the point conversion forms please read the X9.62 standard. **Note** Due to patent issues the **compressed** option is disabled by default for binary curves and can be enabled by defining the preprocessor macro **OPENSSL_EC_BIN_PT_COMP** at compile time.

−param_enc arg
This specifies how the elliptic curve parameters are encoded. Possible value are: **named_curve**, i.e. the ec parameters are specified by a OID, or **explicit** where the ec parameters are explicitly given (see RFC 3279 for the definition of the EC parameters structures). The default value is **named_curve**. **Note** the **implicitlyCA** alternative ,as specified in RFC 3279, is currently not implemented in OpenSSL.

−engine id
specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

NOTES

The PEM private key format uses the header and footer lines:

```
-----BEGIN EC PRIVATE KEY-----
-----END EC PRIVATE KEY-----
```

The PEM public key format uses the header and footer lines:

```
-----BEGIN PUBLIC KEY-----
-----END PUBLIC KEY-----
```

EXAMPLES

To encrypt a private key using triple DES:

```
openssl ec -in key.pem -des3 -out keyout.pem
```

To convert a private key from PEM to DER format:

```
openssl ec -in key.pem -outform DER -out keyout.der
```

To print out the components of a private key to standard output:

```
openssl ec -in key.pem -text -noout
```

To just output the public part of a private key:

```
openssl ec -in key.pem -pubout -out pubkey.pem
```

To change the parameters encoding to **explicit**:

```
openssl ec -in key.pem -param_enc explicit -out keyout.pem
```

To change the point conversion form to **compressed**:

```
openssl ec -in key.pem -conv_form compressed -out keyout.pem
```

SEE ALSO

ecparam (1), *openssl_dsa* (1), *openssl_rsa* (1)

HISTORY

The ec command was first introduced in OpenSSL 0.9.8.

AUTHOR

Nils Larsch for the OpenSSL project (http://www.openssl.org).

NAME

ecparam − EC parameter manipulation and generation

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl ecparam [**−inform DER|PEM**] [**−outform DER|PEM**] [**−in filename**] [**−out filename**] [**−noout**] [**−text**] [**−C**] [**−check**] [**−name arg**] [**−list_curve**] [**−conv_form arg**] [**−param_enc arg**] [**−no_seed**] [**−rand file(s)**] [**−genkey**] [**−engine id**]

DESCRIPTION

This command is used to manipulate or generate EC parameter files.

OPTIONS

−inform DER|PEM

This specifies the input format. The **DER** option uses an ASN.1 DER encoded form compatible with RFC 3279 EcpkParameters. The PEM form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read parameters from or standard input if this option is not specified.

−out filename

This specifies the output filename parameters to. Standard output is used if this option is not present. The output filename should **not** be the same as the input filename.

−noout

This option inhibits the output of the encoded version of the parameters.

−text

This option prints out the EC parameters in human readable form.

−C This option converts the EC parameters into C code. The parameters can then be loaded by calling the ***get_ec_group_XXX()*** function.

−check

Validate the elliptic curve parameters.

−name arg

Use the EC parameters with the specified 'short' name. Use **−list_curves** to get a list of all currently implemented EC parameters.

−list_curves

If this options is specified **ecparam** will print out a list of all currently implemented EC parameters names and exit.

−conv_form

This specifies how the points on the elliptic curve are converted into octet strings. Possible values are: **compressed** (the default value), **uncompressed** and **hybrid**. For more information regarding the point conversion forms please read the X9.62 standard. **Note** Due to patent issues the **compressed** option is disabled by default for binary curves and can be enabled by defining the preprocessor macro **OPENSSL_EC_BIN_PT_COMP** at compile time.

−param_enc arg

This specifies how the elliptic curve parameters are encoded. Possible value are: **named_curve**, i.e. the ec parameters are specified by a OID, or **explicit** where the ec parameters are explicitly given (see RFC 3279 for the definition of the EC parameters structures). The default value is **named_curve**. **Note** the **implicitlyCA** alternative ,as specified in RFC 3279, is currently not implemented in OpenSSL.

−no_seed

This option inhibits that the 'seed' for the parameter generation is included in the ECParameters structure (see RFC 3279).

−genkey

This option will generate a EC private key using the specified parameters.

−rand file(s)

a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd* (3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

NOTES

PEM format EC parameters use the header and footer lines:

```
-----BEGIN EC PARAMETERS-----
-----END EC PARAMETERS-----
```

OpenSSL is currently not able to generate new groups and therefore **ecparam** can only create EC parameters from known (named) curves.

EXAMPLES

To create EC parameters with the group 'prime192v1':

```
openssl ecparam -out ec_param.pem -name prime192v1
```

To create EC parameters with explicit parameters:

```
openssl ecparam -out ec_param.pem -name prime192v1 -param_enc explicit
```

To validate given EC parameters:

```
openssl ecparam -in ec_param.pem -check
```

To create EC parameters and a private key:

```
openssl ecparam -out ec_key.pem -name prime192v1 -genkey
```

To change the point encoding to 'compressed':

```
openssl ecparam -in ec_in.pem -out ec_out.pem -conv_form compressed
```

To print out the EC parameters to standard output:

```
openssl ecparam -in ec_param.pem -noout -text
```

SEE ALSO

ec (1), *openssl_dsaparam* (1)

HISTORY

The ecparam command was first introduced in OpenSSL 0.9.8.

AUTHOR

Nils Larsch for the OpenSSL project (http://www.openssl.org)

NAME

enc − symmetric cipher routines

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl enc −ciphername [**−in filename**] [**−out filename**] [**−pass arg**] [**−e**] [**−d**] [**−a**] [**−A**] [**−k password**] [**−kfile filename**] [**−K key**] [**−iv IV**] [**−p**] [**−P**] [**−bufsize number**] [**−nopad**] [**−debug**]

DESCRIPTION

The symmetric cipher commands allow data to be encrypted or decrypted using various block and stream ciphers using keys based on passwords or explicitly provided. Base64 encoding or decoding can also be performed either by itself or in addition to the encryption or decryption.

OPTIONS

−in filename
the input filename, standard input by default.

−out filename
the output filename, standard output by default.

−pass arg
the password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−salt
use a salt in the key derivation routines. This option should **ALWAYS** be used unless compatibility with previous versions of OpenSSL or SSLeay is required. This option is only present on OpenSSL versions 0.9.5 or above.

−nosalt
don't use a salt in the key derivation routines. This is the default for compatibility with previous versions of OpenSSL and SSLeay.

−e encrypt the input data: this is the default.

−d decrypt the input data.

−a base64 process the data. This means that if encryption is taking place the data is base64 encoded after encryption. If decryption is set then the input data is base64 decoded before being decrypted.

−A if the **−a** option is set then base64 process the data on one line.

−k password
the password to derive the key from. This is for compatibility with previous versions of OpenSSL. Superseded by the **−pass** argument.

−kfile filename
read the password to derive the key from the first line of **filename**. This is for compatibility with previous versions of OpenSSL. Superseded by the **−pass** argument.

−S salt
the actual salt to use: this must be represented as a string comprised only of hex digits.

−K key
the actual key to use: this must be represented as a string comprised only of hex digits. If only the key is specified, the IV must additionally specified using the **−iv** option. When both a key and a password are specified, the key given with the **−K** option will be used and the IV generated from the password will be taken. It probably does not make much sense to specify both key and password.

−iv IV
the actual IV to use: this must be represented as a string comprised only of hex digits. When only the key is specified using the **−K** option, the IV must explicitly be defined. When a password is being specified using one of the other options, the IV is generated from this password.

−p print out the key and IV used.

−P print out the key and IV used then immediately exit: don't do any encryption or decryption.

−bufsize number
set the buffer size for I/O

−nopad
disable standard block padding

−debug
debug the BIOs used for I/O.

NOTES

The program can be called either as **openssl ciphername** or **openssl enc −ciphername**.

A password will be prompted for to derive the key and IV if necessary.

The **−salt** option should **ALWAYS** be used if the key is being derived from a password unless you want compatibility with previous versions of OpenSSL and SSLeay.

Without the **−salt** option it is possible to perform efficient dictionary attacks on the password and to attack stream cipher encrypted data. The reason for this is that without the salt the same password always generates the same encryption key. When the salt is being used the first eight bytes of the encrypted data are reserved for the salt: it is generated at random when encrypting a file and read from the encrypted file when it is decrypted.

Some of the ciphers do not have large keys and others have security implications if not used correctly. A beginner is advised to just use a strong block cipher in CBC mode such as bf or des3.

All the block ciphers normally use PKCS#5 padding also known as standard block padding: this allows a rudimentary integrity or password check to be performed. However since the chance of random data passing the test is better than 1 in 256 it isn't a very good test.

If padding is disabled then the input data must be a multiple of the cipher block length.

All RC2 ciphers have the same key and effective key length.

Blowfish and RC5 algorithms use a 128 bit key.

SUPPORTED CIPHERS

```
base64              Base 64

bf-cbc              Blowfish in CBC mode
bf                  Alias for bf-cbc
bf-cfb              Blowfish in CFB mode
bf-ecb              Blowfish in ECB mode
bf-ofb              Blowfish in OFB mode

cast-cbc            CAST in CBC mode
cast                Alias for cast-cbc
cast5-cbc           CAST5 in CBC mode
cast5-cfb           CAST5 in CFB mode
cast5-ecb           CAST5 in ECB mode
cast5-ofb           CAST5 in OFB mode

des-cbc             DES in CBC mode
des                 Alias for des-cbc
des-cfb             DES in CBC mode
des-ofb             DES in OFB mode
des-ecb             DES in ECB mode
```

```
des-ede-cbc        Two key triple DES EDE in CBC mode
des-ede            Two key triple DES EDE in ECB mode
des-ede-cfb        Two key triple DES EDE in CFB mode
des-ede-ofb        Two key triple DES EDE in OFB mode

des-ede3-cbc       Three key triple DES EDE in CBC mode
des-ede3           Three key triple DES EDE in ECB mode
des3               Alias for des-ede3-cbc
des-ede3-cfb       Three key triple DES EDE CFB mode
des-ede3-ofb       Three key triple DES EDE in OFB mode

desx               DESX algorithm.

idea-cbc           IDEA algorithm in CBC mode
idea               same as idea-cbc
idea-cfb           IDEA in CFB mode
idea-ecb           IDEA in ECB mode
idea-ofb           IDEA in OFB mode

rc2-cbc            128 bit RC2 in CBC mode
rc2                Alias for rc2-cbc
rc2-cfb            128 bit RC2 in CFB mode
rc2-ecb            128 bit RC2 in ECB mode
rc2-ofb            128 bit RC2 in OFB mode
rc2-64-cbc         64 bit RC2 in CBC mode
rc2-40-cbc         40 bit RC2 in CBC mode

rc4                128 bit RC4
rc4-64             64 bit RC4
rc4-40             40 bit RC4

rc5-cbc            RC5 cipher in CBC mode
rc5                Alias for rc5-cbc
rc5-cfb            RC5 cipher in CFB mode
rc5-ecb            RC5 cipher in ECB mode
rc5-ofb            RC5 cipher in OFB mode

aes-[128|192|256]-cbc  128/192/256 bit AES in CBC mode
aes-[128|192|256]      Alias for aes-[128|192|256]-cbc
aes-[128|192|256]-cfb  128/192/256 bit AES in 128 bit CFB mode
aes-[128|192|256]-cfb1 128/192/256 bit AES in 1 bit CFB mode
aes-[128|192|256]-cfb8 128/192/256 bit AES in 8 bit CFB mode
aes-[128|192|256]-ecb  128/192/256 bit AES in ECB mode
aes-[128|192|256]-ofb  128/192/256 bit AES in OFB mode
```

EXAMPLES

Just base64 encode a binary file:

```
openssl base64 -in file.bin -out file.b64
```

Decode the same file

```
openssl base64 -d -in file.b64 -out file.bin
```

Encrypt a file using triple DES in CBC mode using a prompted password:

```
openssl des3 -salt -in file.txt -out file.des3
```

Decrypt a file using a supplied password:

```
openssl des3 -d -salt -in file.des3 -out file.txt -k mypassword
```

Encrypt a file then base64 encode it (so it can be sent via mail for example) using Blowfish in CBC mode:

```
openssl bf -a -salt -in file.txt -out file.bf
```

Base64 decode a file then decrypt it:

```
openssl bf -d -salt -a -in file.bf -out file.txt
```

Decrypt some data using a supplied 40 bit RC4 key:

```
openssl rc4-40 -in file.rc4 -out file.txt -K 0102030405
```

BUGS

The **−A** option when used with large files doesn't work properly.

There should be an option to allow an iteration count to be included.

The **enc** program only supports a fixed number of algorithms with certain parameters. So if, for example, you want to use RC2 with a 76 bit key or RC4 with an 84 bit key you can't use this program.

NAME

errstr − lookup error codes

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl errstr error_code

DESCRIPTION

Sometimes an application will not load error message and only numerical forms will be available. The **errstr** utility can be used to display the meaning of the hex code. The hex code is the hex digits after the second colon.

EXAMPLE

The error code:

```
27594:error:2006D080:lib(32):func(109):reason(128):bss_file.c:107:
```

can be displayed with:

```
openssl errstr 2006D080
```

to produce the error message:

```
error:2006D080:BIO routines:BIO_new_file:no such file
```

SEE ALSO

openssl_err (3), *ERR_load_crypto_strings* (3), *SSL_load_error_strings* (3)

NAME

gendsa − generate a DSA private key from a set of parameters

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl gendsa [**−out filename**] [**−des**] [**−des3**] [**−idea**] [**−rand file(s)**] [**−engine id**] [**paramfile**]

DESCRIPTION

The **gendsa** command generates a DSA private key from a DSA parameter file (which will be typically generated by the **openssl dsaparam** command).

OPTIONS

−des|−des3|−idea

These options encrypt the private key with the DES, triple DES, or the IDEA ciphers respectively before outputting it. A pass phrase is prompted for. If none of these options is specified no encryption is used.

−rand file(s)

a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd* (3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

paramfile

This option specifies the DSA parameter file to use. The parameters in this file determine the size of the private key. DSA parameters can be generated and examined using the **openssl dsaparam** command.

NOTES

DSA key generation is little more than random number generation so it is much quicker that RSA key generation for example.

SEE ALSO

openssl_dsaparam (1), *openssl_dsa* (1), *openssl_genrsa* (1), *openssl_rsa* (1)

NAME

genrsa − generate an RSA private key

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl genrsa [**−out filename**] [**−passout arg**] [**−des**] [**−des3**] [**−idea**] [**−f4**] [**−3**] [**−rand file(s)**] [**−engine id**] [**numbits**]

DESCRIPTION

The **genrsa** command generates an RSA private key.

OPTIONS

−out filename

the output filename. If this argument is not specified then standard output is used.

−passout arg

the output file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−des|−des3|−idea

These options encrypt the private key with the DES, triple DES, or the IDEA ciphers respectively before outputting it. If none of these options is specified no encryption is used. If encryption is used a pass phrase is prompted for if it is not supplied via the **−passout** argument.

−F4|−3

the public exponent to use, either 65537 or 3. The default is 65537.

−rand file(s)

a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd* (3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

numbits

the size of the private key to generate in bits. This must be the last option specified. The default is 512.

NOTES

RSA private key generation essentially involves the generation of two prime numbers. When generating a private key various symbols will be output to indicate the progress of the generation. A **.** represents each number which has passed an initial sieve test, **+** means a number has passed a single round of the Miller-Rabin primality test. A newline means that the number has passed all the prime tests (the actual number depends on the key size).

Because key generation is a random process the time taken to generate a key may vary somewhat.

BUGS

A quirk of the prime generation algorithm is that it cannot generate small primes. Therefore the number of bits should not be less that 64. For typical private keys this will not matter because for security reasons they will be much larger (typically 1024 bits).

SEE ALSO

openssl_gendsa (1)

NAME

nseq − create or examine a netscape certificate sequence

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl nseq [**−in filename**] [**−out filename**] [**−toseq**]

DESCRIPTION

The **nseq** command takes a file containing a Netscape certificate sequence and prints out the certificates contained in it or takes a file of certificates and converts it into a Netscape certificate sequence.

COMMAND OPTIONS

−in filename

This specifies the input filename to read or standard input if this option is not specified.

−out filename

specifies the output filename or standard output by default.

−toseq

normally a Netscape certificate sequence will be input and the output is the certificates contained in it. With the **−toseq** option the situation is reversed: a Netscape certificate sequence is created from a file of certificates.

EXAMPLES

Output the certificates in a Netscape certificate sequence

```
openssl nseq -in nseq.pem -out certs.pem
```

Create a Netscape certificate sequence

```
openssl nseq -in certs.pem -toseq -out nseq.pem
```

NOTES

The **PEM** encoded form uses the same headers and footers as a certificate:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

A Netscape certificate sequence is a Netscape specific form that can be sent to browsers as an alternative to the standard PKCS#7 format when several certificates are sent to the browser: for example during certificate enrollment. It is used by Netscape certificate server for example.

BUGS

This program needs a few more options: like allowing DER or PEM input and output files and allowing multiple certificate files to be used.

NAME

ocsp − Online Certificate Status Protocol utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl ocsp [**−out file**] [**−issuer file**] [**−cert file**] [**−serial n**] [**−signer file**] [**−signkey file**] [**−sign_other file**] [**−no_certs**] [**−req_text**] [**−resp_text**] [**−text**] [**−reqout file**] [**−respout file**] [**−reqin file**] [**−respin file**] [**−nonce**] [**−no_nonce**] [**−url URL**] [**−host host:n**] [**−path**] [**−CApath dir**] [**−CAfile file**] [**−VAfile file**] [**−validity_period n**] [**−status_age n**] [**−noverify**] [**−verify_other file**] [**−trust_other**] [**−no_intern**] [**−no_signature_verify**] [**−no_cert_verify**] [**−no_chain**] [**−no_cert_checks**] [**−port num**] [**−index file**] [**−CA file**] [**−rsigner file**] [**−rkey file**] [**−rother file**] [**−resp_no_certs**] [**−nmin n**] [**−ndays n**] [**−resp_key_id**] [**−nrequest n**] [**−md5|−sha1|...**]

DESCRIPTION

The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of an identified certificate (RFC 2560).

The **ocsp** command performs many common OCSP tasks. It can be used to print out requests and responses, create requests and send queries to an OCSP responder and behave like a mini OCSP server itself.

OCSP CLIENT OPTIONS

−out filename

specify output filename, default is standard output.

−issuer filename

This specifies the current issuer certificate. This option can be used multiple times. The certificate specified in **filename** must be in PEM format. This option **MUST** come before any **−cert** options.

−cert filename

Add the certificate **filename** to the request. The issuer certificate is taken from the previous **issuer** option, or an error occurs if no issuer certificate is specified.

−serial num

Same as the **cert** option except the certificate with serial number **num** is added to the request. The serial number is interpreted as a decimal integer unless preceded by **0x**. Negative integers can also be specified by preceding the value by a − sign.

−signer filename, **−signkey filename**

Sign the OCSP request using the certificate specified in the **signer** option and the private key specified by the **signkey** option. If the **signkey** option is not present then the private key is read from the same file as the certificate. If neither option is specified then the OCSP request is not signed.

−sign_other filename

Additional certificates to include in the signed request.

−nonce, **−no_nonce**

Add an OCSP nonce extension to a request or disable OCSP nonce addition. Normally if an OCSP request is input using the **respin** option no nonce is added: using the **nonce** option will force addition of a nonce. If an OCSP request is being created (using **cert** and **serial** options) a nonce is automatically added specifying **no_nonce** overrides this.

−req_text, **−resp_text**, **−text**

print out the text form of the OCSP request, response or both respectively.

−reqout file, **−respout file**

write out the DER encoded certificate request or response to **file**.

−reqin file, **−respin file**

read OCSP request or response file from **file**. These option are ignored if OCSP request or response creation is implied by other options (for example with **serial**, **cert** and **host** options).

−url responder_url
specify the responder URL. Both HTTP and HTTPS (SSL/TLS) URLs can be specified.

−host hostname:port, **−path pathname**
if the **host** option is present then the OCSP request is sent to the host **hostname** on port **port**. **path** specifies the HTTP path name to use or ‘‘/’’ by default.

−CAfile file, **−CApath pathname**
file or pathname containing trusted CA certificates. These are used to verify the signature on the OCSP response.

−verify_other file
file containing additional certificates to search when attempting to locate the OCSP response signing certificate. Some responders omit the actual signer’s certificate from the response: this option can be used to supply the necessary certificate in such cases.

−trust_other
the certificates specified by the **−verify_other** option should be explicitly trusted and no additional checks will be performed on them. This is useful when the complete responder certificate chain is not available or trusting a root CA is not appropriate.

−VAfile file
file containing explicitly trusted responder certificates. Equivalent to the **−verify_other** and **−trust_other** options.

−noverify
don’t attempt to verify the OCSP response signature or the nonce values. This option will normally only be used for debugging since it disables all verification of the responders certificate.

−no_intern
ignore certificates contained in the OCSP response when searching for the signers certificate. With this option the signers certificate must be specified with either the **−verify_other** or **−VAfile** options.

−no_signature_verify
don’t check the signature on the OCSP response. Since this option tolerates invalid signatures on OCSP responses it will normally only be used for testing purposes.

−no_cert_verify
don’t verify the OCSP response signers certificate at all. Since this option allows the OCSP response to be signed by any certificate it should only be used for testing purposes.

−no_chain
do not use certificates in the response as additional untrusted CA certificates.

−no_cert_checks
don’t perform any additional checks on the OCSP response signers certificate. That is do not make any checks to see if the signers certificate is authorised to provide the necessary status information: as a result this option should only be used for testing purposes.

−validity_period nsec, **−status_age age**
these options specify the range of times, in seconds, which will be tolerated in an OCSP response. Each certificate status response includes a **notBefore** time and an optional **notAfter** time. The current time should fall between these two values, but the interval between the two times may be only a few seconds. In practice the OCSP responder and clients clocks may not be precisely synchronised and so such a check may fail. To avoid this the **−validity_period** option can be used to specify an acceptable error range in seconds, the default value is 5 minutes.

If the **notAfter** time is omitted from a response then this means that new status information is immediately available. In this case the age of the **notBefore** field is checked to see it is not older than **age** seconds old. By default this additional check is not performed.

−md5|−sha1|−sha256|−ripemod160|...
this option sets digest algorithm to use for certificate identification in the OCSP request. By default SHA−1 is used.

OCSP SERVER OPTIONS

−index indexfile
indexfile is a text index file in **ca** format containing certificate revocation information.

If the **index** option is specified the **ocsp** utility is in responder mode, otherwise it is in client mode. The request(s) the responder processes can be either specified on the command line (using **issuer** and **serial** options), supplied in a file (using the **respin** option) or via external OCSP clients (if **port** or **url** is specified).

If the **index** option is present then the **CA** and **rsigner** options must also be present.

−CA file
CA certificate corresponding to the revocation information in **indexfile**.

−rsigner file
The certificate to sign OCSP responses with.

−rother file
Additional certificates to include in the OCSP response.

−resp_no_certs
Don't include any certificates in the OCSP response.

−resp_key_id
Identify the signer certificate using the key ID, default is to use the subject name.

−rkey file
The private key to sign OCSP responses with: if not present the file specified in the **rsigner** option is used.

−port portnum
Port to listen for OCSP requests on. The port may also be specified using the **url** option.

−nrequest number
The OCSP server will exit after receiving **number** requests, default unlimited.

−nmin minutes, **−ndays days**
Number of minutes or days when fresh revocation information is available: used in the **nextUpdate** field. If neither option is present then the **nextUpdate** field is omitted meaning fresh revocation information is immediately available.

OCSP Response verification.

OCSP Response follows the rules specified in RFC2560.

Initially the OCSP responder certificate is located and the signature on the OCSP request checked using the responder certificate's public key.

Then a normal certificate verify is performed on the OCSP responder certificate building up a certificate chain in the process. The locations of the trusted certificates used to build the chain can be specified by the **CAfile** and **CApath** options or they will be looked for in the standard OpenSSL certificates directory.

If the initial verify fails then the OCSP verify process halts with an error.

Otherwise the issuing CA certificate in the request is compared to the OCSP responder certificate: if there is a match then the OCSP verify succeeds.

Otherwise the OCSP responder certificate's CA is checked against the issuing CA certificate in the request. If there is a match and the OCSPSigning extended key usage is present in the OCSP responder certificate then the OCSP verify succeeds.

Otherwise the root CA of the OCSP responders CA is checked to see if it is trusted for OCSP signing. If it is the OCSP verify succeeds.

If none of these checks is successful then the OCSP verify fails.

What this effectively means if that if the OCSP responder certificate is authorised directly by the CA it is issuing revocation information about (and it is correctly configured) then verification will succeed.

If the OCSP responder is a ''global responder'' which can give details about multiple CAs and has its own separate certificate chain then its root CA can be trusted for OCSP signing. For example:

```
openssl x509 -in ocspCA.pem -addtrust OCSPSigning -out trustedCA.pem
```

Alternatively the responder certificate itself can be explicitly trusted with the **−VAfile** option.

NOTES

As noted, most of the verify options are for testing or debugging purposes. Normally only the **−CApath**, **−CAfile** and (if the responder is a 'global VA') **−VAfile** options need to be used.

The OCSP server is only useful for test and demonstration purposes: it is not really usable as a full OCSP responder. It contains only a very simple HTTP request handling and can only handle the POST form of OCSP queries. It also handles requests serially meaning it cannot respond to new requests until it has processed the current one. The text index file format of revocation is also inefficient for large quantities of revocation data.

It is possible to run the **ocsp** application in responder mode via a CGI script using the **respin** and **respout** options.

EXAMPLES

Create an OCSP request and write it to a file:

```
openssl ocsp -issuer issuer.pem -cert c1.pem -cert c2.pem -reqout req.der
```

Send a query to an OCSP responder with URL http://ocsp.myhost.com/ save the response to a file and print it out in text form

```
openssl ocsp -issuer issuer.pem -cert c1.pem -cert c2.pem \
    -url http://ocsp.myhost.com/ -resp_text -respout resp.der
```

Read in an OCSP response and print out text form:

```
openssl ocsp -respin resp.der -text
```

OCSP server on port 8888 using a standard **ca** configuration, and a separate responder certificate. All requests and responses are printed to a file.

```
openssl ocsp -index demoCA/index.txt -port 8888 -rsigner rcert.pem -CA demoCA/cacer
       -text -out log.txt
```

As above but exit after processing one request:

```
openssl ocsp -index demoCA/index.txt -port 8888 -rsigner rcert.pem -CA demoCA/cacer
    -nrequest 1
```

Query status information using internally generated request:

```
openssl ocsp -index demoCA/index.txt -rsigner rcert.pem -CA demoCA/cacert.pem
    -issuer demoCA/cacert.pem -serial 1
```

Query status information using request read from a file, write response to a second file.

```
openssl ocsp -index demoCA/index.txt -rsigner rcert.pem -CA demoCA/cacert.pem
    -reqin req.der -respout resp.der
```

NAME

passwd − compute password hashes

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl passwd [**−crypt**] [**−1**] [**−apr1**] [**−salt** *string*] [**−in** *file*] [**−stdin**] [**−noverify**] [**−quiet**] [**−table**] {*password*}

DESCRIPTION

The **passwd** command computes the hash of a password typed at run-time or the hash of each password in a list. The password list is taken from the named file for option **−in file**, from stdin for option **−stdin**, or from the command line, or from the terminal otherwise. The Unix standard algorithm **crypt** and the MD5−based BSD password algorithm **1** and its Apache variant **apr1** are available.

OPTIONS

−crypt
Use the **crypt** algorithm (default).

−1 Use the MD5 based BSD password algorithm **1**.

−apr1
Use the **apr1** algorithm (Apache variant of the BSD algorithm).

−salt *string*
Use the specified salt. When reading a password from the terminal, this implies **−noverify**.

−in *file*
Read passwords from *file*.

−stdin
Read passwords from **stdin**.

−noverify
Don’t verify when reading a password from the terminal.

−quiet
Don’t output warnings when passwords given at the command line are truncated.

−table
In the output list, prepend the cleartext password and a TAB character to each password hash.

EXAMPLES

openssl passwd −crypt −salt xx password prints **xxj31ZMTZzkVA**.

openssl passwd −1 −salt xxxxxxxx password prints **1xxxxxxxx$UYCIxa628.9qXjpQCjM4a.**.

openssl passwd −apr1 −salt xxxxxxxx password prints **$apr1$xxxxxxxx$dxHfLAsjHkDRmG83UXe8K0**.

NAME

pkcs12 − PKCS#12 file utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl pkcs12 [**−export**] [**−chain**] [**−inkey filename**] [**−certfile filename**] [**−name name**] [**−caname name**] [**−in filename**] [**−out filename**] [**−noout**] [**−nomacver**] [**−nocerts**] [**−clcerts**] [**−cacerts**] [**−nokeys**] [**−info**] [**−des | −des3 | −idea | −aes128 | −aes192 | −aes256 | −camellia128 | −camellia192 | −camellia256 | −nodes**] [**−noiter**] [**−maciter | −nomaciter | −nomac**] [**−twopass**] [**−descert**] [**−certpbe cipher**] [**−keypbe cipher**] [**−macalg digest**] [**−keyex**] [**−keysig**] [**−password arg**] [**−passin arg**] [**−passout arg**] [**−rand file(s)**] [**−CAfile file**] [**−CApath dir**] [**−CSP name**]

DESCRIPTION

The **pkcs12** command allows PKCS#12 files (sometimes referred to as PFX files) to be created and parsed. PKCS#12 files are used by several programs including Netscape, MSIE and MS Outlook.

COMMAND OPTIONS

There are a lot of options the meaning of some depends of whether a PKCS#12 file is being created or parsed. By default a PKCS#12 file is parsed. A PKCS#12 file can be created by using the **−export** option (see below).

PARSING OPTIONS

−in filename

This specifies filename of the PKCS#12 file to be parsed. Standard input is used by default.

−out filename

The filename to write certificates and private keys to, standard output by default. They are all written in PEM format.

−pass arg, **−passin arg**

the PKCS#12 file (i.e. input file) password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−passout arg

pass phrase source to encrypt any outputed private keys with. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−noout

this option inhibits output of the keys and certificates to the output file version of the PKCS#12 file.

−clcerts

only output client certificates (not CA certificates).

−cacerts

only output CA certificates (not client certificates).

−nocerts

no certificates at all will be output.

−nokeys

no private keys will be output.

−info

output additional information about the PKCS#12 file structure, algorithms used and iteration counts.

−des

use DES to encrypt private keys before outputting.

−des3

use triple DES to encrypt private keys before outputting, this is the default.

−idea
use IDEA to encrypt private keys before outputting.

−aes128, **−aes192**, **−aes256**
use AES to encrypt private keys before outputting.

−camellia128, **−camellia192**, **−camellia256**
use Camellia to encrypt private keys before outputting.

−nodes
don't encrypt the private keys at all.

−nomacver
don't attempt to verify the integrity MAC before reading the file.

−twopass
prompt for separate integrity and encryption passwords: most software always assumes these are the same so this option will render such PKCS#12 files unreadable.

FILE CREATION OPTIONS

−export
This option specifies that a PKCS#12 file will be created rather than parsed.

−out filename
This specifies filename to write the PKCS#12 file to. Standard output is used by default.

−in filename
The filename to read certificates and private keys from, standard input by default. They must all be in PEM format. The order doesn't matter but one private key and its corresponding certificate should be present. If additional certificates are present they will also be included in the PKCS#12 file.

−inkey filename
file to read private key from. If not present then a private key must be present in the input file.

−name friendlyname
This specifies the ''friendly name'' for the certificate and private key. This name is typically displayed in list boxes by software importing the file.

−certfile filename
A filename to read additional certificates from.

−caname friendlyname
This specifies the ''friendly name'' for other certificates. This option may be used multiple times to specify names for all certificates in the order they appear. Netscape ignores friendly names on other certificates whereas MSIE displays them.

−pass arg, **−passout arg**
the PKCS#12 file (i.e. output file) password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−passin password
pass phrase source to decrypt any input private keys with. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−chain
if this option is present then an attempt is made to include the entire certificate chain of the user certificate. The standard CA store is used for this search. If the search fails it is considered a fatal error.

−descert
encrypt the certificate using triple DES, this may render the PKCS#12 file unreadable by some ''export grade'' software. By default the private key is encrypted using triple DES and the certificate using 40 bit RC2.

−keypbe alg, **−certpbe alg**
these options allow the algorithm used to encrypt the private key and certificates to be selected. Any PKCS#5 v1.5 or PKCS#12 PBE algorithm name can be used (see **NOTES** section for more information). If a a cipher name (as output by the **list-cipher-algorithms** command is specified then it is used with PKCS#5 v2.0. For interoperability reasons it is advisable to only use PKCS#12 algorithms.

−keyex|−keysig
specifies that the private key is to be used for key exchange or just signing. This option is only interpreted by MSIE and similar MS software. Normally ''export grade'' software will only allow 512 bit RSA keys to be used for encryption purposes but arbitrary length keys for signing. The **−keysig** option marks the key for signing only. Signing only keys can be used for S/MIME signing, authenticode (ActiveX control signing) and SSL client authentication, however due to a bug only MSIE 5.0 and later support the use of signing only keys for SSL client authentication.

−macalg digest
specify the MAC digest algorithm. If not included them SHA1 will be used.

−nomaciter, **−noiter**
these options affect the iteration counts on the MAC and key algorithms. Unless you wish to produce files compatible with MSIE 4.0 you should leave these options alone.

To discourage attacks by using large dictionaries of common passwords the algorithm that derives keys from passwords can have an iteration count applied to it: this causes a certain part of the algorithm to be repeated and slows it down. The MAC is used to check the file integrity but since it will normally have the same password as the keys and certificates it could also be attacked. By default both MAC and encryption iteration counts are set to 2048, using these options the MAC and encryption iteration counts can be set to 1, since this reduces the file security you should not use these options unless you really have to. Most software supports both MAC and key iteration counts. MSIE 4.0 doesn't support MAC iteration counts so it needs the **−nomaciter** option.

−maciter
This option is included for compatibility with previous versions, it used to be needed to use MAC iterations counts but they are now used by default.

−nomac
don't attempt to provide the MAC integrity.

−rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd*(3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

−CAfile file
CA storage as a file.

−CApath dir
CA storage as a directory. This directory must be a standard certificate directory: that is a hash of each subject name (using **x509 −hash**) should be linked to each certificate.

−CSP name
write **name** as a Microsoft CSP name.

NOTES

Although there are a large number of options most of them are very rarely used. For PKCS#12 file parsing only **−in** and **−out** need to be used for PKCS#12 file creation **−export** and **−name** are also used.

If none of the **−clcerts**, **−cacerts** or **−nocerts** options are present then all certificates will be output in the order they appear in the input PKCS#12 files. There is no guarantee that the first certificate present is the one corresponding to the private key. Certain software which requires a private key and certificate and assumes the first certificate in the file is the one corresponding to the private key: this may not always be the case. Using the **−clcerts** option will solve this problem by only outputting the certificate corresponding to the private key. If the CA certificates are required then they can be output to a separate file using the

−nokeys −cacerts options to just output CA certificates.

The **−keypbe** and **−certpbe** algorithms allow the precise encryption algorithms for private keys and certificates to be specified. Normally the defaults are fine but occasionally software can't handle triple DES encrypted private keys, then the option **−keypbe PBE−SHA1−RC2−40** can be used to reduce the private key encryption to 40 bit RC2. A complete description of all algorithms is contained in the **pkcs8** manual page.

EXAMPLES

Parse a PKCS#12 file and output it to a file:

```
openssl pkcs12 -in file.p12 -out file.pem
```

Output only client certificates to a file:

```
openssl pkcs12 -in file.p12 -clcerts -out file.pem
```

Don't encrypt the private key:

```
openssl pkcs12 -in file.p12 -out file.pem -nodes
```

Print some info about a PKCS#12 file:

```
openssl pkcs12 -in file.p12 -info -noout
```

Create a PKCS#12 file:

```
openssl pkcs12 -export -in file.pem -out file.p12 -name "My Certificate"
```

Include some extra certificates:

```
openssl pkcs12 -export -in file.pem -out file.p12 -name "My Certificate" \
 -certfile othercerts.pem
```

BUGS

Some would argue that the PKCS#12 standard is one big bug :−)

Versions of OpenSSL before 0.9.6a had a bug in the PKCS#12 key generation routines. Under rare circumstances this could produce a PKCS#12 file encrypted with an invalid key. As a result some PKCS#12 files which triggered this bug from other implementations (MSIE or Netscape) could not be decrypted by OpenSSL and similarly OpenSSL could produce PKCS#12 files which could not be decrypted by other implementations. The chances of producing such a file are relatively small: less than 1 in 256.

A side effect of fixing this bug is that any old invalidly encrypted PKCS#12 files cannot no longer be parsed by the fixed version. Under such circumstances the **pkcs12** utility will report that the MAC is OK but fail with a decryption error when extracting private keys.

This problem can be resolved by extracting the private keys and certificates from the PKCS#12 file using an older version of OpenSSL and recreating the PKCS#12 file from the keys and certificates using a newer version of OpenSSL. For example:

```
old-openssl -in bad.p12 -out keycerts.pem
openssl -in keycerts.pem -export -name "My PKCS#12 file" -out fixed.p12
```

SEE ALSO

openssl_pkcs8 (1)

NAME

pkcs7 − PKCS#7 utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl pkcs7 [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−out filename**] [**−print_certs**] [**−text**] [**−noout**] [**−engine id**]

DESCRIPTION

The **pkcs7** command processes PKCS#7 files in DER or PEM format.

COMMAND OPTIONS

−inform DER|PEM

This specifies the input format. **DER** format is DER encoded PKCS#7 v1.5 structure.**PEM** (the default) is a base64 encoded version of the DER form with header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read from or standard input if this option is not specified.

−out filename

specifies the output filename to write to or standard output by default.

−print_certs

prints out any certificates or CRLs contained in the file. They are preceded by their subject and issuer names in one line format.

−text

prints out certificates details in full rather than just subject and issuer names.

−noout

don't output the encoded version of the PKCS#7 structure (or certificates is **−print_certs** is set).

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

EXAMPLES

Convert a PKCS#7 file from PEM to DER:

```
openssl pkcs7 -in file.pem -outform DER -out file.der
```

Output all certificates in a file:

```
openssl pkcs7 -in file.pem -print_certs -out certs.pem
```

NOTES

The PEM PKCS#7 format uses the header and footer lines:

```
-----BEGIN PKCS7-----
-----END PKCS7-----
```

For compatibility with some CAs it will also accept:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

RESTRICTIONS

There is no option to print out all the fields of a PKCS#7 file.

This PKCS#7 routines only understand PKCS#7 v 1.5 as specified in RFC2315 they cannot currently parse, for example, the new CMS as described in RFC2630.

SEE ALSO

openssl_crl2pkcs7 (1)

NAME

pkcs8 − PKCS#8 format private key conversion tool

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl pkcs8 [**−topk8**] [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−passin arg**] [**−out filename**] [**−passout arg**] [**−noiter**] [**−nocrypt**] [**−nooct**] [**−embed**] [**−nsdb**] [**−v2 alg**] [**−v1 alg**] [**−engine id**]

DESCRIPTION

The **pkcs8** command processes private keys in PKCS#8 format. It can handle both unencrypted PKCS#8 PrivateKeyInfo format and EncryptedPrivateKeyInfo format with a variety of PKCS#5 (v1.5 and v2.0) and PKCS#12 algorithms.

COMMAND OPTIONS

−topk8

Normally a PKCS#8 private key is expected on input and a traditional format private key will be written. With the **−topk8** option the situation is reversed: it reads a traditional format private key and writes a PKCS#8 format key.

−inform DER|PEM

This specifies the input format. If a PKCS#8 format key is expected on input then either a **DER** or **PEM** encoded version of a PKCS#8 key will be expected. Otherwise the **DER** or **PEM** format of the traditional format private key is used.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read a key from or standard input if this option is not specified. If the key is encrypted a pass phrase will be prompted for.

−passin arg

the input file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−out filename

This specifies the output filename to write a key to or standard output by default. If any encryption options are set then a pass phrase will be prompted for. The output filename should **not** be the same as the input filename.

−passout arg

the output file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−nocrypt

PKCS#8 keys generated or input are normally PKCS#8 EncryptedPrivateKeyInfo structures using an appropriate password based encryption algorithm. With this option an unencrypted PrivateKeyInfo structure is expected or output. This option does not encrypt private keys at all and should only be used when absolutely necessary. Certain software such as some versions of Java code signing software used unencrypted private keys.

−nooct

This option generates RSA private keys in a broken format that some software uses. Specifically the private key should be enclosed in a OCTET STRING but some software just includes the structure itself without the surrounding OCTET STRING.

−embed

This option generates DSA keys in a broken format. The DSA parameters are embedded inside the PrivateKey structure. In this form the OCTET STRING contains an ASN1 SEQUENCE consisting of two

structures: a SEQUENCE containing the parameters and an ASN1 INTEGER containing the private key.

−nsdb

This option generates DSA keys in a broken format compatible with Netscape private key databases. The PrivateKey contains a SEQUENCE consisting of the public and private keys respectively.

−v2 alg

This option enables the use of PKCS#5 v2.0 algorithms. Normally PKCS#8 private keys are encrypted with the password based encryption algorithm called **pbeWithMD5AndDES−CBC** this uses 56 bit DES encryption but it was the strongest encryption algorithm supported in PKCS#5 v1.5. Using the **−v2** option PKCS#5 v2.0 algorithms are used which can use any encryption algorithm such as 168 bit triple DES or 128 bit RC2 however not many implementations support PKCS#5 v2.0 yet. If you are just using private keys with OpenSSL then this doesn't matter.

The **alg** argument is the encryption algorithm to use, valid values include **des**, **des3** and **rc2**. It is recommended that **des3** is used.

−v1 alg

This option specifies a PKCS#5 v1.5 or PKCS#12 algorithm to use. A complete list of possible algorithms is included below.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

NOTES

The encrypted form of a PEM encode PKCS#8 files uses the following headers and footers:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
-----END ENCRYPTED PRIVATE KEY-----
```

The unencrypted form uses:

```
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

Private keys encrypted using PKCS#5 v2.0 algorithms and high iteration counts are more secure that those encrypted using the traditional SSLeay compatible formats. So if additional security is considered important the keys should be converted.

The default encryption is only 56 bits because this is the encryption that most current implementations of PKCS#8 will support.

Some software may use PKCS#12 password based encryption algorithms with PKCS#8 format private keys: these are handled automatically but there is no option to produce them.

It is possible to write out DER encoded encrypted private keys in PKCS#8 format because the encryption details are included at an ASN1 level whereas the traditional format includes them at a PEM level.

PKCS#5 v1.5 and PKCS#12 algorithms.

Various algorithms can be used with the **−v1** command line option, including PKCS#5 v1.5 and PKCS#12. These are described in more detail below.

PBE−MD2−DES PBE−MD5−DES

These algorithms were included in the original PKCS#5 v1.5 specification. They only offer 56 bits of protection since they both use DES.

PBE−SHA1−RC2−64 PBE−MD2−RC2−64 PBE−MD5−RC2−64 PBE−SHA1−DES

These algorithms are not mentioned in the original PKCS#5 v1.5 specification but they use the same key derivation algorithm and are supported by some software. They are mentioned in PKCS#5 v2.0. They use either 64 bit RC2 or 56 bit DES.

PBE-SHA1-RC4-128 PBE-SHA1-RC4-40 PBE-SHA1-3DES PBE-SHA1-2DES PBE-SHA1-RC2-128 PBE-SHA1-RC2-40

These algorithms use the PKCS#12 password based encryption algorithm and allow strong encryption algorithms like triple DES or 128 bit RC2 to be used.

EXAMPLES

Convert a private from traditional to PKCS#5 v2.0 format using triple DES:

```
openssl pkcs8 -in key.pem -topk8 -v2 des3 -out enckey.pem
```

Convert a private key to PKCS#8 using a PKCS#5 1.5 compatible algorithm (DES):

```
openssl pkcs8 -in key.pem -topk8 -out enckey.pem
```

Convert a private key to PKCS#8 using a PKCS#12 compatible algorithm (3DES):

```
openssl pkcs8 -in key.pem -topk8 -out enckey.pem -v1 PBE-SHA1-3DES
```

Read a DER unencrypted PKCS#8 format private key:

```
openssl pkcs8 -inform DER -nocrypt -in key.der -out key.pem
```

Convert a private key from any PKCS#8 format to traditional format:

```
openssl pkcs8 -in pk8.pem -out key.pem
```

STANDARDS

Test vectors from this PKCS#5 v2.0 implementation were posted to the pkcs-tng mailing list using triple DES, DES and RC2 with high iteration counts, several people confirmed that they could decrypt the private keys produced and Therefore it can be assumed that the PKCS#5 v2.0 implementation is reasonably accurate at least as far as these algorithms are concerned.

The format of PKCS#8 DSA (and other) private keys is not well documented: it is hidden away in PKCS#11 v2.01, section 11.9. OpenSSL's default DSA PKCS#8 private key format complies with this standard.

BUGS

There should be an option that prints out the encryption algorithm in use and other details such as the iteration count.

PKCS#8 using triple DES and PKCS#5 v2.0 should be the default private key format for OpenSSL: for compatibility several of the utilities use the old format at present.

SEE ALSO

openssl_dsa (1), *openssl_rsa* (1), *openssl_genrsa* (1), *openssl_gendsa* (1)

NAME

rand − generate pseudo−random bytes

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl rand [**−out** *file*] [**−rand** *file(s)*] [**−base64**] *num*

DESCRIPTION

The **rand** command outputs *num* pseudo-random bytes after seeding the random number generator once. As in other **openssl** command line tools, PRNG seeding uses the file *$HOME/***.rnd** or **.rnd** in addition to the files given in the **−rand** option. A new *$HOME/***.rnd** or **.rnd** file will be written back if enough seeding was obtained from these sources.

OPTIONS

−out *file*

Write to *file* instead of standard output.

−rand *file(s)*

Use specified file or files or EGD socket (see *RAND_egd* (3)) for seeding the random number generator. Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

−base64

Perform base64 encoding on the output.

SEE ALSO

RAND_bytes (3)

NAME

req − PKCS#10 certificate request and certificate generating utility.

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl req [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−passin arg**] [**−out filename**] [**−passout arg**] [**−text**] [**−pubkey**] [**−noout**] [**−verify**] [**−modulus**] [**−new**] [**−rand file(s)**] [**−newkey rsa:bits**] [**−newkey dsa:file**] [**−newkey alg:file**] [**−nodes**] [**−key filename**] [**−keyform PEM|DER**] [**−keyout filename**] [**−[md5|sha1|md2|mdc2]**] [**−config filename**] [**−subj arg**] [**−multivalue−rdn**] [**−x509**] [**−days n**] [**−set_serial n**] [**−asn1−kludge**] [**−newhdr**] [**−extensions section**] [**−reqexts section**] [**−utf8**] [**−nameopt**] [**−batch**] [**−verbose**] [**−engine id**]

DESCRIPTION

The **req** command primarily creates and processes certificate requests in PKCS#10 format. It can additionally create self signed certificates for use as root CAs for example.

COMMAND OPTIONS

−inform DER|PEM

This specifies the input format. The **DER** option uses an ASN1 DER encoded form compatible with the PKCS#10. The **PEM** form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read a request from or standard input if this option is not specified. A request is only read if the creation options (**−new** and **−newkey**) are not specified.

−passin arg

the input file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−out filename

This specifies the output filename to write to or standard output by default.

−passout arg

the output file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−text

prints out the certificate request in text form.

−pubkey

outputs the public key.

−noout

this option prevents output of the encoded version of the request.

−modulus

this option prints out the value of the modulus of the public key contained in the request.

−verify

verifies the signature on the request.

−new

this option generates a new certificate request. It will prompt the user for the relevant field values. The actual fields prompted for and their maximum and minimum sizes are specified in the configuration file and any requested extensions.

If the **−key** option is not used it will generate a new RSA private key using information specified in the configuration file.

−rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd*(3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

−newkey arg
this option creates a new certificate request and a new private key. The argument takes one of several forms. **rsa:nbits**, where **nbits** is the number of bits, generates an RSA key **nbits** in size. **dsa:filename** generates a DSA key using the parameters in the file **filename**. **param:file** generates a key using the parameter file **file**, the algorithm is determined by the parameters. **algname:file** use algorithm **algname** and parameter file **file** the two algorithms must match or an error occurs. **algname** just uses algorithm **algname**.

−pkeyopt opt:value
set the public key algorithm option **opt** to **value**. The precise set of options supported depends on the public key algorithm used and its implementation. See **KEY GENERATION OPTIONS** in the **genpkey** manual page for more details.

−key filename
This specifies the file to read the private key from. It also accepts PKCS#8 format private keys for PEM format files.

−keyform PEM|DER
the format of the private key file specified in the **−key** argument. PEM is the default.

−keyout filename
this gives the filename to write the newly created private key to. If this option is not specified then the filename present in the configuration file is used.

−nodes
if this option is specified then if a private key is created it will not be encrypted.

−[md5|sha1|md2|mdc2]
this specifies the message digest to sign the request with. This overrides the digest algorithm specified in the configuration file. This option is ignored for DSA requests: they always use SHA1.

−config filename
this allows an alternative configuration file to be specified, this overrides the compile time filename or any specified in the **OPENSSL_CONF** environment variable.

−subj arg
sets subject name for new request or supersedes the subject name when processing a request. The arg must be formatted as */type0=value0/type1=value1/type2=...*, characters may be escaped by \ (backslash), no spaces are skipped.

−multivalue−rdn
this option causes the −subj argument to be interpreted with full support for multivalued RDNs. Example:

/DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe

If −multi−rdn is not used then the UID value is *123456+CN=John Doe*.

−x509
this option outputs a self signed certificate instead of a certificate request. This is typically used to generate a test certificate or a self signed root CA. The extensions added to the certificate (if any) are specified in the configuration file. Unless specified using the **set_serial** option **0** will be used for the serial number.

−days n
when the **−x509** option is being used this specifies the number of days to certify the certificate for. The default is 30 days.

−set_serial n
serial number to use when outputting a self signed certificate. This may be specified as a decimal value or a hex value if preceded by **0x**. It is possible to use negative serial numbers but this is not recommended.

−extensions section
−reqexts section
these options specify alternative sections to include certificate extensions (if the **−x509** option is present) or certificate request extensions. This allows several different sections to be used in the same configuration file to specify requests for a variety of purposes.

−utf8
this option causes field values to be interpreted as UTF8 strings, by default they are interpreted as ASCII. This means that the field values, whether prompted from a terminal or obtained from a configuration file, must be valid UTF8 strings.

−nameopt option
option which determines how the subject or issuer names are displayed. The **option** argument can be a single option or multiple options separated by commas. Alternatively the **−nameopt** switch may be used more than once to set multiple options. See the *openssl_x509* (1) manual page for details.

−asn1−kludge
by default the **req** command outputs certificate requests containing no attributes in the correct PKCS#10 format. However certain CAs will only accept requests containing no attributes in an invalid form: this option produces this invalid format.

More precisely the **Attributes** in a PKCS#10 certificate request are defined as a **SET OF Attribute**. They are **not OPTIONAL** so if no attributes are present then they should be encoded as an empty **SET OF**. The invalid form does not include the empty **SET OF** whereas the correct form does.

It should be noted that very few CAs still require the use of this option.

−newhdr
Adds the word **NEW** to the PEM file header and footer lines on the outputed request. Some software (Netscape certificate server) and some CAs need this.

−batch
non-interactive mode.

−verbose
print extra details about the operations being performed.

−engine id
specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

CONFIGURATION FILE FORMAT

The configuration options are specified in the **req** section of the configuration file. As with all configuration files if no value is specified in the specific section (i.e. **req**) then the initial unnamed or **default** section is searched too.

The options available are described in detail below.

input_password output_password
The passwords for the input private key file (if present) and the output private key file (if one will be created). The command line options **passin** and **passout** override the configuration file values.

default_bits
This specifies the default key size in bits. If not specified then 512 is used. It is used if the **−new** option is used. It can be overridden by using the **−newkey** option.

default_keyfile
This is the default filename to write a private key to. If not specified the key is written to standard output. This can be overridden by the **−keyout** option.

oid_file
This specifies a file containing additional **OBJECT IDENTIFIERS**. Each line of the file should consist of the numerical form of the object identifier followed by white space then the short name followed by white space and finally the long name.

oid_section
This specifies a section in the configuration file containing extra object identifiers. Each line should consist of the short name of the object identifier followed by = and the numerical form. The short and long names are the same when this option is used.

RANDFILE
This specifies a filename in which random number seed information is placed and read from, or an EGD socket (see *RAND_egd* (3)). It is used for private key generation.

encrypt_key
If this is set to **no** then if a private key is generated it is **not** encrypted. This is equivalent to the **−nodes** command line option. For compatibility **encrypt_rsa_key** is an equivalent option.

default_md
This option specifies the digest algorithm to use. Possible values include **md5 sha1 mdc2**. If not present then MD5 is used. This option can be overridden on the command line.

string_mask
This option masks out the use of certain string types in certain fields. Most users will not need to change this option.

It can be set to several values **default** which is also the default option uses PrintableStrings, T61Strings and BMPStrings if the **pkix** value is used then only PrintableStrings and BMPStrings will be used. This follows the PKIX recommendation in RFC2459. If the **utf8only** option is used then only UTF8Strings will be used: this is the PKIX recommendation in RFC2459 after 2003. Finally the **nombstr** option just uses PrintableStrings and T61Strings: certain software has problems with BMPStrings and UTF8Strings: in particular Netscape.

req_extensions
this specifies the configuration file section containing a list of extensions to add to the certificate request. It can be overridden by the **−reqexts** command line switch. See the *x509v3_config* (5) manual page for details of the extension section format.

x509_extensions
this specifies the configuration file section containing a list of extensions to add to certificate generated when the **−x509** switch is used. It can be overridden by the **−extensions** command line switch.

prompt
if set to the value **no** this disables prompting of certificate fields and just takes values from the config file directly. It also changes the expected format of the **distinguished_name** and **attributes** sections.

utf8
if set to the value **yes** then field values to be interpreted as UTF8 strings, by default they are interpreted as ASCII. This means that the field values, whether prompted from a terminal or obtained from a configuration file, must be valid UTF8 strings.

attributes
this specifies the section containing any request attributes: its format is the same as **distinguished_name**. Typically these may contain the challengePassword or unstructuredName types. They are currently ignored by OpenSSL's request signing utilities but some CAs might want them.

distinguished_name

This specifies the section containing the distinguished name fields to prompt for when generating a certificate or certificate request. The format is described in the next section.

DISTINGUISHED NAME AND ATTRIBUTE SECTION FORMAT

There are two separate formats for the distinguished name and attribute sections. If the **prompt** option is set to **no** then these sections just consist of field names and values: for example,

```
CN=My Name
OU=My Organization
emailAddress=someone@somewhere.org
```

This allows external programs (e.g. GUI based) to generate a template file with all the field names and values and just pass it to **req**. An example of this kind of configuration file is contained in the **EXAMPLES** section.

Alternatively if the **prompt** option is absent or not set to **no** then the file contains field prompting information. It consists of lines of the form:

```
fieldName="prompt"
fieldName_default="default field value"
fieldName_min= 2
fieldName_max= 4
```

''fieldName'' is the field name being used, for example commonName (or CN). The ''prompt'' string is used to ask the user to enter the relevant details. If the user enters nothing then the default value is used if no default value is present then the field is omitted. A field can still be omitted if a default value is present if the user just enters the '.' character.

The number of characters entered must be between the fieldName_min and fieldName_max limits: there may be additional restrictions based on the field being used (for example countryName can only ever be two characters long and must fit in a PrintableString).

Some fields (such as organizationName) can be used more than once in a DN. This presents a problem because configuration files will not recognize the same name occurring twice. To avoid this problem if the fieldName contains some characters followed by a full stop they will be ignored. So for example a second organizationName can be input by calling it ''1.organizationName''.

The actual permitted field names are any object identifier short or long names. These are compiled into OpenSSL and include the usual values such as commonName, countryName, localityName, organizationName, organizationUnitName, stateOrProvinceName. Additionally emailAddress is include as well as name, surname, givenName initials and dnQualifier.

Additional object identifiers can be defined with the **oid_file** or **oid_section** options in the configuration file. Any additional fields will be treated as though they were a DirectoryString.

EXAMPLES

Examine and verify certificate request:

```
openssl req -in req.pem -text -verify -noout
```

Create a private key and then generate a certificate request from it:

```
openssl genrsa -out key.pem 1024
openssl req -new -key key.pem -out req.pem
```

The same but just using req:

```
openssl req -newkey rsa:1024 -keyout key.pem -out req.pem
```

Generate a self signed root certificate:

```
openssl req -x509 -newkey rsa:1024 -keyout key.pem -out req.pem
```

Example of a file pointed to by the **oid_file** option:

```
1.2.3.4         shortName       A longer Name
1.2.3.6         otherName       Other longer Name
```

Example of a section pointed to by **oid_section** making use of variable expansion:

```
testoid1=1.2.3.5
testoid2=${testoid1}.6
```

Sample configuration file prompting for field values:

```
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions         = v3_ca

dirstring_type = nobmp

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AU
countryName_min                 = 2
countryName_max                 = 2

localityName                    = Locality Name (eg, city)

organizationalUnitName          = Organizational Unit Name (eg, section)

commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 40

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

[ v3_ca ]

subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
```

Sample configuration containing all field values:

```
RANDFILE                = $ENV::HOME/.rnd

[ req ]
default_bits            = 1024
default_keyfile         = keyfile.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
prompt                  = no
output_password         = mypass
```

```
[ req_distinguished_name ]
C                      = GB
ST                     = Test State or Province
L                      = Test Locality
O                      = Organization Name
OU                     = Organizational Unit Name
CN                     = Common Name
emailAddress           = test@email.address

[ req_attributes ]
challengePassword              = A challenge password
```

NOTES

The header and footer lines in the **PEM** format are normally:

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

some software (some versions of Netscape certificate server) instead needs:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
-----END NEW CERTIFICATE REQUEST-----
```

which is produced with the **−newhdr** option but is otherwise compatible. Either form is accepted transparently on input.

The certificate requests generated by **Xenroll** with MSIE have extensions added. It includes the **keyUsage** extension which determines the type of key (signature only or general purpose) and any additional OIDs entered by the script in an extendedKeyUsage extension.

DIAGNOSTICS

The following messages are frequently asked about:

```
Using configuration from /some/path/openssl.cnf
Unable to load config info
```

This is followed some time later by...

```
unable to find 'distinguished_name' in config
problems making Certificate Request
```

The first error message is the clue: it can't find the configuration file! Certain operations (like examining a certificate request) don't need a configuration file so its use isn't enforced. Generation of certificates or requests however does need a configuration file. This could be regarded as a bug.

Another puzzling message is this:

```
Attributes:
    a0:00
```

this is displayed when no attributes are present and the request includes the correct empty **SET OF** structure (the DER encoding of which is 0xa0 0x00). If you just see:

```
Attributes:
```

then the **SET OF** is missing and the encoding is technically invalid (but it is tolerated). See the description of the command line option **−asn1−kludge** for more information.

ENVIRONMENT VARIABLES

The variable **OPENSSL_CONF** if defined allows an alternative configuration file location to be specified, it will be overridden by the **−config** command line switch if it is present. For compatibility reasons the **SSLEAY_CONF** environment variable serves the same purpose but its use is discouraged.

BUGS

OpenSSL's handling of T61Strings (aka TeletexStrings) is broken: it effectively treats them as ISO−8859−1 (Latin 1), Netscape and MSIE have similar behaviour. This can cause problems if you need characters that

aren't available in PrintableStrings and you don't want to or can't use BMPStrings.

As a consequence of the T61String handling the only correct way to represent accented characters in OpenSSL is to use a BMPString: unfortunately Netscape currently chokes on these. If you have to use accented characters with Netscape and MSIE then you currently need to use the invalid T61String form.

The current prompting is not very friendly. It doesn't allow you to confirm what you've just entered. Other things like extensions in certificate requests are statically defined in the configuration file. Some of these: like an email address in subjectAltName should be input by the user.

SEE ALSO

openssl_x509 (1), *openssl_ca* (1), *openssl_genrsa* (1), *openssl_gendsa* (1), *openssl.cnf* (5), *x509v3_config* (5)

NAME

rsa − RSA key processing tool

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl rsa [**−inform PEM|NET|DER**] [**−outform PEM|NET|DER**] [**−in filename**] [**−passin arg**] [**−out filename**] [**−passout arg**] [**−sgckey**] [**−des**] [**−des3**] [**−idea**] [**−text**] [**−noout**] [**−modulus**] [**−check**] [**−pubin**] [**−pubout**] [**−engine id**]

DESCRIPTION

The **rsa** command processes RSA keys. They can be converted between various forms and their components printed out. **Note** this command uses the traditional SSLeay compatible format for private key encryption: newer applications should use the more secure PKCS#8 format using the **pkcs8** utility.

COMMAND OPTIONS

−inform DER|NET|PEM

This specifies the input format. The **DER** option uses an ASN1 DER encoded form compatible with the PKCS#1 RSAPrivateKey or SubjectPublicKeyInfo format. The **PEM** form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines. On input PKCS#8 format private keys are also accepted. The **NET** form is a format is described in the **NOTES** section.

−outform DER|NET|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read a key from or standard input if this option is not specified. If the key is encrypted a pass phrase will be prompted for.

−passin arg

the input file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−out filename

This specifies the output filename to write a key to or standard output if this option is not specified. If any encryption options are set then a pass phrase will be prompted for. The output filename should **not** be the same as the input filename.

−passout password

the output file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−sgckey

use the modified NET algorithm used with some versions of Microsoft IIS and SGC keys.

−des|−des3|−idea

These options encrypt the private key with the DES, triple DES, or the IDEA ciphers respectively before outputting it. A pass phrase is prompted for. If none of these options is specified the key is written in plain text. This means that using the **rsa** utility to read in an encrypted key with no encryption option can be used to remove the pass phrase from a key, or by setting the encryption options it can be use to add or change the pass phrase. These options can only be used with PEM format output files.

−text

prints out the various public or private key components in plain text in addition to the encoded version.

−noout

this option prevents output of the encoded version of the key.

−modulus

this option prints out the value of the modulus of the key.

−check
this option checks the consistency of an RSA private key.

−pubin
by default a private key is read from the input file: with this option a public key is read instead.

−pubout
by default a private key is output: with this option a public key will be output instead. This option is automatically set if the input is a public key.

−engine id
specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

NOTES

The PEM private key format uses the header and footer lines:

```
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----
```

The PEM public key format uses the header and footer lines:

```
-----BEGIN PUBLIC KEY-----
-----END PUBLIC KEY-----
```

The **NET** form is a format compatible with older Netscape servers and Microsoft IIS .key files, this uses unsalted RC4 for its encryption. It is not very secure and so should only be used when necessary.

Some newer version of IIS have additional data in the exported .key files. To use these with the utility, view the file with a binary editor and look for the string ''private−key'', then trace back to the byte sequence 0x30, 0x82 (this is an ASN1 SEQUENCE). Copy all the data from this point onwards to another file and use that as the input to the **rsa** utility with the **−inform NET** option. If you get an error after entering the password try the **−sgckey** option.

EXAMPLES

To remove the pass phrase on an RSA private key:

```
openssl rsa -in key.pem -out keyout.pem
```

To encrypt a private key using triple DES:

```
openssl rsa -in key.pem -des3 -out keyout.pem
```

To convert a private key from PEM to DER format:

```
openssl rsa -in key.pem -outform DER -out keyout.der
```

To print out the components of a private key to standard output:

```
openssl rsa -in key.pem -text -noout
```

To just output the public part of a private key:

```
openssl rsa -in key.pem -pubout -out pubkey.pem
```

BUGS

The command line password arguments don't currently work with **NET** format.

There should be an option that automatically handles .key files, without having to manually edit them.

SEE ALSO

openssl_pkcs8 (1), *openssl_dsa* (1), *openssl_genrsa* (1), *openssl_gendsa* (1)

NAME

rsautl − RSA utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl rsautl [**−in file**] [**−out file**] [**−inkey file**] [**−pubin**] [**−certin**] [**−sign**] [**−verify**] [**−encrypt**] [**−decrypt**] [**−pkcs**] [**−ssl**] [**−raw**] [**−hexdump**] [**−asn1parse**]

DESCRIPTION

The **rsautl** command can be used to sign, verify, encrypt and decrypt data using the RSA algorithm.

COMMAND OPTIONS

−in filename
: This specifies the input filename to read data from or standard input if this option is not specified.

−out filename
: specifies the output filename to write to or standard output by default.

−inkey file
: the input key file, by default it should be an RSA private key.

−pubin
: the input file is an RSA public key.

−certin
: the input is a certificate containing an RSA public key.

−sign
: sign the input data and output the signed result. This requires and RSA private key.

−verify
: verify the input data and output the recovered data.

−encrypt
: encrypt the input data using an RSA public key.

−decrypt
: decrypt the input data using an RSA private key.

−pkcs, −oaep, −ssl, −raw
: the padding to use: PKCS#1 v1.5 (the default), PKCS#1 OAEP, special padding used in SSL v2 backwards compatible handshakes, or no padding, respectively. For signatures, only **−pkcs** and **−raw** can be used.

−hexdump
: hex dump the output data.

−asn1parse
: asn1parse the output data, this is useful when combined with the **−verify** option.

NOTES

rsautl because it uses the RSA algorithm directly can only be used to sign or verify small pieces of data.

EXAMPLES

Sign some data using a private key:

```
openssl rsautl -sign -in file -inkey key.pem -out sig
```

Recover the signed data

```
openssl rsautl -verify -in sig -inkey key.pem
```

Examine the raw signed data:

```
openssl rsautl -verify -in file -inkey key.pem -raw -hexdump
```

```
0000 - 00 01 ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0010 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0020 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0030 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0040 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0050 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0060 - ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff ff   ................
0070 - ff ff ff ff 00 68 65 6c-6c 6f 20 77 6f 72 6c 64   .....hello world
```

The PKCS#1 block formatting is evident from this. If this was done using encrypt and decrypt the block would have been of type 2 (the second byte) and random padding data visible instead of the 0xff bytes.

It is possible to analyse the signature of certificates using this utility in conjunction with **asn1parse**. Consider the self signed example in certs/pca−cert.pem . Running **asn1parse** as follows yields:

```
openssl asn1parse -in pca-cert.pem
```

```
   0:d=0  hl=4 l= 742 cons: SEQUENCE
   4:d=1  hl=4 l= 591 cons:  SEQUENCE
   8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
  10:d=3  hl=2 l=   1 prim:    INTEGER           :02
  13:d=2  hl=2 l=   1 prim:   INTEGER           :00
  16:d=2  hl=2 l=  13 cons:   SEQUENCE
  18:d=3  hl=2 l=   9 prim:    OBJECT            :md5WithRSAEncryption
  29:d=3  hl=2 l=   0 prim:    NULL
  31:d=2  hl=2 l=  92 cons:   SEQUENCE
  33:d=3  hl=2 l=  11 cons:    SET
  35:d=4  hl=2 l=   9 cons:     SEQUENCE
  37:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
  42:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :AU
 ....
 599:d=1  hl=2 l=  13 cons:  SEQUENCE
 601:d=2  hl=2 l=   9 prim:   OBJECT            :md5WithRSAEncryption
 612:d=2  hl=2 l=   0 prim:   NULL
 614:d=1  hl=3 l= 129 prim:  BIT STRING
```

The final BIT STRING contains the actual signature. It can be extracted with:

```
openssl asn1parse -in pca-cert.pem -out sig -noout -strparse 614
```

The certificate public key can be extracted with:

```
openssl x509 -in test/testx509.pem -pubkey -noout >pubkey.pem
```

The signature can be analysed with:

```
openssl rsautl -in sig -verify -asn1parse -inkey pubkey.pem -pubin
```

```
   0:d=0  hl=2 l=  32 cons: SEQUENCE
   2:d=1  hl=2 l=  12 cons:  SEQUENCE
   4:d=2  hl=2 l=   8 prim:   OBJECT            :md5
  14:d=2  hl=2 l=   0 prim:   NULL
  16:d=1  hl=2 l=  16 prim:  OCTET STRING
     0000 - f3 46 9e aa 1a 4a 73 c9-37 ea 93 00 48 25 08 b5   .F...Js.7...H%..
```

This is the parsed version of an ASN1 DigestInfo structure. It can be seen that the digest used was md5. The actual part of the certificate that was signed can be extracted with:

```
openssl asn1parse -in pca-cert.pem -out tbs -noout -strparse 4
```

and its digest computed with:

```
openssl md5 -c tbs
MD5(tbs)= f3:46:9e:aa:1a:4a:73:c9:37:ea:93:00:48:25:08:b5
```

which it can be seen agrees with the recovered value above.

SEE ALSO

openssl_dgst (1), *openssl_rsa* (1), *openssl_genrsa* (1)

NAME

s_client − SSL/TLS client program

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl s_client [**−connect host:port**] [**−verify depth**] [**−cert filename**] [**−certform DER|PEM**] [**−key filename**] [**−keyform DER|PEM**] [**−pass arg**] [**−CApath directory**] [**−CAfile filename**] [**−reconnect**] [**−pause**] [**−showcerts**] [**−debug**] [**−msg**] [**−nbio_test**] [**−state**] [**−nbio**] [**−crlf**] [**−ign_eof**] [**−quiet**] [**−ssl2**] [**−ssl3**] [**−tls1**] [**−no_ssl2**] [**−no_ssl3**] [**−no_tls1**] [**−bugs**] [**−cipher cipherlist**] [**−starttls protocol**] [**−engine id**] [**−tlsextdebug**] [**−no_ticket**] [**−sess_out filename**] [**−sess_in filename**] [**−rand file(s)**]

DESCRIPTION

The **s_client** command implements a generic SSL/TLS client which connects to a remote host using SSL/TLS. It is a *very* useful diagnostic tool for SSL servers.

OPTIONS

−connect host:port

This specifies the host and optional port to connect to. If not specified then an attempt is made to connect to the local host on port 4433.

−cert certname

The certificate to use, if one is requested by the server. The default is not to use a certificate.

−certform format

The certificate format to use: DER or PEM. PEM is the default.

−key keyfile

The private key to use. If not specified then the certificate file will be used.

−keyform format

The private format to use: DER or PEM. PEM is the default.

−pass arg

the private key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−verify depth

The verify depth to use. This specifies the maximum length of the server certificate chain and turns on server certificate verification. Currently the verify operation continues after errors so all the problems with a certificate chain can be seen. As a side effect the connection will never fail due to a server certificate verify failure.

−CApath directory

The directory to use for server certificate verification. This directory must be in ‘‘hash format’’, see **verify** for more information. These are also used when building the client certificate chain.

−CAfile file

A file containing trusted certificates to use during server authentication and to use when attempting to build the client certificate chain.

−reconnect

reconnects to the same server 5 times using the same session ID, this can be used as a test that session caching is working.

−pause

pauses 1 second between each read and write call.

−showcerts

display the whole server certificate chain: normally only the server certificate itself is displayed.

−prexit
print session information when the program exits. This will always attempt to print out information even if the connection fails. Normally information will only be printed out once if the connection succeeds. This option is useful because the cipher in use may be renegotiated or the connection may fail because a client certificate is required or is requested only after an attempt is made to access a certain URL. Note: the output produced by this option is not always accurate because a connection might never have been established.

−state
prints out the SSL session states.

−debug
print extensive debugging information including a hex dump of all traffic.

−msg
show all protocol messages with hex dump.

−nbio_test
tests non-blocking I/O

−nbio
turns on non-blocking I/O

−crlf
this option translated a line feed from the terminal into CR+LF as required by some servers.

−ign_eof
inhibit shutting down the connection when end of file is reached in the input.

−quiet
inhibit printing of session and certificate information. This implicitly turns on **−ign_eof** as well.

−psk_identity identity
Use the PSK identity **identity** when using a PSK cipher suite.

−psk key
Use the PSK key **key** when using a PSK cipher suite. The key is given as a hexadecimal number without leading 0x, for example −psk 1a2b3c4d.

−ssl2, **−ssl3**, **−tls1**, **−no_ssl2**, **−no_ssl3**, **−no_tls1**
these options disable the use of certain SSL or TLS protocols. By default the initial handshake uses a method which should be compatible with all servers and permit them to use SSL v3, SSL v2 or TLS as appropriate.

Unfortunately there are a lot of ancient and broken servers in use which cannot handle this technique and will fail to connect. Some servers only work if TLS is turned off with the **−no_tls** option others will only support SSL v2 and may need the **−ssl2** option.

−bugs
there are several known bug in SSL and TLS implementations. Adding this option enables various workarounds.

−cipher cipherlist
this allows the cipher list sent by the client to be modified. Although the server determines which cipher suite is used it should take the first supported cipher in the list sent by the client. See the **ciphers** command for more information.

−starttls protocol
send the protocol-specific message(s) to switch to TLS for communication. **protocol** is a keyword for the intended protocol. Currently, the only supported keywords are ‘‘smtp’’, ‘‘pop3’’, ‘‘imap’’, and ‘‘ftp’’.

−tlsextdebug
print out a hex dump of any TLS extensions received from the server.

−no_ticket
disable RFC4507bis session ticket support.

−sess_out filename
output SSL session to **filename**

−sess_in sess.pem
load SSL session from **filename**. The client will attempt to resume a connection from this session.

−engine id
specifying an engine (by it's unique **id** string) will cause **s_client** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

−rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd*(3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

CONNECTED COMMANDS

If a connection is established with an SSL server then any data received from the server is displayed and any key presses will be sent to the server. When used interactively (which means neither **−quiet** nor **−ign_eof** have been given), the session will be renegotiated if the line begins with an **R**, and if the line begins with a **Q** or if end of file is reached, the connection will be closed down.

NOTES

s_client can be used to debug SSL servers. To connect to an SSL HTTP server the command:

```
openssl s_client -connect servername:443
```

would typically be used (https uses port 443). If the connection succeeds then an HTTP command can be given such as ''GET /'' to retrieve a web page.

If the handshake fails then there are several possible causes, if it is nothing obvious like no client certificate then the **−bugs**, **−ssl2**, **−ssl3**, **−tls1**, **−no_ssl2**, **−no_ssl3**, **−no_tls1** options can be tried in case it is a buggy server. In particular you should play with these options **before** submitting a bug report to an OpenSSL mailing list.

A frequent problem when attempting to get client certificates working is that a web client complains it has no certificates or gives an empty list to choose from. This is normally because the server is not sending the clients certificate authority in its ''acceptable CA list'' when it requests a certificate. By using **s_client** the CA list can be viewed and checked. However some servers only request client authentication after a specific URL is requested. To obtain the list in this case it is necessary to use the **−prexit** option and send an HTTP request for an appropriate page.

If a certificate is specified on the command line using the **−cert** option it will not be used unless the server specifically requests a client certificate. Therefor merely including a client certificate on the command line is no guarantee that the certificate works.

If there are problems verifying a server certificate then the **−showcerts** option can be used to show the whole chain.

Since the SSLv23 client hello cannot include compression methods or extensions these will only be supported if its use is disabled, for example by using the **−no_sslv2** option.

BUGS

Because this program has a lot of options and also because some of the techniques used are rather old, the C source of s_client is rather hard to read and not a model of how things should be done. A typical SSL client program would be much simpler.

The **−verify** option should really exit if the server verification fails.

The **−prexit** option is a bit of a hack. We should really report information whenever a session is renegotiated.

SEE ALSO

openssl_sess_id (1), *openssl_s_server* (1), *openssl_ciphers* (1)

NAME

s_server − SSL/TLS server program

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl s_server [**−accept port**] [**−context id**] [**−verify depth**] [**−Verify depth**] [**−cert filename**] [**−certform DER|PEM**] [**−key keyfile**] [**−keyform DER|PEM**] [**−pass arg**] [**−dcert filename**] [**−dcertform DER|PEM**] [**−dkey keyfile**] [**−dkeyform DER|PEM**] [**−dpass arg**] [**−dhparam filename**] [**−nbio**] [**−nbio_test**] [**−crlf**] [**−debug**] [**−msg**] [**−state**] [**−CApath directory**] [**−CAfile filename**] [**−nocert**] [**−cipher cipherlist**] [**−quiet**] [**−no_tmp_rsa**] [**−ssl2**] [**−ssl3**] [**−tls1**] [**−no_ssl2**] [**−no_ssl3**] [**−no_tls1**] [**−no_dhe**] [**−bugs**] [**−hack**] [**−www**] [**−WWW**] [**−HTTP**] [**−engine id**] [**−tlsextdebug**] [**−no_ticket**] [**−id_prefix arg**] [**−rand file(s)**]

DESCRIPTION

The **s_server** command implements a generic SSL/TLS server which listens for connections on a given port using SSL/TLS.

OPTIONS

−accept port

the TCP port to listen on for connections. If not specified 4433 is used.

−context id

sets the SSL context id. It can be given any string value. If this option is not present a default value will be used.

−cert certname

The certificate to use, most servers cipher suites require the use of a certificate and some require a certificate with a certain public key type: for example the DSS cipher suites require a certificate containing a DSS (DSA) key. If not specified then the filename ‘‘server.pem’’ will be used.

−certform format

The certificate format to use: DER or PEM. PEM is the default.

−key keyfile

The private key to use. If not specified then the certificate file will be used.

−keyform format

The private format to use: DER or PEM. PEM is the default.

−pass arg

the private key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−dcert filename, **−dkey keyname**

specify an additional certificate and private key, these behave in the same manner as the **−cert** and **−key** options except there is no default if they are not specified (no additional certificate and key is used). As noted above some cipher suites require a certificate containing a key of a certain type. Some cipher suites need a certificate carrying an RSA key and some a DSS (DSA) key. By using RSA and DSS certificates and keys a server can support clients which only support RSA or DSS cipher suites by using an appropriate certificate.

−dcertform format, **−dkeyform format**, **−dpass arg**

addtional certificate and private key format and passphrase respectively.

−nocert

if this option is set then no certificate is used. This restricts the cipher suites available to the anonymous ones (currently just anonymous DH).

−dhparam filename

the DH parameter file to use. The ephemeral DH cipher suites generate keys using a set of DH parameters. If not specified then an attempt is made to load the parameters from the server certificate file. If

this fails then a static set of parameters hard coded into the s_server program will be used.

−no_dhe
if this option is set then no DH parameters will be loaded effectively disabling the ephemeral DH cipher suites.

−no_tmp_rsa
certain export cipher suites sometimes use a temporary RSA key, this option disables temporary RSA key generation.

−verify depth, **−Verify depth**
The verify depth to use. This specifies the maximum length of the client certificate chain and makes the server request a certificate from the client. With the **−verify** option a certificate is requested but the client does not have to send one, with the **−Verify** option the client must supply a certificate or an error occurs.

−CApath directory
The directory to use for client certificate verification. This directory must be in ''hash format'', see **verify** for more information. These are also used when building the server certificate chain.

−CAfile file
A file containing trusted certificates to use during client authentication and to use when attempting to build the server certificate chain. The list is also used in the list of acceptable client CAs passed to the client when a certificate is requested.

−state
prints out the SSL session states.

−debug
print extensive debugging information including a hex dump of all traffic.

−msg
show all protocol messages with hex dump.

−nbio_test
tests non blocking I/O

−nbio
turns on non blocking I/O

−crlf
this option translated a line feed from the terminal into CR+LF.

−quiet
inhibit printing of session and certificate information.

−psk_hint hint
Use the PSK identity hint **hint** when using a PSK cipher suite.

−psk key
Use the PSK key **key** when using a PSK cipher suite. The key is given as a hexadecimal number without leading 0x, for example −psk 1a2b3c4d.

−ssl2, **−ssl3**, **−tls1**, **−no_ssl2**, **−no_ssl3**, **−no_tls1**
these options disable the use of certain SSL or TLS protocols. By default the initial handshake uses a method which should be compatible with all servers and permit them to use SSL v3, SSL v2 or TLS as appropriate.

−bugs
there are several known bug in SSL and TLS implementations. Adding this option enables various workarounds.

−hack
this option enables a further workaround for some some early Netscape SSL code (?).

−cipher cipherlist
this allows the cipher list used by the server to be modified. When the client sends a list of supported ciphers the first client cipher also included in the server list is used. Because the client specifies the preference order, the order of the server cipherlist irrelevant. See the **ciphers** command for more information.

−tlsextdebug
print out a hex dump of any TLS extensions received from the server.

−no_ticket
disable RFC4507bis session ticket support.

−www
sends a status message back to the client when it connects. This includes lots of information about the ciphers used and various session parameters. The output is in HTML format so this option will normally be used with a web browser.

−WWW
emulates a simple web server. Pages will be resolved relative to the current directory, for example if the URL https://myhost/page.html is requested the file ./page.html will be loaded.

−HTTP
emulates a simple web server. Pages will be resolved relative to the current directory, for example if the URL https://myhost/page.html is requested the file ./page.html will be loaded. The files loaded are assumed to contain a complete and correct HTTP response (lines that are part of the HTTP response line and headers must end with CRLF).

−engine id
specifying an engine (by it's unique **id** string) will cause **s_server** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

−id_prefix arg
generate SSL/TLS session IDs prefixed by **arg**. This is mostly useful for testing any SSL/TLS code (eg. proxies) that wish to deal with multiple servers, when each of which might be generating a unique range of session IDs (eg. with a certain prefix).

−rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd*(3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

CONNECTED COMMANDS

If a connection request is established with an SSL client and neither the **−www** nor the **−WWW** option has been used then normally any data received from the client is displayed and any key presses will be sent to the client.

Certain single letter commands are also recognized which perform special operations: these are listed below.

q end the current SSL connection but still accept new connections.

Q end the current SSL connection and exit.

r renegotiate the SSL session.

R renegotiate the SSL session and request a client certificate.

P send some plain text down the underlying TCP connection: this should cause the client to disconnect due to a protocol violation.

S print out some session cache status information.

NOTES

s_server can be used to debug SSL clients. To accept connections from a web browser the command:

```
openssl s_server -accept 443 -www
```

can be used for example.

Most web browsers (in particular Netscape and MSIE) only support RSA cipher suites, so they cannot connect to servers which don't use a certificate carrying an RSA key or a version of OpenSSL with RSA disabled.

Although specifying an empty list of CAs when requesting a client certificate is strictly speaking a protocol violation, some SSL clients interpret this to mean any CA is acceptable. This is useful for debugging purposes.

The session parameters can printed out using the **sess_id** program.

BUGS

Because this program has a lot of options and also because some of the techniques used are rather old, the C source of s_server is rather hard to read and not a model of how things should be done. A typical SSL server program would be much simpler.

The output of common ciphers is wrong: it just gives the list of ciphers that OpenSSL recognizes and the client supports.

There should be a way for the **s_server** program to print out details of any unknown cipher suites a client says it supports.

SEE ALSO

openssl_sess_id (1), *openssl_s_client* (1), *openssl_ciphers* (1)

NAME

s_time − SSL/TLS performance timing program

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl s_time [**−connect host:port**] [**−www page**] [**−cert filename**] [**−key filename**] [**−CApath directory**] [**−CAfile filename**] [**−reuse**] [**−new**] [**−verify depth**] [**−nbio**] [**−time seconds**] [**−ssl2**] [**−ssl3**] [**−bugs**] [**−cipher cipherlist**]

DESCRIPTION

The **s_client** command implements a generic SSL/TLS client which connects to a remote host using SSL/TLS. It can request a page from the server and includes the time to transfer the payload data in its timing measurements. It measures the number of connections within a given timeframe, the amount of data transferred (if any), and calculates the average time spent for one connection.

OPTIONS

−connect host:port

This specifies the host and optional port to connect to.

−www page

This specifies the page to GET from the server. A value of '/' gets the index.htm[l] page. If this parameter is not specified, then **s_time** will only perform the handshake to establish SSL connections but not transfer any payload data.

−cert certname

The certificate to use, if one is requested by the server. The default is not to use a certificate. The file is in PEM format.

−key keyfile

The private key to use. If not specified then the certificate file will be used. The file is in PEM format.

−verify depth

The verify depth to use. This specifies the maximum length of the server certificate chain and turns on server certificate verification. Currently the verify operation continues after errors so all the problems with a certificate chain can be seen. As a side effect the connection will never fail due to a server certificate verify failure.

−CApath directory

The directory to use for server certificate verification. This directory must be in ''hash format'', see **verify** for more information. These are also used when building the client certificate chain.

−CAfile file

A file containing trusted certificates to use during server authentication and to use when attempting to build the client certificate chain.

−new

performs the timing test using a new session ID for each connection. If neither **−new** nor **−reuse** are specified, they are both on by default and executed in sequence.

−reuse

performs the timing test using the same session ID; this can be used as a test that session caching is working. If neither **−new** nor **−reuse** are specified, they are both on by default and executed in sequence.

−nbio

turns on non-blocking I/O.

−ssl2, **−ssl3**

these options disable the use of certain SSL or TLS protocols. By default the initial handshake uses a method which should be compatible with all servers and permit them to use SSL v3, SSL v2 or TLS as appropriate. The timing program is not as rich in options to turn protocols on and off as the

openssl_s_client (1) program and may not connect to all servers.

Unfortunately there are a lot of ancient and broken servers in use which cannot handle this technique and will fail to connect. Some servers only work if TLS is turned off with the **−ssl3** option; others will only support SSL v2 and may need the **−ssl2** option.

−bugs
there are several known bug in SSL and TLS implementations. Adding this option enables various workarounds.

−cipher cipherlist
this allows the cipher list sent by the client to be modified. Although the server determines which cipher suite is used it should take the first supported cipher in the list sent by the client. See the *openssl_ciphers* (1) command for more information.

−time length
specifies how long (in seconds) **s_time** should establish connections and optionally transfer payload data from a server. Server and client performance and the link speed determine how many connections **s_time** can establish.

NOTES

s_client can be used to measure the performance of an SSL connection. To connect to an SSL HTTP server and get the default page the command

```
openssl s_time -connect servername:443 -www / -CApath yourdir -CAfile yourfile.pem
```

would typically be used (https uses port 443). 'commoncipher' is a cipher to which both client and server can agree, see the *openssl_ciphers* (1) command for details.

If the handshake fails then there are several possible causes, if it is nothing obvious like no client certificate then the **−bugs**, **−ssl2**, **−ssl3** options can be tried in case it is a buggy server. In particular you should play with these options **before** submitting a bug report to an OpenSSL mailing list.

A frequent problem when attempting to get client certificates working is that a web client complains it has no certificates or gives an empty list to choose from. This is normally because the server is not sending the clients certificate authority in its ''acceptable CA list'' when it requests a certificate. By using *openssl_s_client* (1) the CA list can be viewed and checked. However some servers only request client authentication after a specific URL is requested. To obtain the list in this case it is necessary to use the **−prexit** option of *openssl_s_client* (1) and send an HTTP request for an appropriate page.

If a certificate is specified on the command line using the **−cert** option it will not be used unless the server specifically requests a client certificate. Therefor merely including a client certificate on the command line is no guarantee that the certificate works.

BUGS

Because this program does not have all the options of the *openssl_s_client* (1) program to turn protocols on and off, you may not be able to measure the performance of all protocols with all servers.

The **−verify** option should really exit if the server verification fails.

SEE ALSO

openssl_s_client (1), *openssl_s_server* (1), *openssl_ciphers* (1)

NAME

sess_id − SSL/TLS session handling utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl sess_id [**−inform PEM|DER**] [**−outform PEM|DER**] [**−in filename**] [**−out filename**] [**−text**] [**−noout**] [**−context ID**]

DESCRIPTION

The **sess_id** process the encoded version of the SSL session structure and optionally prints out SSL session details (for example the SSL session master key) in human readable format. Since this is a diagnostic tool that needs some knowledge of the SSL protocol to use properly, most users will not need to use it.

−inform DER|PEM

This specifies the input format. The **DER** option uses an ASN1 DER encoded format containing session details. The precise format can vary from one version to the next. The **PEM** form is the default format: it consists of the **DER** format base64 encoded with additional header and footer lines.

−outform DER|PEM

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read session information from or standard input by default.

−out filename

This specifies the output filename to write session information to or standard output if this option is not specified.

−text

prints out the various public or private key components in plain text in addition to the encoded version.

−cert

if a certificate is present in the session it will be output using this option, if the **−text** option is also present then it will be printed out in text form.

−noout

this option prevents output of the encoded version of the session.

−context ID

this option can set the session id so the output session information uses the supplied ID. The ID can be any string of characters. This option wont normally be used.

OUTPUT

Typical output:

```
SSL-Session:
    Protocol  : TLSv1
    Cipher    : 0016
    Session-ID: 871E62626C554CE95488823752CBD5F3673A3EF3DCE9C67BD916C809914B40ED
    Session-ID-ctx: 01000000
    Master-Key: A7CEFC571974BE02CAC305269DC59F76EA9F0B180CB6642697A68251F2D2BB57E51
    Key-Arg   : None
    Start Time: 948459261
    Timeout   : 300 (sec)
    Verify return code 0 (ok)
```

Theses are described below in more detail.

Protocol

this is the protocol in use TLSv1, SSLv3 or SSLv2.

Cipher
the cipher used this is the actual raw SSL or TLS cipher code, see the SSL or TLS specifications for more information.

Session-ID
the SSL session ID in hex format.

Session-ID-ctx
the session ID context in hex format.

Master-Key
this is the SSL session master key.

Key-Arg
the key argument, this is only used in SSL v2.

Start Time
this is the session start time represented as an integer in standard Unix format.

Timeout
the timeout in seconds.

Verify return code
this is the return code when an SSL client certificate is verified.

NOTES

The PEM encoded session format uses the header and footer lines:

```
-----BEGIN SSL SESSION PARAMETERS-----
-----END SSL SESSION PARAMETERS-----
```

Since the SSL session output contains the master key it is possible to read the contents of an encrypted session using this information. Therefore appropriate security precautions should be taken if the information is being output by a ''real'' application. This is however strongly discouraged and should only be used for debugging purposes.

BUGS

The cipher and start time should be printed out in human readable form.

SEE ALSO

openssl_ciphers (1), *openssl_s_server* (1)

NAME

smime − S/MIME utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl smime [**−encrypt**] [**−decrypt**] [**−sign**] [**−resign**] [**−verify**] [**−pk7out**] [**−des**] [**−des3**] [**−rc2−40**] [**−rc2−64**] [**−rc2−128**] [**−aes128**] [**−aes192**] [**−aes256**] [**−camellia128**] [**−camellia192**] [**−camellia256**] [**−in file**] [**−certfile file**] [**−signer file**] [**−recip file**] [**−inform SMIME|PEM|DER**] [**−passin arg**] [**−inkey file**] [**−out file**] [**−outform SMIME|PEM|DER**] [**−content file**] [**−to addr**] [**−from ad**] [**−subject s**] [**−text**] [**−indef**] [**−noindef**] [**−stream**] [**−rand file(s)**] [**−md digest**] [cert.pem]...

DESCRIPTION

The **smime** command handles S/MIME mail. It can encrypt, decrypt, sign and verify S/MIME messages.

COMMAND OPTIONS

There are six operation options that set the type of operation to be performed. The meaning of the other options varies according to the operation type.

−encrypt

encrypt mail for the given recipient certificates. Input file is the message to be encrypted. The output file is the encrypted mail in MIME format.

−decrypt

decrypt mail using the supplied certificate and private key. Expects an encrypted mail message in MIME format for the input file. The decrypted mail is written to the output file.

−sign

sign mail using the supplied certificate and private key. Input file is the message to be signed. The signed message in MIME format is written to the output file.

−verify

verify signed mail. Expects a signed mail message on input and outputs the signed data. Both clear text and opaque signing is supported.

−pk7out

takes an input message and writes out a PEM encoded PKCS#7 structure.

−resign

resign a message: take an existing message and one or more new signers.

−in filename

the input message to be encrypted or signed or the MIME message to be decrypted or verified.

−inform SMIME|PEM|DER

this specifies the input format for the PKCS#7 structure. The default is **SMIME** which reads an S/MIME format message. **PEM** and **DER** format change this to expect PEM and DER format PKCS#7 structures instead. This currently only affects the input format of the PKCS#7 structure, if no PKCS#7 structure is being input (for example with **−encrypt** or **−sign**) this option has no effect.

−out filename

the message text that has been decrypted or verified or the output MIME format message that has been signed or verified.

−outform SMIME|PEM|DER

this specifies the output format for the PKCS#7 structure. The default is **SMIME** which write an S/MIME format message. **PEM** and **DER** format change this to write PEM and DER format PKCS#7 structures instead. This currently only affects the output format of the PKCS#7 structure, if no PKCS#7 structure is being output (for example with **−verify** or **−decrypt**) this option has no effect.

−stream −indef −noindef

the **−stream** and **−indef** options are equivalent and enable streaming I/O for encoding operations. This permits single pass processing of data without the need to hold the entire contents in memory, potentially supporting very large files. Streaming is automatically set for S/MIME signing with detached data if the output format is **SMIME** it is currently off by default for all other operations.

−noindef

disable streaming I/O where it would produce and indefinite length constructed encoding. This option currently has no effect. In future streaming will be enabled by default on all relevant operations and this option will disable it.

−content filename

This specifies a file containing the detached content, this is only useful with the **−verify** command. This is only usable if the PKCS#7 structure is using the detached signature form where the content is not included. This option will override any content if the input format is S/MIME and it uses the multipart/signed MIME content type.

−text

this option adds plain text (text/plain) MIME headers to the supplied message if encrypting or signing. If decrypting or verifying it strips off text headers: if the decrypted or verified message is not of MIME type text/plain then an error occurs.

−CAfile file

a file containing trusted CA certificates, only used with **−verify**.

−CApath dir

a directory containing trusted CA certificates, only used with **−verify**. This directory must be a standard certificate directory: that is a hash of each subject name (using **x509 −hash**) should be linked to each certificate.

−md digest

digest algorithm to use when signing or resigning. If not present then the default digest algorithm for the signing key will be used (usually SHA1).

−des −des3 −rc2−40 −rc2−64 −rc2−128 −aes128 −aes192 −aes256 −camellia128 −camellia192 −camellia256

the encryption algorithm to use. DES (56 bits), triple DES (168 bits), 40, 64 or 128 bit RC2, 128, 192 or 256 bit AES, or 128, 192 or 256 bit Camellia respectively. Any other cipher name (as recognized by the *EVP_get_cipherbyname()* function) can also be used preceded by a dash, for example **−aes_128_cbc**.

If not specified 40 bit RC2 is used. Only used with **−encrypt**.

−nointern

when verifying a message normally certificates (if any) included in the message are searched for the signing certificate. With this option only the certificates specified in the **−certfile** option are used. The supplied certificates can still be used as untrusted CAs however.

−noverify

do not verify the signers certificate of a signed message.

−nochain

do not do chain verification of signers certificates: that is don't use the certificates in the signed message as untrusted CAs.

−nosigs

don't try to verify the signatures on the message.

−nocerts

when signing a message the signer's certificate is normally included with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the **−certfile** option for example).

−noattr
normally when a message is signed a set of attributes are included which include the signing time and supported symmetric algorithms. With this option they are not included.

−binary
normally the input message is converted to ''canonical'' format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this option is present no translation occurs. This is useful when handling binary data which may not be in MIME format.

−nodetach
when signing a message use opaque signing: this form is more resistant to translation by mail relays but it cannot be read by mail agents that do not support S/MIME. Without this option cleartext signing with the MIME type multipart/signed is used.

−certfile file
allows additional certificates to be specified. When signing these will be included with the message. When verifying these will be searched for the signers certificates. The certificates should be in PEM format.

−signer file
a signing certificate when signing or resigning a message, this option can be used multiple times if more than one signer is required. If a message is being verified then the signers certificates will be written to this file if the verification was successful.

−recip file
the recipients certificate when decrypting a message. This certificate must match one of the recipients of the message or an error occurs.

−inkey file
the private key to use when signing or decrypting. This must match the corresponding certificate. If this option is not specified then the private key must be included in the certificate file specified with the **−recip** or **−signer** file. When signing this option can be used multiple times to specify successive keys.

−passin arg
the private key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−rand file(s)
a file or files containing random data used to seed the random number generator, or an EGD socket (see *RAND_egd* (3)). Multiple files can be specified separated by a OS-dependent character. The separator is **;** for MS−Windows, **,** for OpenVMS, and **:** for all others.

cert.pem...
one or more certificates of message recipients: used when encrypting a message.

−to, −from, −subject
the relevant mail headers. These are included outside the signed portion of a message so they may be included manually. If signing then many S/MIME mail clients check the signers certificate's email address matches that specified in the From: address.

NOTES

The MIME message must be sent without any blank lines between the headers and the output. Some mail programs will automatically add a blank line. Piping the mail directly to sendmail is one way to achieve the correct format.

The supplied message to be signed or encrypted must include the necessary MIME headers or many S/MIME clients wont display it properly (if at all). You can use the **−text** option to automatically add plain text headers.

A ''signed and encrypted'' message is one where a signed message is then encrypted. This can be produced by encrypting an already signed message: see the examples section.

This version of the program only allows one signer per message but it will verify multiple signers on

received messages. Some S/MIME clients choke if a message contains multiple signers. It is possible to sign messages ''in parallel'' by signing an already signed message.

The options **−encrypt** and **−decrypt** reflect common usage in S/MIME clients. Strictly speaking these process PKCS#7 enveloped data: PKCS#7 encrypted data is used for other purposes.

The **−resign** option uses an existing message digest when adding a new signer. This means that attributes must be present in at least one existing signer using the same message digest or this operation will fail.

The **−stream** and **−indef** options enable experimental streaming I/O support. As a result the encoding is BER using indefinite length constructed encoding and no longer DER. Streaming is supported for the **−encrypt** operation and the **−sign** operation if the content is not detached.

Streaming is always used for the **−sign** operation with detached data but since the content is no longer part of the PKCS#7 structure the encoding remains DER.

EXIT CODES

0 the operation was completely successfully.

1 an error occurred parsing the command options.

2 one of the input files could not be read.

3 an error occurred creating the PKCS#7 file or when reading the MIME message.

4 an error occurred decrypting or verifying the message.

5 the message was verified correctly but an error occurred writing out the signers certificates.

EXAMPLES

Create a cleartext signed message:

```
openssl smime -sign -in message.txt -text -out mail.msg \
       -signer mycert.pem
```

Create an opaque signed message

```
openssl smime -sign -in message.txt -text -out mail.msg -nodetach \
       -signer mycert.pem
```

Create a signed message, include some additional certificates and read the private key from another file:

```
openssl smime -sign -in in.txt -text -out mail.msg \
       -signer mycert.pem -inkey mykey.pem -certfile mycerts.pem
```

Create a signed message with two signers:

```
openssl smime -sign -in message.txt -text -out mail.msg \
       -signer mycert.pem -signer othercert.pem
```

Send a signed message under Unix directly to sendmail, including headers:

```
openssl smime -sign -in in.txt -text -signer mycert.pem \
       -from steve@openssl.org -to someone@somewhere \
       -subject "Signed message" | sendmail someone@somewhere
```

Verify a message and extract the signer's certificate if successful:

```
openssl smime -verify -in mail.msg -signer user.pem -out signedtext.txt
```

Send encrypted mail using triple DES:

```
openssl smime -encrypt -in in.txt -from steve@openssl.org \
       -to someone@somewhere -subject "Encrypted message" \
       -des3 user.pem -out mail.msg
```

Sign and encrypt mail:

```
openssl smime -sign -in ml.txt -signer my.pem -text \
       | openssl smime -encrypt -out mail.msg \
       -from steve@openssl.org -to someone@somewhere \
       -subject "Signed and Encrypted message" -des3 user.pem
```

Note: the encryption command does not include the **−text** option because the message being encrypted already has MIME headers.

Decrypt mail:

```
openssl smime -decrypt -in mail.msg -recip mycert.pem -inkey key.pem
```

The output from Netscape form signing is a PKCS#7 structure with the detached signature format. You can use this program to verify the signature by line wrapping the base64 encoded structure and surrounding it with:

```
-----BEGIN PKCS7-----
-----END PKCS7-----
```

and using the command,

```
openssl smime -verify -inform PEM -in signature.pem -content content.txt
```

alternatively you can base64 decode the signature and use

```
openssl smime -verify -inform DER -in signature.der -content content.txt
```

Create an encrypted message using 128 bit Camellia:

```
openssl smime -encrypt -in plain.txt -camellia128 -out mail.msg cert.pem
```

Add a signer to an existing message:

```
openssl smime -resign -in mail.msg -signer newsign.pem -out mail2.msg
```

BUGS

The MIME parser isn't very clever: it seems to handle most messages that I've thrown at it but it may choke on others.

The code currently will only write out the signer's certificate to a file: if the signer has a separate encryption certificate this must be manually extracted. There should be some heuristic that determines the correct encryption certificate.

Ideally a database should be maintained of a certificates for each email address.

The code doesn't currently take note of the permitted symmetric encryption algorithms as supplied in the SMIMECapabilities signed attribute. this means the user has to manually include the correct encryption algorithm. It should store the list of permitted ciphers in a database and only use those.

No revocation checking is done on the signer's certificate.

The current code can only handle S/MIME v2 messages, the more complex S/MIME v3 structures may cause parsing errors.

HISTORY

The use of multiple **−signer** options and the **−resign** command were first added in OpenSSL 0.9.9

NAME

speed − test library performance

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl speed [**−engine id**] [**md2**] [**mdc2**] [**md5**] [**hmac**] [**sha1**] [**rmd160**] [**idea-cbc**] [**rc2−cbc**] [**rc5−cbc**] [**bf-cbc**] [**des-cbc**] [**des−ede3**] [**rc4**] [**rsa512**] [**rsa1024**] [**rsa2048**] [**rsa4096**] [**dsa512**] [**dsa1024**] [**dsa2048**] [**idea**] [**rc2**] [**des**] [**rsa**] [**blowfish**]

DESCRIPTION

This command is used to test the performance of cryptographic algorithms.

OPTIONS

−engine id

specifying an engine (by it's unique **id** string) will cause **speed** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

[zero or more test algorithms]

If any options are given, **speed** tests those algorithms, otherwise all of the above are tested.

NAME

spkac − SPKAC printing and generating utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl spkac [**−in filename**] [**−out filename**] [**−key keyfile**] [**−passin arg**] [**−challenge string**] [**−pubkey**] [**−spkac spkacname**] [**−spksect section**] [**−noout**] [**−verify**] [**−engine id**]

DESCRIPTION

The **spkac** command processes Netscape signed public key and challenge (SPKAC) files. It can print out their contents, verify the signature and produce its own SPKACs from a supplied private key.

COMMAND OPTIONS

−in filename

This specifies the input filename to read from or standard input if this option is not specified. Ignored if the **−key** option is used.

−out filename

specifies the output filename to write to or standard output by default.

−key keyfile

create an SPKAC file using the private key in **keyfile**. The **−in**, **−noout**, **−spksect** and **−verify** options are ignored if present.

−passin password

the input file password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl* (1).

−challenge string

specifies the challenge string if an SPKAC is being created.

−spkac spkacname

allows an alternative name form the variable containing the SPKAC. The default is ‘‘SPKAC’’. This option affects both generated and input SPKAC files.

−spksect section

allows an alternative name form the section containing the SPKAC. The default is the default section.

−noout

don’t output the text version of the SPKAC (not used if an SPKAC is being created).

−pubkey

output the public key of an SPKAC (not used if an SPKAC is being created).

−verify

verifies the digital signature on the supplied SPKAC.

−engine id

specifying an engine (by it’s unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

EXAMPLES

Print out the contents of an SPKAC:

```
openssl spkac -in spkac.cnf
```

Verify the signature of an SPKAC:

```
openssl spkac -in spkac.cnf -noout -verify
```

Create an SPKAC using the challenge string ‘‘hello’’:

```
openssl spkac -key key.pem -challenge hello -out spkac.cnf
```

Example of an SPKAC, (long lines split up for clarity):

```
SPKAC=MIG5MGUwXDANBgkqhkiG9w0BAQEFAANLADBIAkEA1cCoq2Wa3Ixs47uI7F\
PVwHVIPDx5yso105Y6zpozam135a8R0CpoRvkkigIyXfcCjiVi5oWk+6FfPaD03u\
PFoQIDAQABFgVoZWxsbzANBgkqhkiG9w0BAQQFAANBAFpQtY/FojdwkJh1bEIYuc\
2EeM2KHTWPEepWYeawvHD0gQ3DngSC75YCWnnDdq+NQ3F+X4deMx9AaEglZtULwV\
4=
```

NOTES

A created SPKAC with suitable DN components appended can be fed into the **ca** utility.

SPKACs are typically generated by Netscape when a form is submitted containing the **KEYGEN** tag as part of the certificate enrollment process.

The challenge string permits a primitive form of proof of possession of private key. By checking the SPKAC signature and a random challenge string some guarantee is given that the user knows the private key corresponding to the public key being certified. This is important in some applications. Without this it is possible for a previous SPKAC to be used in a ‘‘replay attack’’.

SEE ALSO

openssl_ca (1)

NAME

verify − Utility to verify certificates.

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl verify [**−CApath directory**] [**−CAfile file**] [**−purpose purpose**] [**−untrusted file**] [**−help**] [**−issuer_checks**] [**−verbose**] [−] [certificates]

DESCRIPTION

The **verify** command verifies certificate chains.

COMMAND OPTIONS

−CApath directory

A directory of trusted certificates. The certificates should have names of the form: hash.0 or have symbolic links to them of this form (''hash'' is the hashed certificate subject name: see the **−hash** option of the **x509** utility). Under Unix the **c_rehash** script will automatically create symbolic links to a directory of certificates.

−CAfile file

A file of trusted certificates. The file should contain multiple certificates in PEM format concatenated together.

−untrusted file

A file of untrusted certificates. The file should contain multiple certificates

−purpose purpose

the intended use for the certificate. Without this option no chain verification will be done. Currently accepted uses are **sslclient**, **sslserver**, **nssslserver**, **smimesign**, **smimeencrypt**. See the **VERIFY OPERATION** section for more information.

−help

prints out a usage message.

−verbose

print extra information about the operations being performed.

−issuer_checks

print out diagnostics relating to searches for the issuer certificate of the current certificate. This shows why each candidate issuer certificate was rejected. However the presence of rejection messages does not itself imply that anything is wrong: during the normal verify process several rejections may take place.

− marks the last option. All arguments following this are assumed to be certificate files. This is useful if the first certificate filename begins with a −.

certificates

one or more certificates to verify. If no certificate filenames are included then an attempt is made to read a certificate from standard input. They should all be in PEM format.

VERIFY OPERATION

The **verify** program uses the same functions as the internal SSL and S/MIME verification, therefore this description applies to these verify operations too.

There is one crucial difference between the verify operations performed by the **verify** program: wherever possible an attempt is made to continue after an error whereas normally the verify operation would halt on the first error. This allows all the problems with a certificate chain to be determined.

The verify operation consists of a number of separate steps.

Firstly a certificate chain is built up starting from the supplied certificate and ending in the root CA. It is an error if the whole chain cannot be built up. The chain is built up by looking up the issuers certificate of the current certificate. If a certificate is found which is its own issuer it is assumed to be the root CA.

The process of 'looking up the issuers certificate' itself involves a number of steps. In versions of OpenSSL before 0.9.5a the first certificate whose subject name matched the issuer of the current certificate was assumed to be the issuers certificate. In OpenSSL 0.9.6 and later all certificates whose subject name matches the issuer name of the current certificate are subject to further tests. The relevant authority key identifier components of the current certificate (if present) must match the subject key identifier (if present) and issuer and serial number of the candidate issuer, in addition the keyUsage extension of the candidate issuer (if present) must permit certificate signing.

The lookup first looks in the list of untrusted certificates and if no match is found the remaining lookups are from the trusted certificates. The root CA is always looked up in the trusted certificate list: if the certificate to verify is a root certificate then an exact match must be found in the trusted list.

The second operation is to check every untrusted certificate's extensions for consistency with the supplied purpose. If the **−purpose** option is not included then no checks are done. The supplied or ''leaf'' certificate must have extensions compatible with the supplied purpose and all other certificates must also be valid CA certificates. The precise extensions required are described in more detail in the **CERTIFICATE EXTENSIONS** section of the **x509** utility.

The third operation is to check the trust settings on the root CA. The root CA should be trusted for the supplied purpose. For compatibility with previous versions of SSLeay and OpenSSL a certificate with no trust settings is considered to be valid for all purposes.

The final operation is to check the validity of the certificate chain. The validity period is checked against the current system time and the notBefore and notAfter dates in the certificate. The certificate signatures are also checked at this point.

If all operations complete successfully then certificate is considered valid. If any operation fails then the certificate is not valid.

DIAGNOSTICS

When a verify operation fails the output messages can be somewhat cryptic. The general form of the error message is:

```
server.pem: /C=AU/ST=Queensland/O=CryptSoft Pty Ltd/CN=Test CA (1024 bit)
error 24 at 1 depth lookup:invalid CA certificate
```

The first line contains the name of the certificate being verified followed by the subject name of the certificate. The second line contains the error number and the depth. The depth is number of the certificate being verified when a problem was detected starting with zero for the certificate being verified itself then 1 for the CA that signed the certificate and so on. Finally a text version of the error number is presented.

An exhaustive list of the error codes and messages is shown below, this also includes the name of the error code as defined in the header file x509_vfy.h Some of the error codes are defined but never returned: these are described as ''unused''.

0 X509_V_OK: ok
the operation was successful.

2 X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT: unable to get issuer certificate
the issuer certificate could not be found: this occurs if the issuer certificate of an untrusted certificate cannot be found.

3 X509_V_ERR_UNABLE_TO_GET_CRL: unable to get certificate CRL
the CRL of a certificate could not be found. Unused.

4 X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE: unable to decrypt certificate's signature
the certificate signature could not be decrypted. This means that the actual signature value could not be determined rather than it not matching the expected value, this is only meaningful for RSA keys.

5 X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE: unable to decrypt CRL's signature
the CRL signature could not be decrypted: this means that the actual signature value could not be determined rather than it not matching the expected value. Unused.

6 X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY: unable to decode issuer public key

the public key in the certificate SubjectPublicKeyInfo could not be read.

7 X509_V_ERR_CERT_SIGNATURE_FAILURE: certificate signature failure

the signature of the certificate is invalid.

8 X509_V_ERR_CRL_SIGNATURE_FAILURE: CRL signature failure

the signature of the certificate is invalid. Unused.

9 X509_V_ERR_CERT_NOT_YET_VALID: certificate is not yet valid

the certificate is not yet valid: the notBefore date is after the current time.

10 X509_V_ERR_CERT_HAS_EXPIRED: certificate has expired

the certificate has expired: that is the notAfter date is before the current time.

11 X509_V_ERR_CRL_NOT_YET_VALID: CRL is not yet valid

the CRL is not yet valid. Unused.

12 X509_V_ERR_CRL_HAS_EXPIRED: CRL has expired

the CRL has expired. Unused.

13 X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD: format error in certificate's notBefore field

the certificate notBefore field contains an invalid time.

14 X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD: format error in certificate's notAfter field

the certificate notAfter field contains an invalid time.

15 X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD: format error in CRL's lastUpdate field

the CRL lastUpdate field contains an invalid time. Unused.

16 X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD: format error in CRL's nextUpdate field

the CRL nextUpdate field contains an invalid time. Unused.

17 X509_V_ERR_OUT_OF_MEM: out of memory

an error occurred trying to allocate memory. This should never happen.

18 X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT: self signed certificate

the passed certificate is self signed and the same certificate cannot be found in the list of trusted certificates.

19 X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN: self signed certificate in certificate chain

the certificate chain could be built up using the untrusted certificates but the root could not be found locally.

20 X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY: unable to get local issuer certificate

the issuer certificate of a locally looked up certificate could not be found. This normally means the list of trusted certificates is not complete.

21 X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE: unable to verify the first certificate

no signatures could be verified because the chain contains only one certificate and it is not self signed.

22 X509_V_ERR_CERT_CHAIN_TOO_LONG: certificate chain too long

the certificate chain length is greater than the supplied maximum depth. Unused.

23 X509_V_ERR_CERT_REVOKED: certificate revoked

the certificate has been revoked. Unused.

24 X509_V_ERR_INVALID_CA: invalid CA certificate
a CA certificate is invalid. Either it is not a CA or its extensions are not consistent with the supplied purpose.

25 X509_V_ERR_PATH_LENGTH_EXCEEDED: path length constraint exceeded
the basicConstraints pathlength parameter has been exceeded.

26 X509_V_ERR_INVALID_PURPOSE: unsupported certificate purpose
the supplied certificate cannot be used for the specified purpose.

27 X509_V_ERR_CERT_UNTRUSTED: certificate not trusted
the root CA is not marked as trusted for the specified purpose.

28 X509_V_ERR_CERT_REJECTED: certificate rejected
the root CA is marked to reject the specified purpose.

29 X509_V_ERR_SUBJECT_ISSUER_MISMATCH: subject issuer mismatch
the current candidate issuer certificate was rejected because its subject name did not match the issuer name of the current certificate. Only displayed when the **−issuer_checks** option is set.

30 X509_V_ERR_AKID_SKID_MISMATCH: authority and subject key identifier mismatch
the current candidate issuer certificate was rejected because its subject key identifier was present and did not match the authority key identifier current certificate. Only displayed when the **−issuer_checks** option is set.

31 X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH: authority and issuer serial number mismatch
the current candidate issuer certificate was rejected because its issuer name and serial number was present and did not match the authority key identifier of the current certificate. Only displayed when the **−issuer_checks** option is set.

32 X509_V_ERR_KEYUSAGE_NO_CERTSIGN:key usage does not include certificate signing
the current candidate issuer certificate was rejected because its keyUsage extension does not permit certificate signing.

50 X509_V_ERR_APPLICATION_VERIFICATION: application verification failure
an application specific error. Unused.

BUGS

Although the issuer checks are a considerably improvement over the old technique they still suffer from limitations in the underlying X509_LOOKUP API. One consequence of this is that trusted certificates with matching subject name must either appear in a file (as specified by the **−CAfile** option) or a directory (as specified by **−CApath**. If they occur in both then only the certificates in the file will be recognised.

Previous versions of OpenSSL assume certificates with matching subject name are identical and mishandled them.

SEE ALSO

openssl_x509 (1)

NAME

version − print OpenSSL version information

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl version [**−a**] [**−v**] [**−b**] [**−o**] [**−f**] [**−p**]

DESCRIPTION

This command is used to print out version information about OpenSSL.

OPTIONS

−a all information, this is the same as setting all the other flags.

−v the current OpenSSL version.

−b the date the current version of OpenSSL was built.

−o option information: various options set when the library was built.

−c compilation flags.

−p platform setting.

−d OPENSSLDIR setting.

NOTES

The output of **openssl version −a** would typically be used when sending in a bug report.

HISTORY

The **−d** option was added in OpenSSL 0.9.7.

NAME

x509 − Certificate display and signing utility

LIBRARY

libcrypto, -lcrypto

SYNOPSIS

openssl x509 [**−inform DER|PEM|NET**] [**−outform DER|PEM|NET**] [**−keyform DER|PEM**] [**−CAform DER|PEM**] [**−CAkeyform DER|PEM**] [**−in filename**] [**−out filename**] [**−serial**] [**−hash**] [**−subject_hash**] [**−issuer_hash**] [**−subject**] [**−issuer**] [**−nameopt option**] [**−email**] [**−startdate**] [**−enddate**] [**−purpose**] [**−dates**] [**−modulus**] [**−fingerprint**] [**−alias**] [**−noout**] [**−trustout**] [**−clrtrust**] [**−clrreject**] [**−addtrust arg**] [**−addreject arg**] [**−setalias arg**] [**−days arg**] [**−set_serial n**] [**−signkey filename**] [**−x509toreq**] [**−req**] [**−CA filename**] [**−CAkey filename**] [**−CAcreateserial**] [**−CAserial filename**] [**−text**] [**−C**] [**−md2|−md5|−sha1|−mdc2**] [**−clrext**] [**−extfile filename**] [**−extensions section**] [**−engine id**]

DESCRIPTION

The **x509** command is a multi purpose certificate utility. It can be used to display certificate information, convert certificates to various forms, sign certificate requests like a ''mini CA'' or edit certificate trust settings.

Since there are a large number of options they will split up into various sections.

OPTIONS

INPUT, OUTPUT AND GENERAL PURPOSE OPTIONS

−inform DER|PEM|NET

This specifies the input format normally the command will expect an X509 certificate but this can change if other options such as **−req** are present. The DER format is the DER encoding of the certificate and PEM is the base64 encoding of the DER encoding with header and footer lines added. The NET option is an obscure Netscape server format that is now obsolete.

−outform DER|PEM|NET

This specifies the output format, the options have the same meaning as the **−inform** option.

−in filename

This specifies the input filename to read a certificate from or standard input if this option is not specified.

−out filename

This specifies the output filename to write to or standard output by default.

−md2|−md5|−sha1|−mdc2

the digest to use. This affects any signing or display option that uses a message digest, such as the **−fingerprint**, **−signkey** and **−CA** options. If not specified then SHA1 is used. If the key being used to sign with is a DSA key then this option has no effect: SHA1 is always used with DSA keys.

−engine id

specifying an engine (by it's unique **id** string) will cause **req** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

DISPLAY OPTIONS

Note: the **−alias** and **−purpose** options are also display options but are described in the **TRUST SETTINGS** section.

−text

prints out the certificate in text form. Full details are output including the public key, signature algorithms, issuer and subject names, serial number any extensions present and any trust settings.

−certopt option
customise the output format used with **−text**. The **option** argument can be a single option or multiple options separated by commas. The **−certopt** switch may be also be used more than once to set multiple options. See the **TEXT OPTIONS** section for more information.

−noout
this option prevents output of the encoded version of the request.

−modulus
this option prints out the value of the modulus of the public key contained in the certificate.

−serial
outputs the certificate serial number.

−subject_hash
outputs the ‘‘hash’’ of the certificate subject name. This is used in OpenSSL to form an index to allow certificates in a directory to be looked up by subject name.

−issuer_hash
outputs the ‘‘hash’’ of the certificate issuer name.

−hash
synonym for ‘‘−hash’’ for backward compatibility reasons.

−subject
outputs the subject name.

−issuer
outputs the issuer name.

−nameopt option
option which determines how the subject or issuer names are displayed. The **option** argument can be a single option or multiple options separated by commas. Alternatively the **−nameopt** switch may be used more than once to set multiple options. See the **NAME OPTIONS** section for more information.

−email
outputs the email address(es) if any.

−startdate
prints out the start date of the certificate, that is the notBefore date.

−enddate
prints out the expiry date of the certificate, that is the notAfter date.

−dates
prints out the start and expiry dates of a certificate.

−fingerprint
prints out the digest of the DER encoded version of the whole certificate (see digest options).

−C this outputs the certificate in the form of a C source file.

TRUST SETTINGS

Please note these options are currently experimental and may well change.

A **trusted certificate** is an ordinary certificate which has several additional pieces of information attached to it such as the permitted and prohibited uses of the certificate and an ‘‘alias’’.

Normally when a certificate is being verified at least one certificate must be ‘‘trusted’’. By default a trusted certificate must be stored locally and must be a root CA: any certificate chain ending in this CA is then usable for any purpose.

Trust settings currently are only used with a root CA. They allow a finer control over the purposes the root CA can be used for. For example a CA may be trusted for SSL client but not SSL server use.

See the description of the **verify** utility for more information on the meaning of trust settings.

Future versions of OpenSSL will recognize trust settings on any certificate: not just root CAs.

−trustout
this causes **x509** to output a **trusted** certificate. An ordinary or trusted certificate can be input but by default an ordinary certificate is output and any trust settings are discarded. With the **−trustout** option a trusted certificate is output. A trusted certificate is automatically output if any trust settings are modified.

−setalias arg
sets the alias of the certificate. This will allow the certificate to be referred to using a nickname for example ‘‘Steve’s Certificate’’.

−alias
outputs the certificate alias, if any.

−clrtrust
clears all the permitted or trusted uses of the certificate.

−clrreject
clears all the prohibited or rejected uses of the certificate.

−addtrust arg
adds a trusted certificate use. Any object name can be used here but currently only **clientAuth** (SSL client use), **serverAuth** (SSL server use) and **emailProtection** (S/MIME email) are used. Other OpenSSL applications may define additional uses.

−addreject arg
adds a prohibited use. It accepts the same values as the **−addtrust** option.

−purpose
this option performs tests on the certificate extensions and outputs the results. For a more complete description see the **CERTIFICATE EXTENSIONS** section.

SIGNING OPTIONS

The **x509** utility can be used to sign certificates and requests: it can thus behave like a ‘‘mini CA’’.

−signkey filename
this option causes the input file to be self signed using the supplied private key.

If the input file is a certificate it sets the issuer name to the subject name (i.e. makes it self signed) changes the public key to the supplied value and changes the start and end dates. The start date is set to the current time and the end date is set to a value determined by the **−days** option. Any certificate extensions are retained unless the **−clrext** option is supplied.

If the input is a certificate request then a self signed certificate is created using the supplied private key using the subject name in the request.

−clrext
delete any extensions from a certificate. This option is used when a certificate is being created from another certificate (for example with the **−signkey** or the **−CA** options). Normally all extensions are retained.

−keyform PEM|DER
specifies the format (DER or PEM) of the private key file used in the **−signkey** option.

−days arg
specifies the number of days to make a certificate valid for. The default is 30 days.

−x509toreq
converts a certificate into a certificate request. The **−signkey** option is used to pass the required private key.

−req
by default a certificate is expected on input. With this option a certificate request is expected instead.

−set_serial n
specifies the serial number to use. This option can be used with either the **−signkey** or **−CA** options. If used in conjunction with the **−CA** option the serial number file (as specified by the **−CAserial** or **−CAcreateserial** options) is not used.

The serial number can be decimal or hex (if preceded by **0x**). Negative serial numbers can also be specified but their use is not recommended.

−CA filename
specifies the CA certificate to be used for signing. When this option is present **x509** behaves like a ‘‘mini CA’’. The input file is signed by this CA using this option: that is its issuer name is set to the subject name of the CA and it is digitally signed using the CAs private key.

This option is normally combined with the **−req** option. Without the **−req** option the input is a certificate which must be self signed.

−CAkey filename
sets the CA private key to sign a certificate with. If this option is not specified then it is assumed that the CA private key is present in the CA certificate file.

−CAserial filename
sets the CA serial number file to use.

When the **−CA** option is used to sign a certificate it uses a serial number specified in a file. This file consist of one line containing an even number of hex digits with the serial number to use. After each use the serial number is incremented and written out to the file again.

The default filename consists of the CA certificate file base name with ‘‘.srl’’ appended. For example if the CA certificate file is called ‘‘mycacert.pem’’ it expects to find a serial number file called ‘‘mycacert.srl’’.

−CAcreateserial
with this option the CA serial number file is created if it does not exist: it will contain the serial number ‘‘02’’ and the certificate being signed will have the 1 as its serial number. Normally if the **−CA** option is specified and the serial number file does not exist it is an error.

−extfile filename
file containing certificate extensions to use. If not specified then no extensions are added to the certificate.

−extensions section
the section to add certificate extensions from. If this option is not specified then the extensions should either be contained in the unnamed (default) section or the default section should contain a variable called ‘‘extensions’’ which contains the section to use. See the *x509v3_config*(5) manual page for details of the extension section format.

NAME OPTIONS

The **nameopt** command line switch determines how the subject and issuer names are displayed. If no **nameopt** switch is present the default ‘‘oneline’’ format is used which is compatible with previous versions of OpenSSL. Each option is described in detail below, all options can be preceded by a **−** to turn the option off. Only the first four will normally be used.

compat
use the old format. This is equivalent to specifying no name options at all.

RFC2253
displays names compatible with RFC2253 equivalent to **esc_2253**, **esc_ctrl**, **esc_msb**, **utf8**, **dump_nostr**, **dump_unknown**, **dump_der**, **sep_comma_plus**, **dn_rev** and **sname**.

oneline
a oneline format which is more readable than RFC2253. It is equivalent to specifying the **esc_2253**, **esc_ctrl**, **esc_msb**, **utf8**, **dump_nostr**, **dump_der**, **use_quote**, **sep_comma_plus_space**, **space_eq** and **sname** options.

multiline
a multiline format. It is equivalent **esc_ctrl**, **esc_msb**, **sep_multiline**, **space_eq**, **lname** and **align**.

esc_2253
escape the ''special'' characters required by RFC2253 in a field That is **,+"<>;**. Additionally **#** is escaped at the beginning of a string and a space character at the beginning or end of a string.

esc_ctrl
escape control characters. That is those with ASCII values less than 0x20 (space) and the delete (0x7f) character. They are escaped using the RFC2253 \XX notation (where XX are two hex digits representing the character value).

esc_msb
escape characters with the MSB set, that is with ASCII values larger than 127.

use_quote
escapes some characters by surrounding the whole string with **"** characters, without the option all escaping is done with the **** character.

utf8
convert all strings to UTF8 format first. This is required by RFC2253. If you are lucky enough to have a UTF8 compatible terminal then the use of this option (and **not** setting **esc_msb**) may result in the correct display of multibyte (international) characters. Is this option is not present then multibyte characters larger than 0xff will be represented using the format \UXXXX for 16 bits and \WXXXXXXXX for 32 bits. Also if this option is off any UTF8Strings will be converted to their character form first.

no_type
this option does not attempt to interpret multibyte characters in any way. That is their content octets are merely dumped as though one octet represents each character. This is useful for diagnostic purposes but will result in rather odd looking output.

show_type
show the type of the ASN1 character string. The type precedes the field contents. For example ''BMPSTRING: Hello World''.

dump_der
when this option is set any fields that need to be hexdumped will be dumped using the DER encoding of the field. Otherwise just the content octets will be displayed. Both options use the RFC2253 **#XXXX...** format.

dump_nostr
dump non character string types (for example OCTET STRING) if this option is not set then non character string types will be displayed as though each content octet represents a single character.

dump_all
dump all fields. This option when used with **dump_der** allows the DER encoding of the structure to be unambiguously determined.

dump_unknown
dump any field whose OID is not recognised by OpenSSL.

sep_comma_plus, **sep_comma_plus_space**, **sep_semi_plus_space**, **sep_multiline**
these options determine the field separators. The first character is between RDNs and the second between multiple AVAs (multiple AVAs are very rare and their use is discouraged). The options ending in ''space'' additionally place a space after the separator to make it more readable. The **sep_multiline** uses a linefeed character for the RDN separator and a spaced **+** for the AVA separator. It also indents the fields by four characters.

dn_rev
reverse the fields of the DN. This is required by RFC2253. As a side effect this also reverses the order of multiple AVAs but this is permissible.

nofname, **sname**, **lname**, **oid**
these options alter how the field name is displayed. **nofname** does not display the field at all. **sname** uses the ‘‘short name’’ form (CN for commonName for example). **lname** uses the long form. **oid** represents the OID in numerical form and is useful for diagnostic purpose.

align
align field values for a more readable output. Only usable with **sep_multiline**.

space_eq
places spaces round the = character which follows the field name.

TEXT OPTIONS

As well as customising the name output format, it is also possible to customise the actual fields printed using the **certopt** options when the **text** option is present. The default behaviour is to print all fields.

compatible
use the old format. This is equivalent to specifying no output options at all.

no_header
don’t print header information: that is the lines saying ‘‘Certificate’’ and ‘‘Data’’.

no_version
don’t print out the version number.

no_serial
don’t print out the serial number.

no_signame
don’t print out the signature algorithm used.

no_validity
don’t print the validity, that is the **notBefore** and **notAfter** fields.

no_subject
don’t print out the subject name.

no_issuer
don’t print out the issuer name.

no_pubkey
don’t print out the public key.

no_sigdump
don’t give a hexadecimal dump of the certificate signature.

no_aux
don’t print out certificate trust information.

no_extensions
don’t print out any X509V3 extensions.

ext_default
retain default extension behaviour: attempt to print out unsupported certificate extensions.

ext_error
print an error message for unsupported certificate extensions.

ext_parse
ASN1 parse unsupported extensions.

ext_dump
hex dump unsupported extensions.

ca_default
the value used by the **ca** utility, equivalent to **no_issuer**, **no_pubkey**, **no_header**, **no_version**, **no_sigdump** and **no_signame**.

EXAMPLES

Note: in these examples the '\' means the example should be all on one line.

Display the contents of a certificate:

```
openssl x509 -in cert.pem -noout -text
```

Display the certificate serial number:

```
openssl x509 -in cert.pem -noout -serial
```

Display the certificate subject name:

```
openssl x509 -in cert.pem -noout -subject
```

Display the certificate subject name in RFC2253 form:

```
openssl x509 -in cert.pem -noout -subject -nameopt RFC2253
```

Display the certificate subject name in oneline form on a terminal supporting UTF8:

```
openssl x509 -in cert.pem -noout -subject -nameopt oneline,-esc_msb
```

Display the certificate MD5 fingerprint:

```
openssl x509 -in cert.pem -noout -fingerprint
```

Display the certificate SHA1 fingerprint:

```
openssl x509 -sha1 -in cert.pem -noout -fingerprint
```

Convert a certificate from PEM to DER format:

```
openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER
```

Convert a certificate to a certificate request:

```
openssl x509 -x509toreq -in cert.pem -out req.pem -signkey key.pem
```

Convert a certificate request into a self signed certificate using extensions for a CA:

```
openssl x509 -req -in careq.pem -extfile openssl.cnf -extensions v3_ca \
       -signkey key.pem -out cacert.pem
```

Sign a certificate request using the CA certificate above and add user certificate extensions:

```
openssl x509 -req -in req.pem -extfile openssl.cnf -extensions v3_usr \
       -CA cacert.pem -CAkey key.pem -CAcreateserial
```

Set a certificate to be trusted for SSL client use and change set its alias to ''Steve's Class 1 CA''

```
openssl x509 -in cert.pem -addtrust clientAuth \
       -setalias "Steve's Class 1 CA" -out trust.pem
```

NOTES

The PEM format uses the header and footer lines:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

it will also handle files containing:

```
-----BEGIN X509 CERTIFICATE-----
-----END X509 CERTIFICATE-----
```

Trusted certificates have the lines

```
-----BEGIN TRUSTED CERTIFICATE-----
-----END TRUSTED CERTIFICATE-----
```

The conversion to UTF8 format used with the name options assumes that T61Strings use the ISO8859−1 character set. This is wrong but Netscape and MSIE do this as do many certificates. So although this is incorrect it is more likely to display the majority of certificates correctly.

The **−fingerprint** option takes the digest of the DER encoded certificate. This is commonly called a ''fingerprint''. Because of the nature of message digests the fingerprint of a certificate is unique to that certificate and two certificates with the same fingerprint can be considered to be the same.

The Netscape fingerprint uses MD5 whereas MSIE uses SHA1.

The **−email** option searches the subject name and the subject alternative name extension. Only unique email addresses will be printed out: it will not print the same address more than once.

CERTIFICATE EXTENSIONS

The **−purpose** option checks the certificate extensions and determines what the certificate can be used for. The actual checks done are rather complex and include various hacks and workarounds to handle broken certificates and software.

The same code is used when verifying untrusted certificates in chains so this section is useful if a chain is rejected by the verify code.

The basicConstraints extension CA flag is used to determine whether the certificate can be used as a CA. If the CA flag is true then it is a CA, if the CA flag is false then it is not a CA. **All** CAs should have the CA flag set to true.

If the basicConstraints extension is absent then the certificate is considered to be a ''possible CA'' other extensions are checked according to the intended use of the certificate. A warning is given in this case because the certificate should really not be regarded as a CA: however it is allowed to be a CA to work around some broken software.

If the certificate is a V1 certificate (and thus has no extensions) and it is self signed it is also assumed to be a CA but a warning is again given: this is to work around the problem of Verisign roots which are V1 self signed certificates.

If the keyUsage extension is present then additional restraints are made on the uses of the certificate. A CA certificate **must** have the keyCertSign bit set if the keyUsage extension is present.

The extended key usage extension places additional restrictions on the certificate uses. If this extension is present (whether critical or not) the key can only be used for the purposes specified.

A complete description of each test is given below. The comments about basicConstraints and keyUsage and V1 certificates above apply to **all** CA certificates.

SSL Client

The extended key usage extension must be absent or include the ''web client authentication'' OID. keyUsage must be absent or it must have the digitalSignature bit set. Netscape certificate type must be absent or it must have the SSL client bit set.

SSL Client CA

The extended key usage extension must be absent or include the ''web client authentication'' OID. Netscape certificate type must be absent or it must have the SSL CA bit set: this is used as a work around if the basicConstraints extension is absent.

SSL Server

The extended key usage extension must be absent or include the ''web server authentication'' and/or one of the SGC OIDs. keyUsage must be absent or it must have the digitalSignature, the keyEncipherment set or both bits set. Netscape certificate type must be absent or have the SSL server bit set.

SSL Server CA

The extended key usage extension must be absent or include the ''web server authentication'' and/or one of the SGC OIDs. Netscape certificate type must be absent or the SSL CA bit must be set: this is

used as a work around if the basicConstraints extension is absent.

Netscape SSL Server

For Netscape SSL clients to connect to an SSL server it must have the keyEncipherment bit set if the keyUsage extension is present. This isn't always valid because some cipher suites use the key for digital signing. Otherwise it is the same as a normal SSL server.

Common S/MIME Client Tests

The extended key usage extension must be absent or include the "email protection" OID. Netscape certificate type must be absent or should have the S/MIME bit set. If the S/MIME bit is not set in netscape certificate type then the SSL client bit is tolerated as an alternative but a warning is shown: this is because some Verisign certificates don't set the S/MIME bit.

S/MIME Signing

In addition to the common S/MIME client tests the digitalSignature bit must be set if the keyUsage extension is present.

S/MIME Encryption

In addition to the common S/MIME tests the keyEncipherment bit must be set if the keyUsage extension is present.

S/MIME CA

The extended key usage extension must be absent or include the "email protection" OID. Netscape certificate type must be absent or must have the S/MIME CA bit set: this is used as a work around if the basicConstraints extension is absent.

CRL Signing

The keyUsage extension must be absent or it must have the CRL signing bit set.

CRL Signing CA

The normal CA tests apply. Except in this case the basicConstraints extension must be present.

BUGS

Extensions in certificates are not transferred to certificate requests and vice versa.

It is possible to produce invalid certificates or requests by specifying the wrong private key or using inconsistent options in some cases: these should be checked.

There should be options to explicitly set such things as start and end dates rather than an offset from the current time.

The code to implement the verify behaviour described in the **TRUST SETTINGS** is currently being developed. It thus describes the intended behaviour rather than the current behaviour. It is hoped that it will represent reality in OpenSSL 0.9.5 and later.

SEE ALSO

openssl_req (1), *openssl_ca* (1), *openssl_genrsa* (1), *openssl_gendsa* (1), *openssl_verify* (1), *x509v3_config* (5)

HISTORY

Before OpenSSL 0.9.8, the default digest for RSA keys was MD5.

NAME

x509v3_config − X509 V3 certificate extension configuration format

DESCRIPTION

Several of the OpenSSL utilities can add extensions to a certificate or certificate request based on the contents of a configuration file.

Typically the application will contain an option to point to an extension section. Each line of the extension section takes the form:

```
extension_name=[critical,] extension_options
```

If **critical** is present then the extension will be critical.

The format of **extension_options** depends on the value of **extension_name**.

There are four main types of extension: *string* extensions, *multi-valued* extensions, *raw* and *arbitrary* extensions.

String extensions simply have a string which contains either the value itself or how it is obtained.

For example:

```
nsComment="This is a Comment"
```

Multi-valued extensions have a short form and a long form. The short form is a list of names and values:

```
basicConstraints=critical,CA:true,pathlen:1
```

The long form allows the values to be placed in a separate section:

```
basicConstraints=critical,@bs_section

[bs_section]

CA=true
pathlen=1
```

Both forms are equivalent.

The syntax of raw extensions is governed by the extension code: it can for example contain data in multiple sections. The correct syntax to use is defined by the extension code itself: check out the certificate policies extension for an example.

If an extension type is unsupported then the *arbitrary* extension syntax must be used, see the ARBITRARY EXTENSIONS section for more details.

STANDARD EXTENSIONS

The following sections describe each supported extension in detail.

Basic Constraints.

This is a multi valued extension which indicates whether a certificate is a CA certificate. The first (mandatory) name is **CA** followed by **TRUE** or **FALSE**. If **CA** is **TRUE** then an optional **pathlen** name followed by an non-negative value can be included.

For example:

```
basicConstraints=CA:TRUE

basicConstraints=CA:FALSE

basicConstraints=critical,CA:TRUE, pathlen:0
```

A CA certificate **must** include the basicConstraints value with the CA field set to TRUE. An end user certificate must either set CA to FALSE or exclude the extension entirely. Some software may require the inclusion of basicConstraints with CA set to FALSE for end entity certificates.

The pathlen parameter indicates the maximum number of CAs that can appear below this one in a chain. So if you have a CA with a pathlen of zero it can only be used to sign end user certificates and not further CAs.

Key Usage.

Key usage is a multi valued extension consisting of a list of names of the permitted key usages.

The supporte names are: digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly and decipherOnly.

Examples:

```
keyUsage=digitalSignature, nonRepudiation
keyUsage=critical, keyCertSign
```

Extended Key Usage.

This extensions consists of a list of usages indicating purposes for which the certificate public key can be used for,

These can either be object short names of the dotted numerical form of OIDs. While any OID can be used only certain values make sense. In particular the following PKIX, NS and MS values are meaningful:

```
Value                Meaning
-----                -------
serverAuth           SSL/TLS Web Server Authentication.
clientAuth           SSL/TLS Web Client Authentication.
codeSigning          Code signing.
emailProtection      E-mail Protection (S/MIME).
timeStamping         Trusted Timestamping
msCodeInd            Microsoft Individual Code Signing (authenticode)
msCodeCom            Microsoft Commercial Code Signing (authenticode)
msCTLSign            Microsoft Trust List Signing
msSGC                Microsoft Server Gated Crypto
msEFS                Microsoft Encrypted File System
nsSGC                Netscape Server Gated Crypto
```

Examples:

```
extendedKeyUsage=critical,codeSigning,1.2.3.4
extendedKeyUsage=nsSGC,msSGC
```

Subject Key Identifier.

This is really a string extension and can take two possible values. Either the word **hash** which will automatically follow the guidelines in RFC3280 or a hex string giving the extension value to include. The use of the hex string is strongly discouraged.

Example:

```
subjectKeyIdentifier=hash
```

Authority Key Identifier.

The authority key identifier extension permits two options. keyid and issuer: both can take the optional value ‘‘always’’.

If the keyid option is present an attempt is made to copy the subject key identifier from the parent certificate. If the value ‘‘always’’ is present then an error is returned if the option fails.

The issuer option copies the issuer and serial number from the issuer certificate. This will only be done if the keyid option fails or is not included unless the ‘‘always’’ flag will always include the value.

Example:

```
authorityKeyIdentifier=keyid,issuer
```

Subject Alternative Name.

The subject alternative name extension allows various literal values to be included in the configuration file. These include **email** (an email address) **URI** a uniform resource indicator, **DNS** (a DNS domain name), **RID** (a registered ID: OBJECT IDENTIFIER), **IP** (an IP address), **dirName** (a distinguished name) and otherName.

The email option include a special 'copy' value. This will automatically include and email addresses contained in the certificate subject name in the extension.

The IP address used in the **IP** options can be in either IPv4 or IPv6 format.

The value of **dirName** should point to a section containing the distinguished name to use as a set of name value pairs. Multi values AVAs can be formed by preceeding the name with a + character.

otherName can include arbitrary data associated with an OID: the value should be the OID followed by a semicolon and the content in standard *ASN1_generate_nconf*(3) format.

Examples:

```
subjectAltName=email:copy,email:my@other.address,URI:http://my.url.here/
subjectAltName=IP:192.168.7.1
subjectAltName=IP:13::17
subjectAltName=email:my@other.address,RID:1.2.3.4
subjectAltName=otherName:1.2.3.4;UTF8:some other identifier
```

```
subjectAltName=dirName:dir_sect
```

```
[dir_sect]
C=UK
O=My Organization
OU=My Unit
CN=My Name
```

Issuer Alternative Name.

The issuer alternative name option supports all the literal options of subject alternative name. It does **not** support the email:copy option because that would not make sense. It does support an additional issuer:copy option that will copy all the subject alternative name values from the issuer certificate (if possible).

Example:

```
issuserAltName = issuer:copy
```

Authority Info Access.

The authority information access extension gives details about how to access certain information relating to the CA. Its syntax is accessOID;location where *location* has the same syntax as subject alternative name (except that email:copy is not supported). accessOID can be any valid OID but only certain values are meaningful, for example OCSP and caIssuers.

Example:

```
authorityInfoAccess = OCSP;URI:http://ocsp.my.host/
authorityInfoAccess = caIssuers;URI:http://my.ca/ca.html
```

CRL distribution points.

This is a multi-valued extension whose options can be either in name:value pair using the same form as subject alternative name or a single value representing a section name containing all the distribution point fields.

For a name:value pair a new DistributionPoint with the fullName field set to the given value both the cRLissuer and reasons fields are omitted in this case.

In the single option case the section indicated contains values for each field. In this section:

If the name is ‘‘fullname’’ the value field should contain the full name of the distribution point in the same format as subject alternative name.

If the name is ‘‘relativename’’ then the value field should contain a section name whose contents represent a DN fragment to be placed in this field.

The name ‘‘CRLIssuer’’ if present should contain a value for this field in subject alternative name format.

If the name is ‘‘reasons’’ the value field should consist of a comma separated field containing the reasons. Valid reasons are: ‘‘keyCompromise’’, ‘‘CACompromise’’, ‘‘affiliationChanged’’, ‘‘superseded’’, ‘‘cessationOfOperation’’, ‘‘certificateHold’’, ‘‘privilegeWithdrawn’’ and ‘‘AACompromise’’.

Simple examples:

```
crlDistributionPoints=URI:http://myhost.com/myca.crl
crlDistributionPoints=URI:http://my.com/my.crl,URI:http://oth.com/my.crl
```

Full distribution point example:

```
crlDistributionPoints=crldp1_section

[crldp1_section]

fullname=URI:http://myhost.com/myca.crl
CRLissuer=dirName:issuer_sect
reasons=keyCompromise, CACompromise

[issuer_sect]
C=UK
O=Organisation
CN=Some Name
```

Issuing Distribution Point

This extension should only appear in CRLs. It is a multi valued extension whose syntax is similar to the ‘‘section’’ pointed to by the CRL distribution points extension with a few differences.

The names ‘‘reasons’’ and ‘‘CRLissuer’’ are not recognized.

The name ‘‘onlysomereasons’’ is accepted which sets this field. The value is in the same format as the CRL distribution point ‘‘reasons’’ field.

The names ‘‘onlyuser’’, ‘‘onlyCA’’, ‘‘onlyAA’’ and ‘‘indirectCRL’’ are also accepted the values should be a boolean value (TRUE or FALSE) to indicate the value of the corresponding field.

Example:

```
issuingDistributionPoint=critical, @idp_section

[idp_section]

fullname=URI:http://myhost.com/myca.crl
indirectCRL=TRUE
onlysomereasons=keyCompromise, CACompromise

[issuer_sect]
C=UK
O=Organisation
CN=Some Name
```

Certificate Policies.

This is a *raw* extension. All the fields of this extension can be set by using the appropriate syntax.

If you follow the PKIX recommendations and just using one OID then you just include the value of that OID. Multiple OIDs can be set separated by commas, for example:

```
certificatePolicies= 1.2.4.5, 1.1.3.4
```

If you wish to include qualifiers then the policy OID and qualifiers need to be specified in a separate section: this is done by using the `@section` syntax instead of a literal OID value.

The section referred to must include the policy OID using the name policyIdentifier, cPSuri qualifiers can be included using the syntax:

```
CPS.nnn=value
```

userNotice qualifiers can be set using the syntax:

```
userNotice.nnn=@notice
```

The value of the userNotice qualifier is specified in the relevant section. This section can include explicitText, organization and noticeNumbers options. explicitText and organization are text strings, noticeNumbers is a comma separated list of numbers. The organization and noticeNumbers options (if included) must BOTH be present. If you use the userNotice option with IE5 then you need the 'ia5org' option at the top level to modify the encoding: otherwise it will not be interpreted properly.

Example:

```
certificatePolicies=ia5org,1.2.3.4,1.5.6.7.8,@polsect

[polsect]

policyIdentifier = 1.3.5.8
CPS.1="http://my.host.name/"
CPS.2="http://my.your.name/"
userNotice.1=@notice

[notice]

explicitText="Explicit Text Here"
organization="Organisation Name"
noticeNumbers=1,2,3,4
```

The **ia5org** option changes the type of the *organization* field. In RFC2459 it can only be of type DisplayText. In RFC3280 IA5Strring is also permissible. Some software (for example some versions of MSIE) may require ia5org.

Policy Constraints

This is a multi-valued extension which consisting of the names **requireExplicitPolicy** or **inhibitPolicyMapping** and a non negative intger value. At least one component must be present.

Example:

```
policyConstraints = requireExplicitPolicy:3
```

Inhibit Any Policy

This is a string extension whose value must be a non negative integer.

Example:

```
inhibitAnyPolicy = 2
```

Name Constraints

The name constraints extension is a multi-valued extension. The name should begin with the word **permitted** or **excluded** followed by a **;**. The rest of the name and the value follows the syntax of subjectAltName except email:copy is not supported and the **IP** form should consist of an IP addresses and subnet mask separated by a /.

Examples:

```
 nameConstraints=permitted;IP:192.168.0.0/255.255.0.0

 nameConstraints=permitted;email:.somedomain.com

 nameConstraints=excluded;email:.com
issuingDistributionPoint = idp_section
```

OCSP No Check

The OCSP No Check extension is a string extension but its value is ignored.

Example:

```
 noCheck = ignored
```

DEPRECATED EXTENSIONS

The following extensions are non standard, Netscape specific and largely obsolete. Their use in new applications is discouraged.

Netscape String extensions.

Netscape Comment (**nsComment**) is a string extension containing a comment which will be displayed when the certificate is viewed in some browsers.

Example:

```
 nsComment = "Some Random Comment"
```

Other supported extensions in this category are: **nsBaseUrl**, **nsRevocationUrl**, **nsCaRevocationUrl**, **nsRenewalUrl**, **nsCaPolicyUrl** and **nsSslServerName**.

Netscape Certificate Type

This is a multi-valued extensions which consists of a list of flags to be included. It was used to indicate the purposes for which a certificate could be used. The basicConstraints, keyUsage and extended key usage extensions are now used instead.

Acceptable values for nsCertType are: **client**, **server**, **email**, **objsign**, **reserved**, **sslCA**, **emailCA**, **objCA**.

ARBITRARY EXTENSIONS

If an extension is not supported by the OpenSSL code then it must be encoded using the arbitrary extension format. It is also possible to use the arbitrary format for supported extensions. Extreme care should be taken to ensure that the data is formatted correctly for the given extension type.

There are two ways to encode arbitrary extensions.

The first way is to use the word ASN1 followed by the extension content using the same syntax as *ASN1_generate_nconf*(3). For example:

```
 1.2.3.4=critical,ASN1:UTF8String:Some random data

 1.2.3.4=ASN1:SEQUENCE:seq_sect

 [seq_sect]

 field1 = UTF8:field1
 field2 = UTF8:field2
```

It is also possible to use the word DER to include the raw encoded data in any extension.

```
 1.2.3.4=critical,DER:01:02:03:04
 1.2.3.4=DER:01020304
```

The value following DER is a hex dump of the DER encoding of the extension Any extension can be placed in this form to override the default behaviour. For example:

```
 basicConstraints=critical,DER:00:01:02:03
```

WARNING

There is no guarantee that a specific implementation will process a given extension. It may therefore be sometimes possible to use certificates for purposes prohibited by their extensions because a specific application does not recognize or honour the values of the relevant extensions.

The DER and ASN1 options should be used with caution. It is possible to create totally invalid extensions if they are not used carefully.

NOTES

If an extension is multi-value and a field value must contain a comma the long form must be used otherwise the comma would be misinterpreted as a field separator. For example:

```
subjectAltName=URI:ldap://somehost.com/CN=foo,OU=bar
```

will produce an error but the equivalent form:

```
subjectAltName=@subject_alt_section

[subject_alt_section]
subjectAltName=URI:ldap://somehost.com/CN=foo,OU=bar
```

is valid.

Due to the behaviour of the OpenSSL **conf** library the same field name can only occur once in a section. This means that:

```
subjectAltName=@alt_section

[alt_section]

email=steve@here
email=steve@there
```

will only recognize the last value. This can be worked around by using the form:

```
[alt_section]

email.1=steve@here
email.2=steve@there
```

HISTORY

The X509v3 extension code was first added to OpenSSL 0.9.2.

Policy mappings, inhibit any policy and name constraints support was added in OpenSSL 0.9.8

The **directoryName** and **otherName** option as well as the **ASN1** option for arbitrary extensions was added in OpenSSL 0.9.8

SEE ALSO

openssl_req (1), *openssl_ca* (1), *openssl_x509* (1), *ASN1_generate_nconf* (3)

NAME

otp — manages one-time passwords

SYNOPSIS

otp [**-dhlor**] [**-f** *algorithm*] [**-u** *user*] *sequence-number seed*

DESCRIPTION

The **otp** program initializes and updates your current series of one-time passwords (OTPs).

Use this to set a new series of one-time passwords. Only perform this on the console or over an encrypted link as you will have to supply your pass-phrase. The other two parameters are *sequence-number* and *seed*.

Options are:

-d To delete a one-time password.

-f Choose a different *algorithm* from the default md5. Pick any of: md4, md5, and sha.

-h For getting a help message.

-l List the current table of one-time passwords.

-o To open (unlock) the otp-entry for a user.

-r To renew a one-time password series. This operation can be performed over an potentially eavesdropped link because you do not supply the pass-phrase. First you need to supply the current one-time password and then the new one corresponding to the supplied *sequence-number* and *seed*.

-u To choose a different *user* to set one-time passwords for. This only works when running **otp** as root.

SEE ALSO

otpprint(1)

NAME

otpprint — print lists of one-time passwords

SYNOPSIS

otp [**-n** *count*] [**-e**] [**-h**] [**-f** *algorithm*] *sequence-number seed*

DESCRIPTION

The **otpprint** program prints lists of OTPs.

Use this to print out a series of one-time passwords. You will have to supply the *sequence number* and the *seed* as arguments and then the program will prompt you for your pass-phrase.

There are several different print formats. The default is to print each password with six short english words.

Options are:

-e Print the passwords in ‘‘extended’’ format. In this format a prefix that says ‘‘hex:’’ or ‘‘word:’’ is included.

-f To choose a different *algorithm* from the default md5. Pick any of: md4, md5, and sha.

-h Print the passwords in hex.

-n Print *count* one-time passwords, starting at *sequence-number* and going backwards. The default is 10.

SEE ALSO

otp(1)

NAME

pagesize — print system page size

SYNOPSIS

pagesize

DESCRIPTION

pagesize prints the size of a page of memory in bytes, as returned by getpagesize(3). This program is useful in constructing portable shell scripts.

SEE ALSO

getpagesize(3)

HISTORY

The **pagesize** command appeared in 4.2BSD.

NAME

pagsh — creates a new credential cache sandbox

SYNOPSIS

pagsh [**-c**] [**-h** | **--help**] [**--version**] [**--cache-type=***string*] *command [args...]*

DESCRIPTION

Supported options:

-c

--cache-type=*string*

-h, **--help**

--version

pagsh creates a new credential cache sandbox for the user to live in. If AFS is installed on the computer, the user is put in a newly created PAG.

For Kerberos 5, the credential cache type that is used is the same as the credential cache type that was used at the time of **pagsh** invocation. The credential cache type can be controlled by the option **--cache-type**.

EXAMPLES

Create a new sandbox where new credentials can be used, while the old credentials can be used by other processes.

```
$ klist
Credentials cache: FILE:/tmp/krb5cc_913
        Principal: lha@E.KTH.SE

  Issued           Expires          Principal
Feb 12 10:08:31  Feb 12 20:06:36  krbtgt/E.KTH.SE@E.KTH.SE
$ pagsh
$ klist
klist: No ticket file: /tmp/krb5cc_03014a
```

SEE ALSO

afslog(1)

NAME

palette — manipulate the text screen colormap

SYNOPSIS

palette [*red green blue* [*code*]]

DESCRIPTION

The **palette** utility manipulates the colormap of x68k's console. The value of the color cell number *code* is replaced according to the arguments *red, green, blue* (each value is an integer from 0 to 31).

Each part of the console corresponds to the *code*:

background	0
cursor	1
foreground (default)	7

If no arguments are supplied, **palette** resets the colormap to the default.

BUGS

Quite a few.

NAME

passwd — modify a user's password

SYNOPSIS

passwd [*user*]
passwd [**-d** *files* | **-l**] [*user*]
passwd [**-d** *nis* | **-y**] [*user*]
passwd [**-d** *krb5* | **-k**] [*principal*]

DESCRIPTION

passwd changes the user's password. First, the user is prompted for their current password. If the current password is correctly typed, a new password is requested. The new password must be entered twice to avoid typing errors.

The new password should be at least six characters long and not purely alphabetic. Its total length must be less than _PASSWORD_LEN (currently 128 characters). Numbers, upper case letters and meta characters are encouraged.

All options may not be available on all systems.

-d *database*
: This option specifies the password database that should be updated. The following databases are supported:

  files
  : This specifies that the password change should be applied to the local password file. When changing only the local password, **passwd** uses pwd_mkdb(8) to update the password databases.

  nis
  : This specifies that the password change should be applied to the NIS password database. The rpc.yppasswdd(8) daemon should be running on the master NIS server.

  krb5
  : This specifies that the user's Kerberos 5 password should be changed. The host must be configured to use Kerberos. See krb5.conf(5).

-l This is the equivalent of **-d** *files*.

-y This is the equivalent of **-d** *nis*.

-k This is the equivalent of **-d** *krb5*.

If a password database is not specified, **passwd** will change the password database as determined by the Pluggable Authentication Module (PAM) library.

The type of cipher used to encrypt the password depends on the configuration in passwd.conf(5). It can be different for local and NIS passwords.

FILES

/etc/master.passwd	The user database
/etc/passwd	A Version 7 format password file
/etc/passwd.XXXXXX	Temporary copy of the password file

SEE ALSO

chpass(1), login(1), pwhash(1), passwd(5), passwd.conf(5), pam(8), pwd_mkdb(8), vipw(8)

Robert Morris and Ken Thompson, *UNIX password security*.

HISTORY

A **passwd** command appeared in Version 6 AT&T UNIX.

NAME

paste — merge corresponding or subsequent lines of files

SYNOPSIS

paste [**-s**] [**-d** *list*] *file ...*

DESCRIPTION

The **paste** utility concatenates the corresponding lines of the given input files, replacing all but the last file's newline characters with a single tab character, and writes the resulting lines to standard output. If end-of-file is reached on an input file while other input files still contain data, the file is treated as if it were an endless source of empty lines.

The options are as follows:

-d *list* Use one or more of the provided characters to replace the newline characters instead of the default tab. The characters in *list* are used circularly, i.e., when *list* is exhausted the first character from *list* is reused. This continues until a line from the last input file (in default operation) or the last line in each file (using the -s option) is displayed, at which time **paste** begins selecting characters from the beginning of *list* again.

The following special characters can also be used in list:

`\n`	newline character
`\t`	tab character
`\\`	backslash character
`\0`	Empty string (not a null character).

Any other character preceded by a backslash is equivalent to the character itself.

-s Concatenate all of the lines of each separate input file in command line order. The newline character of every line except the last line in each input file is replaced with the tab character, unless otherwise specified by the -d option.

If '**-**' is specified for one or more of the input files, the standard input is used; standard input is read one line at a time, circularly, for each instance of '**-**'.

The **paste** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

`cut`(1)

STANDARDS

The **paste** utility is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

NAME

patch - apply a diff file to an original

SYNOPSIS

patch [options] [origfile [patchfile]] [+ [options] [origfile]]...

but usually just

patch <patchfile

DESCRIPTION

Patch will take a patch file containing any of the four forms of difference listing produced by the *diff* program and apply those differences to an original file, producing a patched version. By default, the patched version is put in place of the original, with the original file backed up to the same name with the extension ‘‘.orig’’ (‘‘~’’ on systems that do not support long filenames), or as specified by the **-b**, **-B**, or **-V** switches. The extension used for making backup files may also be specified in the **SIMPLE_BACKUP_SUFFIX** environment variable, which is overridden by above switches.

If the backup file already exists, **patch** creates a new backup file name by changing the first lowercase letter in the last component of the file’s name into uppercase. If there are no more lowercase letters in the name, it removes the first character from the name. It repeats this process until it comes up with a backup file that does not already exist.

You may also specify where you want the output to go with a **-o** switch; if that file already exists, it is backed up first.

If no *patchfile* argument is specified using the **-i** option, and the *patchfile* argument is omitted, or is a hyphen, the patch will be read from standard input.

Upon startup, patch will attempt to determine the type of the diff listing, unless over-ruled by a **-c**, **-e**, **-n**, or **-u** switch. Context diffs (old-style, new-style, and unified) and normal diffs are applied by the *patch* program itself, while ed diffs are simply fed to the *ed* editor via a pipe.

Patch will try to skip any leading garbage, apply the diff, and then skip any trailing garbage. Thus you could feed an article or message containing a diff listing to *patch*, and it should work. If the entire diff is indented by a consistent amount, this will be taken into account.

With context diffs, and to a lesser extent with normal diffs, *patch* can detect when the line numbers mentioned in the patch are incorrect, and will attempt to find the correct place to apply each hunk of the patch. As a first guess, it takes the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that is not the correct place, *patch* will scan both forwards and backwards for a set of lines matching the context given in the hunk. First *patch* looks for a place where all lines of the context match. If no such place is found, and it’s a context diff, and the maximum fuzz factor is set to 1 or more, then another scan takes place ignoring the first and last line of context. If that fails, and the maximum fuzz factor is set to 2 or more, the first two and last two lines of context are ignored, and another scan is made. (The default maximum fuzz factor is 2.) If *patch* cannot find a place to install that hunk of the patch, it will put the hunk out to a reject file, which normally is the name of the output file plus ‘‘.rej’’ (‘‘#’’ on systems that do not support long filenames). (Note that the rejected hunk will come out in context diff form whether the input patch was a context diff or a normal diff. If the input was a normal diff, many of the contexts will simply be null.) The line numbers on the hunks in the reject file may be different than in the patch file: they reflect the approximate location patch thinks the failed hunks belong in the new file rather than the old one.

As each hunk is completed, you will be told whether the hunk succeeded or failed, and which line (in the new file) *patch* thought the hunk should go on. If this is different from the line number specified in the diff you will be told the offset. A single large offset MAY be an indication that a hunk was installed in the wrong place. You will also be told if a fuzz factor was used to make the match, in which case you should also be slightly suspicious.

If no original file is specified on the command line, *patch* will try to figure out from the leading garbage what the name of the file to edit is. In the header of a context diff, the filename is found from lines

beginning with ‘‘***’’ or ‘‘---’’, with the shortest name of an existing file winning. Only context diffs have lines like that, but if there is an ‘‘Index:’’ line in the leading garbage, *patch* will try to use the filename from that line. The context diff header takes precedence over an Index line. If no filename can be intuited from the leading garbage, you will be asked for the name of the file to patch.

If the original file cannot be found or is read-only, but a suitable SCCS or RCS file is handy, *patch* will attempt to get or check out the file.

Additionally, if the leading garbage contains a ‘‘Prereq: ’’ line, *patch* will take the first word from the prerequisites line (normally a version number) and check the input file to see if that word can be found. If not, *patch* will ask for confirmation before proceeding.

The upshot of all this is that you should be able to say, while in a news interface, the following:

```
| patch -d /usr/src/local/blurfl
```

and patch a file in the blurfl directory directly from the article containing the patch.

If the patch file contains more than one patch, *patch* will try to apply each of them as if they came from separate patch files. This means, among other things, that it is assumed that the name of the file to patch must be determined for each diff listing, and that the garbage before each diff listing will be examined for interesting things such as filenames and revision level, as mentioned previously. You can give switches (and another original file name) for the second and subsequent patches by separating the corresponding argument lists by a ‘+’. (The argument list for a second or subsequent patch may not specify a new patch file, however.)

Patch recognizes the following switches:

−B or −−prefix
causes the next argument to be interpreted as a prefix to the backup file name. If this argument is specified any argument from -b will be ignored.

−b or −−suffix
causes the next argument to be interpreted as the backup extension, to be used in place of ‘‘.orig’’ or ‘‘~’’.

−C or −−dry−run
causes *patch* to report what would be done, but not to actually modify any files or create any rejects.

−c or −−context
forces *patch* to interpret the patch file as a context diff.

−D or −−ifdef
causes *patch* to use the "#ifdef...#endif" construct to mark changes. The argument following will be used as the differentiating symbol. Note that, unlike the C compiler, there must be a space between the **−D** and the argument.

−d or −−directory
causes *patch* to interpret the next argument as a directory, and cd to it before doing anything else.

−E or −−remove-empty-files
causes *patch* to remove output files that are empty after the patches have been applied.

−e or −−ed
forces *patch* to interpret the patch file as an ed script.

−F<number> or −−fuzz <number>
sets the maximum fuzz factor. This switch only applies to context diffs, and causes *patch* to ignore up to that many lines in looking for places to install a hunk. Note that a larger fuzz factor increases the odds of a faulty patch. The default fuzz factor is 2, and it may not be set to more than the number of lines of context in the context diff, ordinarily 3.

−f or −−force

forces *patch* to assume that the user knows exactly what he or she is doing, and to not ask any questions. It assumes the following: skip patches for which a file to patch can't be found; patch files even though they have the wrong version for the ''Prereq:'' line in the patch; and assume that patches are not reversed even if they look like they are. This option does not suppress commentary; use **−s** for that.

−i <patchfile> or −−patchfile <patchfile>

Read the patch to be applied from the specified file.

−l or −−ignore-whitespace

causes the pattern matching to be done loosely, in case the tabs and spaces have been munged in your input file. Any sequence of whitespace in the pattern line will match any sequence in the input file. Normal characters must still match exactly. Each line of the context must still match a line in the input file.

−N or −−forward

causes *patch* to ignore patches that it thinks are reversed or already applied. See also **−R .**

−n or −−normal

forces *patch* to interpret the patch file as a normal diff.

−o or −−output

causes the next argument to be interpreted as the output file name.

−p<number> or −−strip <number>

sets the pathname strip count, which controls how pathnames found in the patch file are treated, in case you keep your files in a different directory than the person who sent out the patch. The strip count specifies how many slashes are to be stripped from the front of the pathname. (Any intervening directory names also go away.) For example, supposing the filename in the patch file was

/u/howard/src/blurfl/blurfl.c

setting **−p** or **−p0** gives the entire pathname unmodified, **−p1** gives

u/howard/src/blurfl/blurfl.c

without the leading slash, **−p4** gives

blurfl/blurfl.c

and not specifying **−p** at all just gives you "blurfl.c", unless all of the directories in the leading path (u/howard/src/blurfl) exist and that path is relative, in which case you get the entire pathname unmodified. Whatever you end up with is looked for either in the current directory, or the directory specified by the **−d** switch.

−R or −−reverse

tells *patch* that this patch was created with the old and new files swapped. (Yes, I'm afraid that does happen occasionally, human nature being what it is.) *Patch* will attempt to swap each hunk around before applying it. Rejects will come out in the swapped format. The **−R** switch will not work with ed diff scripts because there is too little information to reconstruct the reverse operation.

If the first hunk of a patch fails, *patch* will reverse the hunk to see if it can be applied that way. If it can, you will be asked if you want to have the **−R** switch set. If it can't, the patch will continue to be applied normally. (Note: this method cannot detect a reversed patch if it is a normal diff and if the first command is an append (i.e. it should have been a delete) since appends always succeed, due to the fact that a null context will match anywhere. Luckily, most patches add or change lines rather than delete them, so most reversed normal diffs will begin with a delete, which will fail, triggering the heuristic.)

−r or −−reject-file
causes the next argument to be interpreted as the reject file name.

−S or −−skip
causes *patch* to ignore this patch from the patch file, but continue on looking for the next patch in the file. Thus

```
patch -S + -S + <patchfile
```

will ignore the first and second of three patches.

−s or −−quiet or −−silent
makes *patch* do its work silently, unless an error occurs.

−t or −−batch
similar to **−f**, in that it suppresses questions, but makes some different assumptions: skip patches for which a file to patch can't be found (the same as **−f**); skip patches for which the file has the wrong version for the ‘‘Prereq:’’ line in the patch; and assume that patches are reversed if they look like they are.

−u or −−unified
forces *patch* to interpret the patch file as a unified context diff (a unidiff).

−V or −−version-control
causes the next argument to be interpreted as a method for creating backup file names. The type of backups made can also be given in the **VERSION_CONTROL** environment variable, which is overridden by this option. The **-B** option overrides this option, causing the prefix to always be used for making backup file names. The value of the **VERSION_CONTROL** environment variable and the argument to the **-V** option are like the GNU Emacs ‘version-control’ variable; they also recognize synonyms that are more descriptive. The valid values are (unique abbreviations are accepted):

‘t’ or ‘numbered’
Always make numbered backups.

‘nil’ or ‘existing’
Make numbered backups of files that already have them, simple backups of the others. This is the default.

‘never’ or ‘simple’
Always make simple backups.

−v or −−version
causes *patch* to print out its revision header and patch level.

−x<number> or −−debug <number>
sets internal debugging flags, and is of interest only to *patch* patchers.

AUTHOR

Larry Wall <lwall@netlabs.com>
with many other contributors.

ENVIRONMENT

TMPDIR
Directory to put temporary files in; default is /tmp.

SIMPLE_BACKUP_SUFFIX
Extension to use for backup file names instead of ‘‘.orig’’ or ‘‘~’’.

VERSION_CONTROL
Selects when numbered backup files are made.

FILES

$TMPDIR/patch*

SEE ALSO

diff(1)

NOTES FOR PATCH SENDERS

There are several things you should bear in mind if you are going to be sending out patches. First, you can save people a lot of grief by keeping a patchlevel.h file which is patched to increment the patch level as the first diff in the patch file you send out. If you put a Prereq: line in with the patch, it won't let them apply patches out of order without some warning. Second, make sure you've specified the filenames right, either in a context diff header, or with an Index: line. If you are patching something in a subdirectory, be sure to tell the patch user to specify a **−p** switch as needed. Third, you can create a file by sending out a diff that compares a null file to the file you want to create. This will only work if the file you want to create doesn't exist already in the target directory. Fourth, take care not to send out reversed patches, since it makes people wonder whether they already applied the patch. Fifth, while you may be able to get away with putting 582 diff listings into one file, it is probably wiser to group related patches into separate files in case something goes haywire.

DIAGNOSTICS

Too many to list here, but generally indicative that *patch* couldn't parse your patch file.

The message ''Hmm...'' indicates that there is unprocessed text in the patch file and that *patch* is attempting to intuit whether there is a patch in that text and, if so, what kind of patch it is.

Patch will exit with a non-zero status if any reject files were created. When applying a set of patches in a loop it behooves you to check this exit status so you don't apply a later patch to a partially patched file.

CAVEATS

Patch cannot tell if the line numbers are off in an ed script, and can only detect bad line numbers in a normal diff when it finds a ''change'' or a ''delete'' command. A context diff using fuzz factor 3 may have the same problem. Until a suitable interactive interface is added, you should probably do a context diff in these cases to see if the changes made sense. Of course, compiling without errors is a pretty good indication that the patch worked, but not always.

Patch usually produces the correct results, even when it has to do a lot of guessing. However, the results are guaranteed to be correct only when the patch is applied to exactly the same version of the file that the patch was generated from.

BUGS

Could be smarter about partial matches, excessively deviant offsets and swapped code, but that would take an extra pass.

If code has been duplicated (for instance with #ifdef OLDCODE ... #else ... #endif), *patch* is incapable of patching both versions, and, if it works at all, will likely patch the wrong one, and tell you that it succeeded to boot.

If you apply a patch you've already applied, *patch* will think it is a reversed patch, and offer to un-apply the patch. This could be construed as a feature.

NAME

pathchk — check pathnames

SYNOPSIS

pathchk [**-p**] *pathname ...*

DESCRIPTION

The **pathchk** utility checks whether each of the specified *pathname* arguments is valid or portable.

A diagnostic message is written for each argument that:

- Is longer than PATH_MAX bytes.
- Contains any component longer than NAME_MAX bytes. (The value of NAME_MAX depends on the underlying file system.)
- Contains a directory component that is not searchable.

It is not considered an error if a *pathname* argument contains a nonexistent component as long as a component by that name could be created.

The options are as follows:

-p Perform portability checks on the specified *pathname* arguments. Diagnostic messages will be written for each argument that:

- Is longer than _POSIX_PATH_MAX (255) bytes.
- Contains a component longer than _POSIX_NAME_MAX (14) bytes.
- Contains any character not in the portable filename character set (that is, alphanumeric characters, '.', '-' and '_'). No component may start with the hyphen ('-') character.

EXAMPLES

Check whether the names of files in the current directory are portable to other POSIX systems:

```
find . -print | xargs pathchk -p
```

SEE ALSO

getconf(1), pathconf(2), stat(2)

STANDARDS

The **pathchk** utility conforms to IEEE Std 1003.1-2001 ("POSIX.1").

HISTORY

A **pathchk** utility appeared in NetBSD 2.0.

NAME

pawd − print automounter working directory

SYNOPSIS

pawd [*path ...*]

DESCRIPTION

pawd is used to print the current working directory, adjusted to reflect proper paths that can be reused to go through the automounter for the shortest possible path. In particular, the path printed back does not include any of **Amd**'s local mount points. Using them is unsafe, because **Amd** may unmount managed file systems from the mount points, and thus including them in paths may not always find the files within.

Without any arguments, **pawd** will print the automounter adjusted current working directory. With any number of arguments, it will print the adjusted path of each one of the arguments.

SEE ALSO

pwd(1), **amd**(8), **amq**(8), **pwd**(1).

''am-utils'' **info**(1) entry.

Linux NFS and Automounter Administration by Erez Zadok, ISBN 0-7821-2739-8, (Sybex, 2001).

http://www.am-utils.org

Amd − The 4.4 BSD Automounter

AUTHORS

Erez Zadok <ezk@cs.sunysb.edu>, Computer Science Department, Stony Brook University, Stony Brook, New York, USA.

Other authors and contributors to am-utils are listed in the **AUTHORS** file distributed with am-utils.

NAME

pax — read and write file archives and copy directory hierarchies

SYNOPSIS

pax [**-0cdjnOVvz**] [**-E** *limit*] [**-f** *archive*] [**-N** *dbdir*] [**-s** *replstr*] ... [**-U** *user*] ... [**-G** *group*] ... [**-T** [*from_date*][,*to_date*]] ... [*pattern ...*]
pax -r [**-AcDdijknOuVvYZz**] [**-E** *limit*] [**-f** *archive*] [**-N** *dbdir*] [**-o** *options*] ... [**-p** *string*] ... [**-s** *replstr*] ... [**-U** *user*] ... [**-G** *group*] ... [**-T** [*from_date*][,*to_date*]] ... [*pattern ...*]
pax -w [**-AdHijLMOPtuVvXz**] [**-b** *blocksize*] [[**-a**] [**-f** *archive*]] [**-x** *format*] [**-B** *bytes*] [**-N** *dbdir*] [**-o** *options*] ... [**-s** *replstr*] ... [**-U** *user*] ... [**-G** *group*] ... [**-T** [*from_date*][,*to_date*][/[**c**][**m**]]] ... [*file ...*]
pax -r -w [**-ADdHijkLlMnOPtuVvXYZz**] [**-N** *dbdir*] [**-p** *string*] ... [**-s** *replstr*] ... [**-U** *user*] ... [**-G** *group*] ... [**-T** [*from_date*][,*to_date*][/[**c**][**m**]]] ... [*file ...*] *directory*

DESCRIPTION

pax will read, write, and list the members of an archive file, and will copy directory hierarchies. If the archive file is of the form: *[[user@]host:]file* then the archive will be processed using `rmt`(8).

pax operation is independent of the specific archive format, and supports a wide variety of different archive formats. A list of supported archive formats can be found under the description of the **-x** option.

The presence of the **-r** and the **-w** options specifies which of the following functional modes **pax** will operate under: *list*, *read*, *write*, and *copy*.

⟨none⟩ *List*. **pax** will write to `standard output` a table of contents of the members of the archive file read from `standard input`, whose pathnames match the specified *patterns*. The table of contents contains one filename per line and is written using single line buffering.

-r *Read*. **pax** extracts the members of the archive file read from the `standard input`, with pathnames matching the specified *patterns*. The archive format and blocking is automatically determined on input. When an extracted file is a directory, the entire file hierarchy rooted at that directory is extracted. All extracted files are created relative to the current file hierarchy. The setting of ownership, access and modification times, and file mode of the extracted files are discussed in more detail under the **-p** option.

-w *Write*. **pax** writes an archive containing the *file* operands to `standard output` using the specified archive format. When no *file* operands are specified, a list of files to copy with one per line is read from `standard input`. When a *file* operand is also a directory, the entire file hierarchy rooted at that directory will be included.

-r -w

Copy. **pax** copies the *file* operands to the destination *directory*. When no *file* operands are specified, a list of files to copy with one per line is read from the `standard input`. When a *file* operand is also a directory the entire file hierarchy rooted at that directory will be included. The effect of the *copy* is as if the copied files were written to an archive file and then subsequently extracted, except that there may be hard links between the original and the copied files (see the **-l** option below).

Warning: The destination *directory* must not be one of the *file* operands or a member of a file hierarchy rooted at one of the *file* operands. The result of a *copy* under these conditions is unpredictable.

While processing a damaged archive during a *read* or *list* operation, **pax** will attempt to recover from media defects and will search through the archive to locate and process the largest number of archive members possible (see the **-E** option for more details on error handling).

OPERANDS

The *directory* operand specifies a destination directory pathname. If the *directory* operand does not exist, or it is not writable by the user, or it is not of type directory, **pax** will exit with a non-zero exit status.

The *pattern* operand is used to select one or more pathnames of archive members. Archive members are selected using the pattern matching notation described by fnmatch(3). When the *pattern* operand is not supplied, all members of the archive will be selected. When a *pattern* matches a directory, the entire file hierarchy rooted at that directory will be selected. When a *pattern* operand does not select at least one archive member, **pax** will write these *pattern* operands in a diagnostic message to standard error and then exit with a non-zero exit status.

The *file* operand specifies the pathname of a file to be copied or archived. When a *file* operand does not select at least one archive member, **pax** will write these *file* operand pathnames in a diagnostic message to standard error and then exit with a non-zero exit status.

OPTIONS

The following options are supported:

-r Read an archive file from standard input and extract the specified *files*. If any intermediate directories are needed in order to extract an archive member, these directories will be created as if mkdir(2) was called with the bitwise inclusive OR of S_IRWXU, S_IRWXG, and S_IRWXO as the mode argument. When the selected archive format supports the specification of linked files and these files cannot be linked while the archive is being extracted, **pax** will write a diagnostic message to standard error and exit with a non-zero exit status at the completion of operation.

-w Write files to the standard output in the specified archive format. When no *file* operands are specified, standard input is read for a list of pathnames with one per line without any leading or trailing ⟨blanks⟩.

-a Append *files* to the end of an archive that was previously written. If an archive format is not specified with a **-x** option, the format currently being used in the archive will be selected. Any attempt to append to an archive in a format different from the format already used in the archive will cause **pax** to exit immediately with a non-zero exit status. The blocking size used in the archive volume where writing starts will continue to be used for the remainder of that archive volume.

Warning: Many storage devices are not able to support the operations necessary to perform an append operation. Any attempt to append to an archive stored on such a device may damage the archive or have other unpredictable results. Tape drives in particular are more likely to not support an append operation. An archive stored in a regular file system file or on a disk device will usually support an append operation.

-b *blocksize*

When *writing* an archive, block the output at a positive decimal integer number of bytes per write to the archive file. The *blocksize* must be a multiple of 512 bytes with a maximum of 32256 bytes. A *blocksize* can end with k or b to specify multiplication by 1024 (1K) or 512, respectively. A pair of *blocksizes* can be separated by x to indicate a product. A specific archive device may impose additional restrictions on the size of blocking it will support. When blocking is not specified, the default *blocksize* is dependent on the specific archive format being used (see the **-x** option).

-c Match all file or archive members *except* those specified by the *pattern* and *file* operands.

-d Cause files of type directory being copied or archived, or archive members of type directory being extracted, to match only the directory file or archive member and not the file hierarchy rooted at the directory.

-f *archive*
Specify *archive* as the pathname of the input or output archive, overriding the default `standard input` (for *list* and *read*) or `standard output` (for *write*). A single archive may span multiple files and different archive devices. When required, **pax** will prompt for the pathname of the file or device of the next volume in the archive.

-i Interactively rename files or archive members. For each archive member matching a *pattern* operand or each file matching a *file* operand, **pax** will prompt to `/dev/tty` giving the name of the file, its file mode and its modification time. **pax** will then read a line from `/dev/tty`. If this line is blank, the file or archive member is skipped. If this line consists of a single period, the file or archive member is processed with no modification to its name. Otherwise, its name is replaced with the contents of the line. **pax** will immediately exit with a non-zero exit status if ⟨EOF⟩ is encountered when reading a response or if `/dev/tty` cannot be opened for reading and writing.

-j Use `bzip2`(1) for compression when reading or writing archive files.

-k Do not overwrite existing files.

-l Link files. (The letter ell). In the *copy* mode (**-r -w**), hard links are made between the source and destination file hierarchies whenever possible.

-n Select the first archive member that matches each *pattern* operand. No more than one archive member is matched for each *pattern*. When members of type directory are matched, the file hierarchy rooted at that directory is also matched (unless **-d** is also specified).

-o *options*
Information to modify the algorithm for extracting or writing archive files which is specific to the archive format specified by **-x**. In general, *options* take the form: **name=value**

-p *string*
Specify one or more file characteristic options (privileges). The *string* option-argument is a string specifying file characteristics to be retained or discarded on extraction. The string consists of the specification characters **a**, **e**, **m**, **o**, and **p**. Multiple characteristics can be concatenated within the same string and multiple **-p** options can be specified. The meaning of the specification characters are as follows:

a Do not preserve file access times. By default, file access times are preserved whenever possible.

e 'Preserve everything', the user ID, group ID, file mode bits, file access time, and file modification time. This is intended to be used by *root*, someone with all the appropriate privileges, in order to preserve all aspects of the files as they are recorded in the archive. The **e** flag is the sum of the **o** and **p** flags.

m Do not preserve file modification times. By default, file modification times are preserved whenever possible.

o Preserve the user ID and group ID.

p 'Preserve' the file mode bits. This is intended to be used by a *user* with regular privileges who wants to preserve all aspects of the file other than the ownership. The file times are preserved by default, but two other flags are offered to disable this and use the time of extraction instead.

In the preceding list, 'preserve' indicates that an attribute stored in the archive is given to the extracted file, subject to the permissions of the invoking process. Otherwise the attribute of the extracted file is determined as part of the normal file creation action. If neither the **e** nor the **o** specification character

is specified, or the user ID and group ID are not preserved for any reason, **pax** will not set the S_ISUID (*setuid*) and S_ISGID (*setgid*) bits of the file mode. If the preservation of any of these items fails for any reason, **pax** will write a diagnostic message to standard error. Failure to preserve these items will affect the final exit status, but will not cause the extracted file to be deleted. If the file characteristic letters in any of the string option-arguments are duplicated or conflict with each other, the one(s) given last will take precedence. For example, if

-p *eme*

is specified, file modification times are still preserved.

-s *replstr*
Modify the file or archive member names specified by the *pattern* or *file* operands according to the substitution expression *replstr*, using the syntax of the ed(1) utility regular expressions. The format of these regular expressions are:

/old/new/[gp]

As in ed(1), **old** is a basic regular expression and **new** can contain an ampersand (&), \n (where n is a digit) back-references, or subexpression matching. The **old** string may also contain ⟨newline⟩ characters. Any non-null character can be used as a delimiter (/ is shown here). Multiple **-s** expressions can be specified. The expressions are applied in the order they are specified on the command line, terminating with the first successful substitution. The optional trailing **g** continues to apply the substitution expression to the pathname substring which starts with the first character following the end of the last successful substitution. The first unsuccessful substitution stops the operation of the **g** option. The optional trailing **p** will cause the final result of a successful substitution to be written to standard error in the following format:

⟨original pathname⟩ >> ⟨new pathname⟩

File or archive member names that substitute to the empty string are not selected and will be skipped.

-t Reset the access times of any file or directory read or accessed by **pax** to be the same as they were before being read or accessed by **pax**, if the user has the appropriate permissions required by utime(3).

-u Ignore files that are older (having a less recent file modification time) than a pre-existing file or archive member with the same name. During *read*, an archive member with the same name as a file in the file system will be extracted if the archive member is newer than the file. During *write*, a file system member with the same name as an archive member will be written to the archive if it is newer than the archive member. During *copy*, the file in the destination hierarchy is replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.

-v During a *list* operation, produce a verbose table of contents using the format of the ls(1) utility with the **-l** option. For pathnames representing a hard link to a previous member of the archive, the output has the format:

⟨ls -l listing⟩ == ⟨link name⟩

Where ⟨ls -l listing⟩ is the output format specified by the ls(1) utility when used with the **-l** option.

Otherwise for all the other operational modes (*read*, *write*, and *copy*), pathnames are written and flushed to standard error without a trailing ⟨newline⟩ as soon as processing begins on that file or archive member. The trailing ⟨newline⟩, is not buffered, and is written only after the file has been read or written.

A final summary of archive operations is printed after they have been completed.

-x *format*
Specify the output archive format, with the default format being *ustar*. **pax** currently supports the following formats:

cpio The extended cpio interchange format specified in the IEEE Std 1003.2 ("POSIX.2") standard. The default blocksize for this format is 5120 bytes. Inode and device information about a file (used for detecting file hard links by this format) which may be truncated by this format is detected by **pax** and is repaired.

bcpio The old binary cpio format. The default blocksize for this format is 5120 bytes. This format is not very portable and should not be used when other formats are available. Inode and device information about a file (used for detecting file hard links by this format) which may be truncated by this format is detected by **pax** and is repaired.

sv4cpio The AT&T System V.4 UNIX cpio. The default blocksize for this format is 5120 bytes. Inode and device information about a file (used for detecting file hard links by this format) which may be truncated by this format is detected by **pax** and is repaired.

sv4crc The AT&T System V.4 UNIX cpio with file crc checksums. The default blocksize for this format is 5120 bytes. Inode and device information about a file (used for detecting file hard links by this format) which may be truncated by this format is detected by **pax** and is repaired.

tar The old BSD tar format as found in 4.3BSD. The default blocksize for this format is 10240 bytes. Pathnames stored by this format must be 100 characters or less in length. Only *regular* files, *hard links*, *soft links*, and *directories* will be archived (other file types are not supported). For backward compatibility with even older tar formats, a **-o** option can be used when writing an archive to omit the storage of directories. This option takes the form:

-o write_opt=nodir

ustar The extended tar interchange format specified in the IEEE Std 1003.2 ("POSIX.2") standard. The default blocksize for this format is 10240 bytes. Pathnames stored by this format must be 250 characters or less in length.

pax will detect and report any file that it is unable to store or extract as the result of any specific archive format restrictions. The individual archive formats may impose additional restrictions on use. Typical archive format restrictions include (but are not limited to): file pathname length, file size, link pathname length and the type of the file.

-z Use `gzip`(1) compression, when reading or writing archive files.

-A Do not strip leading '/'s from file names.

-B *bytes*

Limit the number of bytes written to a single archive volume to *bytes*. The *bytes* limit can end with `m`, `k`, or `b` to specify multiplication by 1048576 (1M), 1024 (1K) or 512, respectively. A pair of *bytes* limits can be separated by `x` to indicate a product.

Warning: Only use this option when writing an archive to a device which supports an end of file read condition based on last (or largest) write offset (such as a regular file or a tape drive). The use of this option with a floppy or hard disk is not recommended.

-D This option is the same as the **-u** option, except that the file inode change time is checked instead of the file modification time. The file inode change time can be used to select files whose inode information (e.g. uid, gid, etc.) is newer than a copy of the file in the destination *directory*.

-E *limit*

Limit the number of consecutive read faults while trying to read a flawed archives to *limit*. With a positive *limit*, **pax** will attempt to recover from an archive read error and will continue processing starting with the next file stored in the archive. A *limit* of 0 will cause **pax** to stop operation after the first read error is detected on an archive volume. A *limit* of `NONE` will cause **pax** to attempt to recover from read errors forever. The default *limit* is a small positive number of retries.

Warning: Using this option with `NONE` should be used with extreme caution as **pax** may get stuck in an infinite loop on a very badly flawed archive.

-G *group*
Select a file based on its *group* name, or when starting with a **#**, a numeric gid. A '\' can be used to escape the **#**. Multiple **-G** options may be supplied and checking stops with the first match.

-H Follow only command line symbolic links while performing a physical file system traversal.

-L Follow all symbolic links to perform a logical file system traversal.

-M During a *write* or *copy* operation, treat the list of files on `standard input` as an `mtree`(8) 'specfile' specification, and write or copy only those items in the specfile.

If the file exists in the underlying file system, its permissions and modification time will be used unless specifically overridden by the specfile. An error will be raised if the type of entry in the specfile conflicts with that of an existing file. A directory entry that is marked '**optional**' will not be copied (even though its contents will be).

Otherwise, the entry will be 'faked-up', and it is necessary to specify at least the following parameters in the specfile: **type**, **mode**, **gname** or **gid**, and **uname** or **uid**, **device** (in the case of block or character devices), and **link** (in the case of symbolic links). If **time** isn't provided, the current time will be used. A 'faked-up' entry that is marked '**optional**' will not be copied.

-N *dbdir*
Except for lookups for the **-G** and **-U** options, use the user database text file `master.passwd` and group database text file `group` from *dbdir*, rather than using the results from the system's `getpwnam`(3) and `getgrnam`(3) (and related) library calls.

-O Force the archive to be one volume. If a volume ends prematurely, **pax** will not prompt for a new volume. This option can be useful for automated tasks where error recovery cannot be performed by a human.

-P Do not follow symbolic links, perform a physical file system traversal. This is the default mode.

-T *[from_date][,to_date][/[c][m]]*
Allow files to be selected based on a file modification or inode change time falling within a specified time range of *from_date* to *to_date* (the dates are inclusive). If only a *from_date* is supplied, all files with a modification or inode change time equal to or younger are selected. If only a *to_date* is supplied, all files with a modification or inode change time equal to or older will be selected. When the *from_date* is equal to the *to_date*, only files with a modification or inode change time of exactly that time will be selected.

When **pax** is in the *write* or *copy* mode, the optional trailing field *[c][m]* can be used to determine which file time (inode change, file modification or both) are used in the comparison. If neither is specified, the default is to use file modification time only. The *m* specifies the comparison of file modification time (the time when the file was last written). The *c* specifies the comparison of inode change time (the time when the file inode was last changed; e.g. a change of owner, group, mode, etc). When *c* and *m* are both specified, then the modification and inode change times are both compared. The inode change time comparison is useful in selecting files whose attributes were recently changed or selecting files which were recently created and had their modification time reset to an older time (as what happens when a file is extracted from an archive and the modification time is preserved). Time comparisons using both file times is useful when **pax** is used to create a time based incremental archive (only files that were changed during a specified time range will be archived).

A time range is made up of six different fields and each field must contain two digits. The format is:

```
[[[[[cc]yy]mm]dd]hh]mm[.ss]
```

Where **cc** is the first two digits of the year (the century), **yy** is the last two digits of the year, the first

mm is the month (from 01 to 12), **dd** is the day of the month (from 01 to 31), **hh** is the hour of the day (from 00 to 23), the second **mm** is the minute (from 00 to 59), and **ss** is the seconds (from 00 to 61). Only the minute field **mm** is required; the others will default to the current system values. The **ss** field may be added independently of the other fields. If the century is not specified, it defaults to 1900 for years between 69 and 99, or 2000 for years between 0 and 68. Time ranges are relative to the current time, so

-T *1234/cm*

would select all files with a modification or inode change time of 12:34 PM today or later. Multiple **-T** time range can be supplied and checking stops with the first match.

-U *user*
Select a file based on its *user* name, or when starting with a **#**, a numeric uid. A '\' can be used to escape the **#**. Multiple **-U** options may be supplied and checking stops with the first match.

-V A final summary of archive operations is printed after they have been completed. Some potentially long-running tape operations are noted.

-X When traversing the file hierarchy specified by a pathname, do not descend into directories that have a different device ID. See the `st_dev` field as described in `stat`(2) for more information about device ID's.

-Y This option is the same as the **-D** option, except that the inode change time is checked using the pathname created after all the file name modifications have completed.

-Z This option is the same as the **-u** option, except that the modification time is checked using the pathname created after all the file name modifications have completed.

-0 Use the nul character instead of \n as the file separator when reading files from standard input.

--force-local
Do not interpret filenames that contain a ':' as remote files.

--insecure
Normally **pax** ignores filenames that contain ".." as a path component. With this option, files that contain ".." can be processed.

The options that operate on the names of files or archive members (**-c**, **-i**, **-n**, **-s**, **-u**, **-v**, **-D**, **-G**, **-T**, **-U**, **-Y**, and **-Z**) interact as follows.

When extracting files during a *read* operation, archive members are 'selected', based only on the user specified pattern operands as modified by the **-c**, **-n**, **-u**, **-D**, **-G**, **-T**, **-U** options. Then any **-s** and **-i** options will modify in that order, the names of these selected files. Then the **-Y** and **-Z** options will be applied based on the final pathname. Finally the **-v** option will write the names resulting from these modifications.

When archiving files during a *write* operation, or copying files during a *copy* operation, archive members are 'selected', based only on the user specified pathnames as modified by the **-n**, **-u**, **-D**, **-G**, **-T**, and **-U** options (the **-D** option only applies during a copy operation). Then any **-s** and **-i** options will modify in that order, the names of these selected files. Then during a *copy* operation the **-Y** and the **-Z** options will be applied based on the final pathname. Finally the **-v** option will write the names resulting from these modifications.

When one or both of the **-u** or **-D** options are specified along with the **-n** option, a file is not considered selected unless it is newer than the file to which it is compared.

EXIT STATUS

pax will exit with one of the following values:

0 All files were processed successfully.

1 An error occurred.

Whenever **pax** cannot create a file or a link when reading an archive or cannot find a file when writing an archive, or cannot preserve the user ID, group ID, or file mode when the **-p** option is specified, a diagnostic message is written to `standard error` and a non-zero exit status will be returned, but processing will continue. In the case where pax cannot create a link to a file, **pax** will not create a second copy of the file.

If the extraction of a file from an archive is prematurely terminated by a signal or error, **pax** may have only partially extracted a file the user wanted. Additionally, the file modes of extracted files and directories may have incorrect file bits, and the modification and access times may be wrong.

If the creation of an archive is prematurely terminated by a signal or error, **pax** may have only partially created the archive which may violate the specific archive format specification.

If while doing a *copy*, **pax** detects a file is about to overwrite itself, the file is not copied, a diagnostic message is written to `standard error` and when **pax** completes it will exit with a non-zero exit status.

EXAMPLES

The command:

```
pax -w -f /dev/rst0 .
```

copies the contents of the current directory to the device `/dev/rst0`.

The command:

```
pax -v -f filename
```

gives the verbose table of contents for an archive stored in `filename`.

The following commands:

```
mkdir newdir
cd olddir
pax -rw -pp . ../newdir
```

will copy the entire `olddir` directory hierarchy to `newdir`, preserving permissions and access times.

When running as root, one may also wish to preserve file ownership when copying directory trees. This can be done with the following commands:

```
cd olddir
pax -rw -pe . .../newdir
```

which will copy the contents of `olddir` into `.../newdir`, preserving ownership, permissions and access times.

The command:

```
pax -r -s ',^//*usr//*,,' -f a.pax
```

reads the archive `a.pax`, with all files rooted in ‘‘/usr’’ into the archive extracted relative to the current directory.

The command:

```
pax -rw -i . dest_dir
```

can be used to interactively select the files to copy from the current directory to `dest_dir`.

The command:

```
pax -r -pe -U root -G bin -f a.pax
```

will extract all files from the archive `a.pax` which are owned by *root* with group *bin* and will preserve all file permissions.

The command:

```
pax -r -w -v -Y -Z home /backup
```

will update (and list) only those files in the destination directory `/backup` which are older (less recent inode change or file modification times) than files with the same name found in the source file tree `home`.

SEE ALSO

`cpio`(1), `tar`(1), `symlink`(7), `mtree`(8)

STANDARDS

The **`pax`** utility is a superset of the IEEE Std 1003.2 ("POSIX.2") standard. The options **`-B`**, **`-D`**, **`-E`**, **`-G`**, **`-H`**, **`-L`**, **`-M`**, **`-O`**, **`-P`**, **`-T`**, **`-U`**, **`-Y`**, **`-Z`**, **`-z`**, the archive formats *`bcpio`*, *`sv4cpio`*, *`sv4crc`*, *`tar`*, and the flawed archive handling during *`list`* and *`read`* operations are extensions to the POSIX standard.

AUTHORS

Keith Muller at the University of California, San Diego. Luke Mewburn implemented **`-M`**.

NAME

pfrom — fetch a list of the current mail via POP

SYNOPSIS

pfrom [**−4** | **−-krb4**][**−5** | **−-krb5**][**−v** | **−-verbose**][**−c** | **−-count**][**−-header**]
[**−p** *port-spec* | **−-port=***port-spec*]

DESCRIPTION

pfrom is a script that does push --from.

SEE ALSO

push(8)

NAME

pkg_add — a utility for installing and upgrading software package distributions

SYNOPSIS

pkg_add [**-AfILnRuVv**] [**-K** *pkg_dbdir*] [**-m** *machine*] [**-p** *prefix*]
[**-s** *verification-type*] [**-t** *template*] [**-W** *viewbase*] [**-w** *view*]
[[ftp|http]://[*user*[:*password]@*]*host*[:*port*]][/*path/*]pkg-name ...

DESCRIPTION

The **pkg_add** command is used to extract and upgrade packages that have been previously created with the pkg_create(1) command. Packages are prepared collections of pre-built binaries, documentation, configurations, installation instructions and/or other files. **pkg_add** can recursively install other packages that the current package depends on or requires from both local disk and via FTP or HTTP.

WARNING

Since the **pkg_add** *command may execute scripts or programs contained within a package file, your system may be susceptible to "Trojan horses" or other subtle attacks from miscreants who create dangerous package files.*

You are advised to verify the competence and identity of those who provide installable package files. For extra protection, use the digital signatures provided where possible (see the **-s** *option), or, failing that, use* tar(*1*) *to extract the package file, and inspect its contents and scripts to ensure it poses no danger to your system's integrity. Pay particular attention to any* +INSTALL *or* +DEINSTALL *files, and inspect the* +CONTENTS *file for* **@cwd**, **@mode** *(check for setuid),* **@dirrm**, **@exec**, *and* **@unexec** *directives, and/or use the* pkg_info(*1*) *command to examine the package file.*

OPTIONS

The following command line arguments are supported:

pkg-name [*...*]
: The named packages are installed. *pkg-name* may be either a URL or a local pathname, a package name of "-" will cause **pkg_add** to read from stdin. If the packages are not found in the current working directory, **pkg_add** will search them in each directory named by the PKG_PATH environment variable. Any dependencies required by the installed package will be searched in the same location that the original package was installed from.

-A
: Mark package as installed automatically, as dependency of another package. You can use
pkg_admin set automatic=YES
to mark packages this way after installation, and
pkg_admin unset automatic
to remove the mark. If you **pkg_add** a package without specifying **-A** after it had already been automatically installed, the mark is removed.

-f
: Force installation to proceed even if prerequisite packages are not installed or the install script fails. Although **pkg_add** will still try to find and auto-install missing prerequisite packages, a failure to find one will not be fatal. This flag also overrides the fatal error when the operating system or architecture the package was built on differ from that of the host.

-I
: If an installation script exists for a given package, do not execute it.

-K *pkg_dbdir*
: Set *pkg_dbdir* as the package database directory. If this option isn't specified, then the package database directory is taken from the value of the environment variable PKG_DBDIR if it's set, otherwise it defaults to /var/db/pkg.

-L Don't add the package to any views after installation.

-m Override the machine architecture returned by uname with *machine*.

-n Don't actually install a package, just report the steps that would be taken if it was.

-p *prefix*
Set *prefix* as the directory in which to extract files from a package. If a package has set its default directory, it will be overridden by this flag. Note that only the first **@cwd** directive will be replaced, since **pkg_add** has no way of knowing which directory settings are relative and which are absolute. Only one directory transition is supported and the second one is expected to go into *pkgdb*.

-R Do not record the installation of a package. This means that you cannot deinstall it later, so only use this option if you know what you are doing!

-s *verification-type*
Use a callout to an external program to verify the binary package being installed against an existing detached signature file. The signature file must reside in the same directory as the binary package. At the present time, the following verification types are defined: none, gpg and pgp5. The signature will be verified at install time, and the results will be displayed. If the signature type is anything other than none, the user will be asked if **pkg_add** should proceed to install the binary package. The user must then take the decision whether to proceed or not, depending upon the amount of trust that is placed in the signatory of the binary package. Please note that, at the current time, it is not possible to use the verification feature when using **pkg_add** to add a binary package via a URL - the package, and the related detached signature file, must be local for the verification to work.

-t *template*
Use *template* as the input to mktemp(3) when creating a "staging area". By default, this is the string /var/tmp/instmp.XXXXXX, but it may be necessary to override it in the situation where space in your /var/tmp directory is limited. Be sure to leave some number of 'X' characters for mktemp(3) to fill in with a unique ID.

You can get a performance boost by setting the staging area *template* to reside on the same disk partition as target directories for package file installation; often this is /usr.

-u If the package that's being installed is already installed, either in the same or a different version, an update is performed. If this is specified twice, then any dependant packages that are too old will also be updated to fulfill the dependency. See below for a more detailed description of the process.

-V Print version number and exit.

-v Turn on verbose output.

-W *viewbase*
Set *viewbase* as the base directory for the managed views. The default *viewbase* directory is set by pkg_view(1). This value also may be set from the LOCALBASE environment variable.

-w *view*
Set the *view* to which packages should be added after installation. The default *view* is set by pkg_view(1). This value also may be set from the PKG_VIEW environment variable.

One or more *pkg-name* arguments may be specified, each being either a file containing the package (these usually ending with the ".tgz" suffix) or a URL pointing at a file available on an ftp or web site. Thus you may extract files directly from their anonymous ftp or WWW locations (e.g., **pkg_add** ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/i386/3.1_2007Q2/shells/bash-3.2.9.tgz or **pkg_add** http://www.example.org/packages/screen-4.0.tbz). Note: For ftp transfers, if you wish to use *passive mode*

ftp in such transfers, set the variable *FTP_PASSIVE_MODE* to some value in your environment. Otherwise, the more standard ACTIVE mode may be used. If **pkg_add** consistently fails to fetch a package from a site known to work, it may be because you have a firewall that demands the usage of *passive mode* ftp.

TECHNICAL DETAILS

pkg_add extracts each package's "packing list" into a special staging directory in /var/tmp (or $PKG_TMPDIR if set) and then runs through the following sequence to fully extract the contents of the package:

1. A check is made to determine if the package or another version of it is already recorded as installed. If it is, installation is terminated if the **-u** option is not given.

   If the **-u** option is given, it's assumed the package should be replaced by the new version instead. Before doing so, all packages that depend on the pkg being upgraded are checked if they also work with the new version. If that test is successful, replacing is prepared by moving an existing +REQUIRED_BY file aside (if it exists), and running pkg_delete(1) on the installed package. Installation then proceeds as if the package was not installed, and restores the +REQUIRED_BY file afterwards.

2. A check is made to determine if the package conflicts (from **@pkgcfl** directives, see pkg_create(1)) with an already recorded as installed package. If it is, installation is terminated.

3. All package dependencies (from **@pkgdep** directives, see pkg_create(1)) are read from the packing list. If any of these required packages are not currently installed, an attempt is made to find and install it; if the missing package cannot be found or installed, the installation is terminated. If the **-u** option was specified twice, any required packages that are installed, but which have a version number that is considered to be too old, are also updated. The dependant packages are found according to the normal PKG_PATH rules.

4. A search is made for any **@option** directives which control how the package is added to the system. The only currently implemented option is **@option preserve**, which tells pkg_add to move any existing files out of the way, preserving the previous contents (which are also resurrected on pkg_delete, so caveat emptor).

5. The package build information is extracted from the +BUILD_INFO file and compared against the result of uname(3). If the operating system or architecture of the package differ from that of the host, installation is aborted. This behavior is overridable with the **-f** flag.

6. The package build information from +BUILD_INFO is then checked for USE_ABI_DEPENDS=NO (or IGNORE_RECOMMENDED). If the package was built with ABI dependency recommendations ignored, a warning will be issued.

7. If the package contains an *install* script, it is executed with the following arguments:

   pkg-name The name of the package being installed.

   PRE-INSTALL Keyword denoting that the script is to perform any actions needed before the package is installed.

   If the *install* script exits with a non-zero status code, the installation is terminated.

8. It is used as a guide for moving (or copying, as necessary) files from the staging area into their final locations.

9. If an *install* script exists for the package, it is executed with the following arguments:

pkg_name The name of the package being installed.

POST-INSTALL Keyword denoting that the script is to perform any actions needed after the package has been installed.

10. After installation is complete, a copy of the packing list, *deinstall* script, description, and display files are copied into /var/db/pkg/<pkg-name> for subsequent possible use by pkg_delete(1). Any package dependencies are recorded in the other packages' /var/db/pkg/<other-pkg>/+REQUIRED_BY file (if an alternate package database directory is specified, then it overrides the /var/db/pkg path shown above).
11. If the package is a depoted package, then add it to the default view.
12. The staging area is deleted and the program terminates.
13. Finally, if we were upgrading a package, any +REQUIRED_BY file that was moved aside before upgrading was started is now moved back into place.

The *install* script is called with the environment variable PKG_PREFIX set to the installation prefix (see the **-p** option above). This allows a package author to write a script that reliably performs some action on the directory where the package is installed, even if the user might change it with the **-p** flag to **pkg_add**. The scripts are also called with the PKG_METADATA_DIR environment variable set to the location of the +* meta-data files, and with the PKG_REFCOUNT_DBDIR environment variable set to the location of the package reference counts database directory.

ENVIRONMENT

LOCALBASE This is the location of the *viewbase* directory in which all the views are managed. The default *viewbase* directory is /usr/pkg.

PKG_DBDIR If the **-K** flag isn't given, then PKG_DBDIR is the location of the package database directory. The default package database directory is /var/db/pkg.

PKG_PATH The value of the PKG_PATH is used if a given package can't be found, it's usually set to /usr/pkgsrc/packages/All. The environment variable should be a series of entries separated by semicolons. Each entry consists of a directory name or URL. The current directory may be indicated implicitly by an empty directory name, or explicitly by a single period. FTP URLs may not end with a slash.

PKG_REFCOUNT_DBDIR
Location of the package reference counts database directory. The default location is the path to the package database directory with ".refcount" appended to the path, e.g. /var/db/pkg.refcount.

PKG_TMPDIR Staging directory for installing packages, defaults to /var/tmp. Set to directory with lots of free disk if you run out of space when installing a binary package.

PKG_VIEW The default view can be specified in the PKG_VIEW environment variable.

EXAMPLES

In all cases, **pkg_add** will try to install binary packages listed in dependencies list.

You can specify a compiled binary package explicitly on the command line.

```
# pkg_add /usr/pkgsrc/packages/All/tcsh-6.14.00.tgz
```

If you omit the version number, **pkg_add** will install the latest version available. With **-v**, **pkg_add** emits more messages to terminal.

```
# pkg_add -v /usr/pkgsrc/packages/All/unzip
```

You can grab a compiled binary package from remote location by specifying a URL. The URL can be put into an environment variable, PKG_PATH.

```
# pkg_add -v ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/i386/3.1_2007Q2/All/fire

# export PKG_PATH=ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/i386/3.1_2007Q2/All
# pkg_add -v firefox
```

Over time, as problems are found in packages, they will be moved from the `All` subdirectory into the `vulnerable` subdirectory. If you want to accept vulnerable packages by default (and know what you are doing), you can add the `vulnerable` directory to your PKG_PATH like this:

```
# export PKG_PATH="ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD/i386/3.1_2007Q2/Al
```

(The quotes are needed because semicolon (';') is a shell meta-character.) If you do this, consider installing and using the `security/audit-packages` package and running it after every **pkg_add**.

SEE ALSO

pkg_admin(1), pkg_create(1), pkg_delete(1), pkg_info(1), tar(1), mktemp(3), sysconf(3), pkgsrc(7)

AUTHORS

Jordan Hubbard
: Initial work and ongoing development.

John Kohl
: NetBSD refinements.

Hubert Feyrer
: NetBSD wildcard dependency processing, pkgdb, upgrading, etc.

Thomas Klausner
: HTTP support.

BUGS

Hard links between files in a distribution are only preserved if either (1) the staging area is on the same file system as the target directory of all the links to the file, or (2) all the links to the file are bracketed by **@cwd** directives in the contents file, *and* and the link names are extracted with a single **tar** command (not split between invocations due to exec argument-space limitations--this depends on the value returned by **sysconf**(*_SC_ARG_MAX*)).

Package upgrading needs a lot more work to be really universal.

Sure to be others.

NAME

pkg_admin — perform various administrative tasks to the pkg system

SYNOPSIS

pkg_admin [**-bqSV**] [**-d** *lsdir*] [**-K** *pkg_dbdir*] [**-s** *sfx_pattern*] *command* [args ...]

DESCRIPTION

This command performs various administrative tasks around the NetBSD Packages System.

OPTIONS

The following command-line options are supported:

-b Print only the basenames when matching package names for **lsall** and **lsbest**.

-d *lsdir*
Set *lsdir* as the path to the directory in which to find matching package names for **lsall** and **lsbest**.

-K *pkg_dbdir*
Set *pkg_dbdir* as the package database directory. If this option isn't specified, then the package database directory is taken from the value of the environment variable PKG_DBDIR if it's set, otherwise it defaults to /var/db/pkg.

-q Perform checks in a quiet manner. In normal operation, **pkg_admin** prints a '.' to standard output to indicate progress. This option suppresses this progress indicator.

-S Set the shell glob pattern for package suffixes when matching package names for **lsall** and **lsbest** to be the null suffix.

-s *sfx_pattern*
Set the shell glob pattern for package suffixes when matching package names for **lsall** and **lsbest**. The default pattern is ".t[bg]z".

-V Print version number and exit.

The following commands are supported:

add *pkg ...*
For each listed package, write the absolute pathnames of the files listed in its +CONTENTS file together with the package they belong to into the package database. This should be used only by pkg_view(1).

check [*pkg ...*]
Use this command to check the files belonging to some or all of the packages installed on the local machine against the checksum which was recorded in the +CONTENTS files at package installation time. Symbolic links also have their integrity checked against the recorded value at package installation time. If no additional argument is given, the files of all installed packages are checked, else only the named packages will be checked (wildcards can be used here, see pkg_info(1)).

The packages' +CONTENTS files will be parsed and the checksum will be checked for every file found. A warning message is printed if the expected checksum differs from the checksum of the file on disk. Symbolic links are also checked, ensuring that the targets on disk are the same as the contents recorded at package installation time.

delete *pkg ...*
For each listed package, remove all file entries in the package database that belong to the package. This should be used only by pkg_view(1).

dump Dump the contents of the package database, similar to **pkg_info -F**. Columns are printed for the keyfield used in the pkgdb - the filename -, and the data field - the package the file belongs to.

lsall */dir/pkgpattern*

lsbest */dir/pkgpattern*

List all/best package matching pattern in the given directory `/dir`. If the **-d** flag is given, then that directory path overrides `/dir`. Can be used to work around limitations of /bin/sh and other filename globbing mechanisms. This option implements matching of pkg-wildcards against arbitrary files and directories, useful mainly in the build system itself. See `pkg_info`(1) for a description of the pattern.

Example:

```
yui# cd /usr/pkgsrc/packages/i386ELF/All/
yui# ls unzip*
unzip-5.40.tgz  unzip-5.41.tgz
yui# pkg_admin lsall 'unzip*'
/usr/pkgsrc/packages/i386ELF/All/unzip-5.40.tgz
/usr/pkgsrc/packages/i386ELF/All/unzip-5.41.tgz
yui# pkg_admin lsall 'unzip≥5.40'
/usr/pkgsrc/packages/i386ELF/All/unzip-5.40.tgz
/usr/pkgsrc/packages/i386ELF/All/unzip-5.41.tgz
yui# pkg_admin lsall 'unzip≥5.41'
/usr/pkgsrc/packages/i386ELF/All/unzip-5.41.tgz
yui# pkg_admin lsbest 'unzip≥5.40'
/usr/pkgsrc/packages/i386ELF/All/unzip-5.41.tgz
yui# pkg_admin lsall /usr/pkgsrc/packages/i386ELF/All/'{mit,unproven}-pthread*
/usr/pkgsrc/packages/i386ELF/All/mit-pthreads-1.60b6.tgz
/usr/pkgsrc/packages/i386ELF/All/unproven-pthreads-0.15.tgz
```

pmatch *pattern pkg*

Returns true if *pkg* matches *pattern*, otherwise returns false.

rebuild

Rebuild the package database mapping from scratch, scanning subdirectories in `/var/db/pkg` for `+CONTENTS` files, parsing them and writing the resulting absolute pathnames together with the package they belong to into the package database.

This option is intended to be used for upgrading from non-pkgdb-pkg_* tools to pkgdb-pkg_* tools, further manipulation of the pkgdb will be done by `pkg_add`(1), `pkg_delete`(1), and `pkg_create`(1).

Needs to be run as root.

rebuild-tree

Rebuild the +REQUIRED_BY files from scratch by reresolving all dependencies.

This option is intended to be used for fixing inconsistencies between the records of depending and depended-on packages, such as can arise by the use of **pkg_delete -f**.

set *variable=value pkg ...*

Set variable with information about the installed package. Use **unset** to remove a variable.

Packages that are not installed directly by the user but pulled in as dependencies are marked by setting "automatic=YES".

unset *variable pkg ...*
Remove an installation variable.

ENVIRONMENT

PKG_DBDIR
If the **-K** flag isn't given, then PKG_DBDIR is the location of the package database directory. The default package database directory is /var/db/pkg.

FILES

/var/db/pkg/pkgdb.byfile.db
/var/db/pkg/<pkg>/+CONTENTS

SEE ALSO

pkg_add(1), pkg_create(1), pkg_delete(1), pkg_info(1), pkg_view(1), pkgsrc(7)

HISTORY

The **pkg_admin** command first appeared in NetBSD 1.4.

AUTHORS

The **pkg_admin** command was written by Hubert Feyrer.

NAME

pkg_create — a utility for creating software package distributions

SYNOPSIS

pkg_create [**-ElORUVv**] [**-B** *build-info-file*] [**-b** *build-version-file*]
[**-C** *cpkgs*] [**-D** *displayfile*] [**-g** *group*] [**-I** *realprefix*]
[**-i** *iscript*] [**-K** *pkg_dbdir*] [**-k** *dscript*] [**-L** *SrcDir*]
[**-n** *preserve-file*] [**-P** *dpkgs*] [**-T** *buildpkgs*] [**-p** *prefix*]
[**-S** *size-all-file*] [**-s** *size-pkg-file*] [**-t** *template*] [**-u** *owner*]
-c *comment* **-d** *description* **-f** *packlist* *pkg-name*

DESCRIPTION

The **pkg_create** command is used to create packages that will subsequently be fed to one of the package extraction/info utilities. The input description and command line arguments for the creation of a package are not really meant to be human-generated, though it is easy enough to do so. It is more expected that you will use a front-end tool for the job rather than muddling through it yourself. Nonetheless, a short description of the input syntax is included in this document.

OPTIONS

The following command line options are supported:

-B *build-info-file*
: Install the file *build-info-file* so that users of binary packages can see what make(1) definitions were used to control the build when creating the binary package. This allows various build definitions to be retained in a binary package and viewed wherever it is installed, using pkg_info(1).

-b *build-version-file*
: Install the file *build-version-file* so that users of binary packages can see what versions of the files used to control the build were used when creating the binary package. This allows some fine-grained version control information to be retained in a binary package and viewed wherever it is installed, using pkg_info(1).

-C *cpkgs*
: Set the initial package conflict list to *cpkgs*. This is assumed to be a whitespace separated list of package names and is meant as a convenient shorthand for specifying multiple **@pkgcfl** directives in the packing list (see PACKING LIST DETAILS section below).

-c *[-]desc*
: Fetch package (one line description) from file *desc* or, if preceded by **-**, the argument itself. This string should also give some idea of which version of the product (if any) the package represents.

-D *displayfile*
: Display the file after installing the package. Useful for things like legal notices on almost-free software, etc.

-d *[-]desc*
: Fetch long description for package from file *desc* or, if preceded by **-**, the argument itself.

-E
: Add an empty views file to the package.

-f *packlist*
: Fetch (packing list) for package from the file *packlist* or **stdin** if *packlist* is a **-** (dash).

-g *group*
Make *group* the default group ownership instead of extracting it from the file system.

-I *realprefix*
Provide the real prefix, as opposed to the staging prefix, for use in staged installations of packages.

-i *iscript*
Set *iscript* to be the install procedure for the package. This can be any executable program (or shell script). It will be invoked automatically when the package is later installed.

-K *pkg_dbdir*
Set *pkg_dbdir* as the package database directory. If this option isn't specified, then the package database directory is taken from the value of the environment variable PKG_DBDIR if it's set, otherwise it defaults to /var/db/pkg.

-k *dscript*
Set *dscript* to be the de-install procedure for the package. This can be any executable program (or shell script). It will be invoked automatically when the package is later (if ever) de-installed.

-L *SrcDir*
This sets the package's @src directive; see below for a description of what this does.

-l Check that any symbolic links which are to be placed in the package are relative to the current prefix. This means using unlink(2) and symlink(2) to remove and re-link any symbolic links which are targeted at full path names.

-n *preserve-file*
The file is used to denote that the package should not be deleted. This is intended for use where the deletion of packages may present a bootstrap problem.

-O Go into a (packing list only) mode. This is used to do (fake pkg_add) operations when a package is installed. In such cases, it is necessary to know what the final, adjusted packing list will look like.

-P *dpkgs*
Set the initial package dependency list to *dpkgs*. This is assumed to be a whitespace separated list of package names and is meant as a convenient shorthand for specifying multiple **@pkgdep** directives in the packing list (see PACKING LIST DETAILS section below). In addition, the exact versions of the packages referred to in the *dpkgs* list will be added to the packing list in the form of **@blddep** directives.

-T *buildpkgs*
The exact versions of the packages referred to in the *buildpkgs* list will be added to the packing list in the form of **@blddep** directives. This directives are stored after those created by the **-P** option. *buildpkgs* is assumed to be a whitespace separated list of package names.

-p *prefix*
Set *prefix* as the initial directory (base) to start from in selecting files for the package.

-R Re-order any directories in the pkg/PLIST file into reverse alphabetic order, so that child directories will automatically be removed before parent directories.

-S *size-all-file*
Store the given file for later querying with the pkg_info(1) **-S** flag. The file is expected to contain the size (in bytes) of all files of this package plus any required packages added up and stored as a ASCII string, terminated by a newline.

-s *size-pkg-file*
Store the given file for later querying with the pkg_info(1) **-s** flag. The file is expected to contain the size (in bytes) of all files of this package added up and stored as a ASCII string, terminated by a newline.

-t *template*
Use *template* as the input to mktemp(3). By default, this is the string /tmp/instmp.XXXXXX, but it may be necessary to override it in the situation where space in your /tmp directory is limited. Be sure to leave some number of 'X' characters for mktemp(3) to fill in with a unique ID.

-U Do not update the package file database with any file information.

-u *owner*
Make *owner* the default owner instead of extracting it from the file system.

-V Print version number and exit.

-v Turn on verbose output.

PACKING LIST DETAILS

The (packing list) format (see **-f**) is fairly simple, being nothing more than a single column of filenames to include in the package. However, since absolute pathnames are generally a bad idea for a package that could be installed potentially anywhere, there is another method of specifying where things are supposed to go and, optionally, what ownership and mode information they should be installed with. This is done by embedding specialized command sequences in the packing list. Briefly described, these sequences are:

@cwd *directory*
Set the internal directory pointer to point to *directory*. All subsequent filenames will be assumed relative to this directory. Note: **@cd** is also an alias for this command.

@src *directory*
Set the internal directory pointer for _creation only_ to *directory*. That is to say that it overrides **@cwd** for package creation but not extraction.

@exec *command*
Execute *command* as part of the unpacking process. If *command* contains any of the following sequences somewhere in it, they will be expanded inline. For the following examples, assume that **@cwd** is set to /usr/local and the last extracted file was bin/emacs.

%F Expands to the last filename extracted (as specified), in the example case bin/emacs

%D Expand to the current directory prefix, as set with **@cwd**, in the example case /usr/local.

%B Expand to the (basename) of the fully qualified filename, that is the current directory prefix, plus the last filespec, minus the trailing filename. In the example case, that would be /usr/local/bin.

%f Expand to the (filename) part of the fully qualified name, or the converse of **%B**, being in the example case, emacs.

@unexec *command*
Execute *command* as part of the deinstallation process. Expansion of special **%** sequences is the same as for **@exec**. This command is not executed during the package add, as **@exec** is, but rather when the package is deleted. This is useful for deleting links and other ancillary files that were created as a result of adding the package, but not directly known to the package's table of contents (and hence not automatically removable). The advantage of using **@unexec** over a deinstallation script is that you can use the (special sequence expansion) to get at files regardless of where they've been potentially redirected (see **-p**).

@mode *mode*
: Set default permission for all subsequently extracted files to *mode*. Format is the same as that used by the **chmod** command (well, considering that it's later handed off to it, that's no surprise). Use without an arg to set back to default (extraction) permissions.

@option *option*
: Set internal package options, the only currently supported one being *preserve*, which tells pkg_add to move any existing files out of the way, preserving the previous contents (which are also resurrected on pkg_delete, so caveat emptor).

@owner *user*
: Set default ownership for all subsequently extracted files to *user*. Use without an arg to set back to default (extraction) ownership.

@group *group*
: Set default group ownership for all subsequently extracted files to *group*. Use without an arg to set back to default (extraction) group ownership.

@comment *string*
: Imbed a comment in the packing list. Useful in trying to document some particularly hairy sequence that may trip someone up later.

@ignore
: Used internally to tell extraction to ignore the next file (don't copy it anywhere), as it's used for some special purpose.

@ignore_inst
: Similar to **@ignore**, but the ignoring of the next file is delayed one evaluation cycle. This makes it possible to use this directive in the *packinglist* file, so you can pack a specialized datafile in with a distribution for your install script (or something) yet have the installer ignore it.

@name *name*
: Set the name of the package. This is mandatory and is usually put at the top. This name is potentially different than the name of the file it came in, and is used when keeping track of the package for later deinstallation. Note that **pkg_create** will derive this field from the *pkg-name* and add it automatically if none is given.

@dirrm *name*
: Declare directory name to be deleted at deinstall time. By default, directories created by a package installation are not deleted when the package is deinstalled; this provides an explicit directory cleanup method. This directive should appear at the end of the package list. If more than one **@dirrm** directives are used, the directories are removed in the order specified. The name directory will not be removed unless it is empty.

@display *name*
: Declare name as the file to be displayed at install time (see **-D** above).

@pkgdep *pkgname*
: Declare a dependency on the *pkgname* package. The *pkgname* package must be installed before this package may be installed, and this package must be deinstalled before the *pkgname* package is deinstalled. Multiple **@pkgdep** directives may be used if the package depends on multiple other packages.

@blddep *pkgname*
: Declare that this package was built with the exact version of *pkgname* (since the **@pkgdep** directive may contain wildcards or relational package version information).

@pkgcfl *pkgcflname*
: Declare a conflict with the *pkgcflname* package, as the two packages contain references to the same files, and so cannot co-exist on the same system.

SEE ALSO

`pkg_add`(1), `pkg_admin`(1), `pkg_delete`(1), `pkg_info`(1), `sysconf`(3), `pkgsrc`(7)

HISTORY

The **`pkg_create`** command first appeared in FreeBSD.

AUTHORS

Jordan Hubbard
: most of the work

John Kohl
: refined it for NetBSD

Hubert Feyrer
: NetBSD wildcard dependency processing, pkgdb, pkg size recording etc.

BUGS

Hard links between files in a distribution must be bracketed by **`@cwd`** directives in order to be preserved as hard links when the package is extracted. They additionally must not end up being split between **`tar`** invocations due to exec argument-space limitations (this depends on the value returned by **`sysconf`**(*_SC_ARG_MAX*)).

NAME

pkg_delete — a utility for deleting previously installed software package distributions

SYNOPSIS

pkg_delete [**-DdFfNnORrVv**] [**-K** *pkg_dbdir*] [**-p** *prefix*] *pkg-name ...*

DESCRIPTION

The **pkg_delete** command is used to delete packages that have been previously installed with the pkg_add(1) command.

WARNING

Since the **pkg_delete** *command may execute scripts or programs provided by a package file, your system may be susceptible to "Trojan horses" or other subtle attacks from miscreants who create dangerous package files.*

You are advised to verify the competence and identity of those who provide installable package files. For extra protection, examine all the package control files in the package record directory (/var/db/pkg/<pkg-name>/*). Pay particular attention to any* +INSTALL *or* +DEINSTALL *files, and inspect the* +CONTENTS *file for* **@cwd**, **@mode** *(check for setuid),* **@dirrm**, **@exec**, *and* **@unexec** *directives, and/or use the* pkg_info(*1*) *command to examine the installed package control files.*

OPTIONS

The following command line options are supported:

pkg-name ...

The named packages are deinstalled, wildcards can be used, see pkg_info(1). If no version is given, the one currently installed will be removed. If the **-F** flag is given, one or more (absolute) filenames may be specified and the Package Database will be consulted for the package to which the given file belongs. These packages are then deinstalled.

-D If a deinstallation script exists for a given package, do not execute it.

-d Remove empty directories created by file cleanup. By default, only files/directories explicitly listed in a package's contents (either as normal files/directories or with the **@dirrm** directive) will be removed at deinstallation time. This option tells **pkg_delete** to also remove any directories that were emptied as a result of removing the package.

-F Any pkg-name given will be interpreted as pathname which is subsequently transformed in a (real) package name via the Package Database. That way, packages can be deleted by giving a filename instead of the package-name.

-f Force removal of the package, even if a dependency is recorded or the deinstall script fails.

-ff Force removal of the package, even if the package is marked as a **preserved** package. Note that this is a dangerous operation.

-K *pkg_dbdir*

Set *pkg_dbdir* as the package database directory. If this option isn't specified, then the package database directory is taken from the value of the environment variable PKG_DBDIR if it's set, otherwise it defaults to /var/db/pkg.

-N Remove the package's registration and its entries from the package database, but leave the files installed. Don't run any deinstall scripts or @unexec lines either.

-n Don't actually deinstall a package, just report the steps that would be taken if it were.

-O Only delete the package's entries from the package database, do not touch the package or its files itself.

-p *prefix*
Set *prefix* as the directory in which to delete files from any installed packages which do not explicitly set theirs. For most packages, the prefix will be set automatically to the installed location by pkg_add(1).

-R This option triggers a recursive delete of the given package and any packages it depends on, unless some other package still needs a dependent package. This **-R** option can be used to clean up by deleting a package and all its then-unneeded dependent packages.

-r **pkg_delete** first builds a list of all packages that require (directly and indirectly) the one being deleted. It then deletes these packages using **pkg_delete** with the given options before deleting the user specified package. This **-r** option can be used to recursively delete a package and all of the packages which depend on that package.

-V Print version number and exit.

-v Turn on verbose output.

TECHNICAL DETAILS

pkg_delete does pretty much what it says. It examines installed package records in /var/db/pkg/<pkg-name>, deletes the package contents, and finally removes the package records (if an alternate package database directory is specified, then it overrides the /var/db/pkg path shown above).

If a package is required by other installed packages, **pkg_delete** will list those dependent packages and refuse to delete the package (unless the **-f** option is given).

If a package has been marked as a **preserved** package, it will not be able to be deleted (unless more than one occurrence of the **-f** option is given).

If a filename is given instead of a package name, the package of which the given file belongs to can be deleted if the **-F** Flag is given. The filename needs to be absolute, see the output produced by the pkg_info **-aF** command.

If a **deinstall** script exists for the package, it is executed before and after any files are removed. It is this script's responsibility to clean up any additional messy details around the package's installation, since all **pkg_delete** knows how to do is delete the files created in the original distribution. The **deinstall** script is called as:

deinstall ⟨*pkg-name*⟩ *VIEW-DEINSTALL*

before removing the package from a view, and as:

deinstall ⟨*pkg-name*⟩ *DEINSTALL*

before deleting all files and as:

deinstall ⟨*pkg-name*⟩ *POST-DEINSTALL*

after deleting them. Passing the keywords *VIEW-DEINSTALL*, *DEINSTALL* and *POST-DEINSTALL* lets you potentially write only one program/script that handles all aspects of installation and deletion.

All scripts are called with the environment variable PKG_PREFIX set to the installation prefix (see the **-p** option above). This allows a package author to write a script that reliably performs some action on the directory where the package is installed, even if the user might have changed it by specifying the **-p** option when running **pkg_delete** or pkg_add(1). The scripts are also called with the PKG_METADATA_DIR environment variable set to the location of the +* meta-data files, and with the PKG_REFCOUNT_DBDIR environment variable set to the location of the package reference counts database directory.

ENVIRONMENT

`PKG_DBDIR` If the **`-K`** flag isn't given, then `PKG_DBDIR` is the location of the package database directory. The default package database directory is `/var/db/pkg`.

`PKG_REFCOUNT_DBDIR`
Location of the package reference counts database directory. The default location is the path to the package database directory with ".refcount" appended to the path, e.g. `/var/db/pkg.refcount`.

SEE ALSO

`pkg_add`(1), `pkg_admin`(1), `pkg_create`(1), `pkg_info`(1), `mktemp`(3), `pkgsrc`(7)

AUTHORS

Jordan Hubbard
most of the work
John Kohl
refined it for NetBSD
Hubert Feyrer
NetBSD wildcard dependency processing, pkgdb, recursive "down" delete, etc.

NAME

pkg_info — a utility for displaying information on software packages

SYNOPSIS

pkg_info [**-BbcDdFfhIikLmNnpqRSsVvX**] [**-e** *package*] [**-E** *package*] [**-K** *pkg_dbdir*] [**-l** *prefix*] *pkg-name ...*
pkg_info [**-a** | **-u**] [flags]
pkg_info [**-Q** *variable*] *pkg-name ...*

DESCRIPTION

The **pkg_info** command is used to dump out information for packages, which may be either packed up in files or already installed on the system with the pkg_create(1) command.

The *pkg-name* may be the name of an installed package (with our without version), a pattern matching several installed packages (see the **PACKAGE WILDCARDS** section for a description of possible patterns), the pathname to a binary package, a filename belonging to an installed package (if **-F** is also given), or a URL to an ftp-available package.

The following command-line options are supported:

-a Show information for all currently installed packages. See also **-u**.

-B Show some of the important definitions used when building the binary package (the “Build information”) for each package. Additionally, any installation information variables (lowercase) can be queried, too. In particular, *automatic* tells if a package was installed automatically as a dependency of another package.

-b Show the NetBSD RCS Id strings from the files used in the construction of the binary package (the "Build version") for each package. These files are the package Makefile, any patch files, any checksum files, and the packing list file.

-c Show the one-line comment field for each package.

-D Show the install-message file (if any) for each package.

-d Show the long-description field for each package.

-E *pkg-name*
This option allows you to test for the existence of a given package. If a package identified by *pkg-name* is currently installed, return code is 0, otherwise 1. The name of the best matching package found installed is printed to stdout unless turned off using the **-q** option. *pkg-name* can contain wildcards (see the **PACKAGE WILDCARDS** section below).

-e *pkg-name*
This option allows you to test for the existence of a given package. If a package identified by *pkg-name* is currently installed, return code is 0, otherwise 1. The names of any package(s) found installed are printed to stdout unless turned off using the **-q** option. *pkg-name* can contain wildcards (see the **PACKAGE WILDCARDS** section below).

-F Interpret any pkg-name given as filename, and translate it to a package name using the Package Database. This can be used to query information on a per-file basis, e.g. in conjunction with the **-e** flag to find out which package a file belongs to.

-f Show the packing list instructions for each package.

-I Show the index entry for each package.

-i Show the install script (if any) for each package.

-K *pkg_dbdir*
Set *pkg_dbdir* as the package database directory. If this option isn't specified, then the package database directory is taken from the value of the environment variable PKG_DBDIR if it's set, otherwise it defaults to /var/db/pkg.

-k Show the de-install script (if any) for each package.

-L Show the files within each package. This is different from just viewing the packing list, since full pathnames for everything are generated. Files that were created dynamically during installation of the package are not listed.

-l *str*
Prefix each information category header (see **-q**) shown with *str*. This is primarily of use to front-end programs that want to request a lot of different information fields at once for a package, but don't necessary want the output intermingled in such a way that they can't organize it. This lets you add a special token to the start of each field.

-m Show the mtree file (if any) for each package.

-N Show which packages each package was built with (exact dependencies), if any.

-n Show which packages each package needs (depends upon), if any.

-p Show the installation prefix for each package.

-Q Show the definition of *variable* from the build information for each package. An empty string is returned if no such variable definition is found for the package(s).

-q Be "quiet" in emitting report headers and such, just dump the raw info (basically, assume a non-human reading).

-R For each package, show the packages that require it.

-S Show the size of this package and all the packages it requires, in bytes.

-s Show the size of this package in bytes. The size is calculated by adding up the size of each file of the package.

-u Show information for all user-installed packages. Automatically installed packages (as dependencies of other packages) are not displayed. See also **-a**.

-V Print version number and exit.

-v Turn on verbose output.

-X Print summary information for each package. The summary format is described in pkg_summary(5). Its primary use is to contain all information about the contents of a (remote) binary package repository needed by package managing software.

TECHNICAL DETAILS

Package info is either extracted from package files named on the command line, or from already installed package information in /var/db/pkg/<pkg-name>.

A filename can be given instead of a (installed) package name to query information on the package this file belongs to. This filename is then resolved to a package name using the Package Database. For this translation to take place, the **-F** flag must be given. The filename must be absolute, compare the output of pkg_info **-aF**.

PACKAGE WILDCARDS

In the places where a package name/version is expected, e.g. for the **-e** switch, several forms can be used. Either use a package name with or without version, or specify a package wildcard that gets matched against all installed packages.

Package wildcards use `fnmatch`(3). In addition, `csh`(1) style {,} alternates have been implemented. Package version numbers can also be matched in a relational manner using the ≥, ≤, >, and < operators. For example, `pkg_info -e 'name≥1.3'` will match versions 1.3 and later of the `name` package. Additionally, ranges can be defined by giving a lower bound with > or ≥ and an upper bound with < or ≤. The lower bound has to come first. For example, `pkg_info -e 'name≥1.3<2.0'` will match versions 1.3 (inclusive) to 2.0 (exclusive) of package `name`.

The collating sequence of the various package version numbers is unusual, but strives to be consistent. The magic string "alpha" equates to `alpha version` and sorts before a beta version. The magic string "beta" equates to `beta version` and sorts before a release candidate. The magic string "rc" equates to `release candidate` and sorts before a release. The magic string "pre", short for "pre-release", is a synonym for "rc". For example, `name-1.3rc3` will sort before `name-1.3` and after `name-1.2.9`. Similarly `name-1.3alpha2` will sort before `name-1.3beta1` and they both sort before `name-1.3rc1`. In addition, alphabetic characters sort in the same place as their numeric counterparts, so that `name-1.2e` has the same sorting value as `name-1.2.5` The magic string "pl" equates to a `patch level` and has the same value as a dot in the dewey-decimal ordering schemes.

ENVIRONMENT

`PKG_DBDIR` If the **-K** flag isn't given, then `PKG_DBDIR` is the location of the package database directory. The default package database directory is `/var/db/pkg`.

`PKG_PATH` This can be used to specify a semicolon-separated list of paths and URLs to search for package files. If `PKG_PATH` is used, the suffix `.tgz` is automatically appended to the *pkg-name*, whereas searching in the current directory uses *pkg-name* literally.

`PKG_TMPDIR`, `TMPDIR`

These are tried in turn (if set) as candidate directories in which to create a "staging area" for any files extracted by **pkg_info** from package files. If neither `PKG_TMPDIR` nor `TMPDIR` yields a suitable scratch directory, `/var/tmp`, `/tmp`, and `/usr/tmp` are tried in turn. Note that `/usr/tmp` may be created, if it doesn't already exist.

Since **pkg_info** requires very little information to be extracted from any package files examined, it is unlikely that these environment variables would ever need to be used to work around limited available space in the default locations.

SEE ALSO

`pkg_add`(1), `pkg_admin`(1), `pkg_create`(1), `pkg_delete`(1), `mktemp`(3), `pkgsrc`(7), `mtree`(8)

AUTHORS

Jordan Hubbard
: most of the work

John Kohl
: refined it for NetBSD

Hubert Feyrer
: NetBSD wildcard dependency processing, pkgdb, depends displaying, pkg size display etc.

NAME

pkg_view — add and delete instances of depoted packages in views

SYNOPSIS

pkg_view [**-nVv**] [**-d** *stowdir*] [**-i** *ignore*] [**-k** *pkg_dbdir*] [**-W** *viewbase*] [**-w** *view*] *command package ...*

DESCRIPTION

The **pkg_view** command is used to add and delete instances of depoted packages in *stowdir* in a *view* in the *viewbase* directory.

WARNING

Since the **pkg_view** *command may execute scripts or programs provided by a package file, your system may be susceptible to "Trojan horses" or other subtle attacks from miscreants who create dangerous package files.*

You are advised to verify the competence and identity of those who provide installable package files. For extra protection, examine all the package control files in the package database directory (/usr/pkg/packages/<pkg-name>/*). Pay particular attention to any* +INSTALL *or* +DEINSTALL *files, and inspect the* +CONTENTS *file for* **@cwd**, **@mode** *(check for setuid),* **@dirrm**, **@exec**, *and* **@unexec** *directives, and/or use the* pkg_info(*1*) *command to examine the installed package control files.*

OPTIONS

The following command-line options are supported:

-d *stowdir*
: Set *stowdir* as the directory in which the depoted packages can be found. If this option isn't specified, then the *stowdir* is taken from the value of the environment variable DEPOTBASE if it's set, otherwise the default *stowdir* is the path to the packages directory under *viewbase*.

-i *ignore*
: Add *ignore* to the list of files in *package* that should ignored when adding or removing the package instance from *view*.

-k *pkg_dbdir*
: Set *pkg_dbdir* as the package database directory for the default (empty) view. If this option isn't specified, then the package database directory is taken from the value of the environment variable PKG_DBDIR, otherwise it defaults to /var/db/pkg.

-n
: Don't actually execute the commands for manipulating the package instances.

-V
: Print the version number and exit.

-v
: Turn on verbose output. Specifying **-v** multiple times increases the level of verbosity.

-W *viewbase*
: Set *viewbase* as the directory in which all the views are managed. The default *viewbase* directory is /usr/pkg but may be overridden by the LOCALBASE environment variable.

-w *view*
: Set *view* as the directory in *viewbase* in which the package instances should be added or deleted. The default *view* is the empty view but may be overridden by the PKG_VIEW environment variable.

The following commands are supported:

add Add the listed package instances into *view*.

check Check whether the listed package instances are present in *view*. If they are not present, then return 0, otherwise return 1.

delete Delete the listed package instances from *view*.

ENVIRONMENT

DEPOTBASE

This is the location of the *stowdir* directory inside which all depoted packages are kept. The default *stowdir* is the packages directory under *viewbase*.

LOCALBASE

This is the location of the *viewbase* directory in which all the views are managed. The default *viewbase* directory is /usr/pkg.

PKG_DBDIR

If the **-k** flag isn't given, then the value of the environment variable PKG_DBDIR is the package database directory for the default view, otherwise it defaults to /var/db/pkg.

PLIST_IGNORE_FILES

This can be used to specify files in *package* that should ignored when adding or removing the package instance from *view*. PLIST_IGNORE_FILES is a space-separated list of shell glob patterns that match files relative to the *package* depot directory, and it defaults to "info/dir *[~#] *.OLD *.orig *,v". This is overridden by any _PLIST_IGNORE_FILES setting in a package's *build-info-file* (see pkg_create(1)) if it exists.

PKG_VIEW

The default view can be specified in the PKG_VIEW environment variable.

FILES

<pkg-dbdir>/<package>/+INSTALL

If the package contains an *install* script (see pkg_create(1)), then after the package instance is added into a view, the script is executed with the following arguments:

package The name of the package instance being added.

VIEW-INSTALL

Keyword denoting that the script is to perform any actions needed after the package instance is added to a view.

If the *install* script exits with a non-zero status code, the installation is terminated.

<pkg-dbdir>/<package>/+DEINSTALL

If the package contains an *deinstall* script (see pkg_create(1)), then before the package instance is deleted from a view, the script is executed with the following arguments:

package The name of the package instance being deleted.

VIEW-DEINSTALL

Keyword denoting that the script is to perform any actions needed before the package instance is deleted from a view.

If the *deinstall* script exits with a non-zero status code, the de-installation is terminated.

The *install* and *deinstall* scripts are called with the environment variable PKG_PREFIX set to the path to the *view* directory.

SEE ALSO

`linkfarm`(1), `pkg_delete`(1)

AUTHORS

The **`pkg_view`** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

pkill — find or signal processes by name

SYNOPSIS

pgrep [**-filnvx**] [**-d** *delim*] [**-G** *gid*] [**-g** *pgrp*] [**-P** *ppid*] [**-s** *sid*] [**-t** *tty*]
[**-U** *uid*] [**-u** *euid*] [*pattern ...*]
pkill [**-signal**] [**-finvx**] [**-G** *gid*] [**-g** *pgrp*] [**-P** *ppid*] [**-s** *sid*] [**-t** *tty*]
[**-U** *uid*] [**-u** *euid*] [*pattern ...*]

DESCRIPTION

The **pgrep** command searches the process table on the running system and prints the process IDs of all processes that match the criteria given on the command line.

The **pkill** command searches the process table on the running system and signals all processes that match the criteria given on the command line.

The following options are available:

-d *delim*
: Specify a delimiter to be printed between each process ID. The default is a newline. This option can only be used with the **pgrep** command.

-f
: Match against full argument lists. The default is to match against process names.

-G *gid*
: Restrict matches to processes with a real group ID in the comma-separated list *gid*.

-g *pgrp*
: Restrict matches to processes with a process group ID in the comma-separated list *pgrp*. The value zero is taken to mean the process group ID of the running **pgrep** or **pkill** command.

-i
: Ignore case distinctions in both the process table and the supplied pattern.

-l
: Long output. Print the process name in addition to the process ID for each matching process. If used in conjunction with **-f**, print the process ID and the full argument list for each matching process.

-n
: Match only the most recently created process, if any.

-P *ppid*
: Restrict matches to processes with a parent process ID in the comma-separated list *ppid*.

-s *sid*
: Restrict matches to processes with a session ID in the comma-separated list *sid*. The value zero is taken to mean the session ID of the running **pgrep** or **pkill** command.

-t *tty*
: Restrict matches to processes associated with a terminal in the comma-separated list *tty*. Terminal names may be of the form ‘ttyxx’ or the shortened form ‘xx’. A single dash (‘-’) matches processes not associated with a terminal.

-U *uid*
: Restrict matches to processes with a real user ID in the comma-separated list *uid*.

-u *euid*
: Restrict matches to processes with an effective user ID in the comma-separated list *euid*.

-v
: Reverse the sense of the matching; display processes that do not match the given criteria.

-x
: Require an exact match of the process name, or argument list if **-f** is given. The default is to match any substring.

-signal
: A non-negative decimal number or symbolic signal name specifying the signal to be sent instead of the default TERM. This option is valid only when given as the first argument to **pkill**.

Note that a running **pgrep** or **pkill** process will never consider itself or system processes (kernel threads) as a potential match.

EXIT STATUS

pgrep and **pkill** return one of the following values upon exit:

0 One or more processes were matched.

1 No processes were matched.

2 Invalid options were specified on the command line.

3 An internal error occurred.

SEE ALSO

grep(1), kill(1), ps(1), kill(2), sigaction(2), re_format(7), signal(7)

HISTORY

pkill and **pgrep** first appeared in NetBSD 1.6. They are modelled after utilities of the same name that appeared in Sun Solaris 7.

NAME

pmap — display process memory map

SYNOPSIS

pmap [**-adlmPRsv**] [**-A** *address*] [**-D** *number*] [**-E** *address*] [**-M** *core*] [**-N** *system*] [**-p** *pid*] [**-S** *address*] [**-V** *address*] [*pid ...*]

DESCRIPTION

The **pmap** utility lists the virtual memory mappings underlying the given process. The start address of each entry is always given, and, depending on the options given, other information such as the end address, the underlying file's device and inode numbers, and various protection information will be displayed, along with the path to the file, if such data is available.

By default, **pmap** displays information for its parent process, so that when run from a shell prompt, the shell's memory information is displayed. If other PIDs are given as arguments on the command line, information for those processes will be printed also. If the special PID of 0 is given, then information for the kernel's memory map is printed.

The options are as follows:

-A *address* Dumps the vm_amap structure found at *address*.

-a Display "all" information from the process's memory map. This output mode is an amalgam of the contents of the Solaris, Linux, and NetBSD style output modes.

-D *number* Enable various debug facilities. The *number* is a bit mask of the values:

0x01	dump the process's vmspace structure
0x02	dump the process's vm_map structure
0x04	dump the vm_map.header structure
0x08	dump each vm_map_entry in its entirety
0x10	dump the vm_amap structure attached to the vm_map_entry, if applicable
0x20	dump the vm_amap slot data, if present (requires 0x10)
0x40	dump the vm_anon data from the am_anon array, if present (requires 0x20)
0x1000	dump the namei cache as it is traversed

-d Dumps the vm_map and vm_map_entry structures in a style similar to that of ddb(4). When combined with the **-v** option, the device number, inode number, name, vnode addresses, or other identifying information from the vm_map_entries will be printed.

-E *address* Dumps the vm_map_entry structure found at *address*.

-l Dumps information in a format like the contents of the maps pseudo-file under the /proc file system which was, in turn, modeled after the similarly named entry in the Linux /proc file system. When combined with the **-v** option, identifiers for all entries are printed.

-M *core* Extract values associated with the name list from the specified core instead of the default /dev/kmem.

-m Dumps information in the same format as the map pseudo-file of the /proc file system. When the **-v** option is also given, device number, inode number, and filename or other identifying information is printed.

-N *system* Extract the name list from the specified system instead of the default /netbsd.

-P Causes **pmap** to print information about itself.

-p *pid* Tells **pmap** to print information about the given process. If **-p** *pid* occurs last on the command line, the **-p** is optional.

-R Recurse into submaps. In some cases, a vm_map_entry in the kernel will point to a submap. Using this flag tells **pmap** to print the entries of the submap as well. The submap output is indented, and does not affect any total printed at the bottom of the output.

-S *address* Dumps the vmspace structure found at *address*.

-s The Solaris style output format, modeled after the Solaris command of the same name. This is the default output style.

-V *address* Dumps the vm_map structure found at *address*. Note that if you print the vm_map of a process, there may not be a way to properly determine which map entries are related to the stack.

-v Verbose output. When used with **-d**, **-l**, or **-m**, more information is printed, possibly including device and inode numbers, file path names, or other identifying information. If specified more than once, a small note will be printed in between two entries that are not adjacent, making the visual identification of spaces in the process's map easier to see, that indicates the number of pages and the amount of memory space that is skipped.

The **-P** and **-p** options override each other, so the last one to appear on the command line takes effect. If you do wish to see information about **pmap** and another process as the same time, simply omit the **-p** and place the extra PID at the end of the command line.

EXIT STATUS

pmap exits 0 on success, and >0 if an error occurred.

EXAMPLES

While the meaning of most of the output is self-evident, some pieces of it may appear to be a little inscrutable.

Here is a portion of the default output from **pmap** being run at an sh(1) prompt showing the starting address of the map entry, the size of the map entry, the current protection level of the map entry, and either the name of the file backing the entry or some other descriptive text.

```
$ pmap
08048000    420K read/exec         /bin/sh
080B1000      8K read/write        /bin/sh
080B3000     28K read/write          [ anon ]
080BA000     16K read/write/exec     [ heap ]
...
```

When the ddb(4) output style is selected, the first thing printed is the contents of the vm_map structure, followed by the individual map entries.

```
$ pmap -d
MAP 0xcf7cac84: [0x0->0xbfbfe000]
        #ent=8, sz=34041856, ref=1, version=20, flags=0x41
        pmap=0xcf44cee0(resident=<unknown>)
 - 0xcfa3a358: 0x8048000->0x80b1000: obj=0xcf45a8e8/0x0, amap=0x0/0
        submap=F, cow=T, nc=T, prot(max)=5/7, inh=1, wc=0, adv=0
...
```

The value of the flags field (in hexadecimal) is taken from the include file ⟨uvm/uvm_map.h⟩:

VM_MAP_PAGEABLE	0x01	entries are pageable
VM_MAP_INTRSAFE	0x02	interrupt safe map
VM_MAP_WIREFUTURE	0x04	future mappings are wired
VM_MAP_BUSY	0x08	map is busy
VM_MAP_WANTLOCK	0x10	want to write-lock
VM_MAP_DYING	0x20	map is being destroyed
VM_MAP_TOPDOWN	0x40	arrange map top-down

The “submap”, “cow”, and “nc” fields are true or false, and indicate whether the map is a submap, whether it is marked for copy on write, and whether it needs a copy. The “prot” (or protection) field, along with “max” (maximum protection allowed) are made up of the following flags from ⟨uvm/uvm_extern.h⟩:

UVM_PROT_READ	0x01	read allowed
UVM_PROT_WRITE	0x02	write allowed
UVM_PROT_EXEC	0x04	execute allowed

The “obj” and “amap” fields are pointers to, and offsets into, the underlying uvm_object or amap. The value for resident is always unknown because digging such information out of the kernel is beyond the scope of this application.

The two output styles that mirror the contents of the /proc file system appear as follows:

```
$ pmap -m
0x8048000 0x80b1000 r-x rwx COW NC 1 0 0
0x80b1000 0x80b3000 rw- rwx COW NC 1 0 0
0x80b3000 0x80ba000 rw- rwx COW NNC 1 0 0
0x80ba000 0x80be000 rwx rwx COW NNC 1 0 0
...

$ pmap -l
08048000-080b1000 r-xp 00000000 00:00 70173     /bin/sh
080b1000-080b3000 rw-p 00068000 00:00 70173     /bin/sh
080b3000-080ba000 rw-p 00000000 00:00 0
080ba000-080be000 rwxp 00000000 00:00 0
...
```

Here the protection and maximum protection values are indicated with ‘r’, ‘w’, and ‘x’ characters, indicating read permission, write permission, and execute permission, respectively. The “COW”, “NC”, and “NNC” values that follow indicate, again, that the map is marked for copy on write and either needs or does not need a copy. It is also possible to see the value “NCOW” here, which indicates that an entry will not be copied. The three following numbers indicate the inheritance type of the map, the wired count of the map, and any advice value assigned via madvise(2).

In the second form, the permissions indicated are followed by a ‘p’ or ‘s’ character indicating whether the map entry is private or shared (copy on write or not), and the numbers are the offset into the underlying object, the device and numbers of the object if it is a file, and the path to the file (if available).

As noted above (see section **DESCRIPTION**), the “all” output format is an amalgam of the previous output formats.

```
$ pmap -a
Start    End         Size  Offset   rwxpc  RWX  I/W/A ...
08048000-080b0fff     420k 00000000 r-xp+ (rwx) 1/0/0 ...
...
```

In this format, the column labeled “rwxpc” contains the permissions for the mapping along with the shared/private flag, and a character indicating whether the mapping needs to be copied on write (‘+’) or has already been copied (‘-’) and is followed by a column that indicates the maximum permissions for the map entry. The column labeled “I/W/A” indicates the inheritance, wired, and advice values for the map entry, as previously described. The pointer value at the end of the output line for entries backed by vnodes is the address of the vnode in question.

SEE ALSO

`ls`(1), `stat`(1), `madvise`(2), `mmap`(2), `kvm`(3), `ddb`(4), `mount_procfs`(8)

HISTORY

The **pmap** utility appeared in NetBSD 2.0.

AUTHORS

The **pmap** utility and documentation was written by Andrew Brown ⟨atatat@NetBSD.org⟩.

BUGS

Very little will work unless **pmap** is reading from the correct kernel in order to retrieve the proper symbol information.

Since processes can change state while **pmap** is running, some of the information printed may be inaccurate. This is especially important to consider when examining the kernel’s map, since merely executing **pmap** will cause some of the information to change.

The pathnames to files backing certain vnodes (such as the text and data sections of programs and shared libraries) are extracted from the kernel’s namei cache which is considerably volatile. If a path is not found there in its entirety, as much information as was available will be printed. In most cases, simply running `ls`(1) or `stat`(1) with the expected path to the file will cause the information to be reentered into the cache.

The Solaris command by the same name has some interesting command line flags that would be nice to emulate here. In particular, the **-r** option that lists a process’s reserved addresses, and the **-x** option that prints resident/shared/private mapping details for each entry.

Some of the output modes can be or are wider than the standard 80 columns of a terminal. Some sort of formatting might be nice.

SECURITY CONSIDERATIONS

The Solaris command controls access to processes the user does not own via the permissions of its `/proc` file system. Since **pmap** uses `kvm`(3) to read the requested data directly from kernel memory, no such limitation exists.

If any of the **-A**, **-E**, **-M**, **-N**, **-S**, or **-V** options are used, any extra privileges that **pmap** has will be dropped.

NAME

pmc — performance-monitoring counter interface for command execution

SYNOPSIS

pmc −h
pmc −C
pmc −c *event command* [*options . . .*]

DESCRIPTION

pmc is a means of using a processor's performance-monitoring counter (PMC) facility to measure various aspects of a program's execution. It is meant to be used in a fashion similar to `time`(1).

The arguments are as follows:

−h Display a list of performance counter events available on the system.

−C Cancel any performance counters that are currently running.

−c *event*
Count the event specified by *event* while running the command.

DIAGNOSTICS

PMC support is not compiled into the kernel Performance-monitoring counter support has not been compiled into the kernel. It may be included using the *PERFCTRS* option. See `options`(4) for details.

PMC counters are not supported by CPU Performance-monitoring counters are not available for the CPU.

SEE ALSO

`time`(1), `options`(4)

HISTORY

The **pmc** command first appeared in NetBSD 1.6.

AUTHORS

The **pmc** command was written by Frank van der Linden ⟨fvdl@wasabisystems.com⟩. The kernel support for reading performance counters on the i386 architecture was written by
Jason R. Thorpe ⟨thorpej@zembu.com⟩.

BUGS

The **pmc** command currently only supports performance-monitoring counters on the i386 architecture.

NAME

postalias − Postfix alias database maintenance

SYNOPSIS

postalias [**-Nfinoprsvw**] [**-c** *config_dir*] [**-d** *key*] [**-q** *key*]
[*file_type*:]*file_name* ...

DESCRIPTION

The **postalias**(1) command creates or queries one or more Postfix alias databases, or updates an existing one. The input and output file formats are expected to be compatible with Sendmail version 8, and are expected to be suitable for the use as NIS alias maps.

If the result files do not exist they will be created with the same group and other read permissions as their source file.

While a database update is in progress, signal delivery is postponed, and an exclusive, advisory, lock is placed on the entire database, in order to avoid surprises in spectator processes.

The format of Postfix alias input files is described in **aliases**(5).

By default the lookup key is mapped to lowercase to make the lookups case insensitive; as of Postfix 2.3 this case folding happens only with tables whose lookup keys are fixed-case strings such as btree:, dbm: or hash:. With earlier versions, the lookup key is folded even with tables where a lookup field can match both upper and lower case text, such as regexp: and pcre:. This resulted in loss of information with $*number* substitutions.

Options:

-c *config_dir*
: Read the **main.cf** configuration file in the named directory instead of the default configuration directory.

-d *key*
: Search the specified maps for *key* and remove one entry per map. The exit status is zero when the requested information was found.

  If a key value of **-** is specified, the program reads key values from the standard input stream. The exit status is zero when at least one of the requested keys was found.

-f
: Do not fold the lookup key to lower case while creating or querying a table.

  With Postfix version 2.3 and later, this option has no effect for regular expression tables. There, case folding is controlled by appending a flag to a pattern.

-i
: Incremental mode. Read entries from standard input and do not truncate an existing database. By default, **postalias**(1) creates a new database from the entries in *file_name*.

-N
: Include the terminating null character that terminates lookup keys and values. By default, **postalias**(1) does whatever is the default for the host operating system.

-n
: Don’t include the terminating null character that terminates lookup keys and values. By default, **postalias**(1) does whatever is the default for the host operating system.

-o
: Do not release root privileges when processing a non-root input file. By default, **postalias**(1) drops root privileges and runs as the source file owner instead.

-p
: Do not inherit the file access permissions from the input file when creating a new file. Instead, create a new file with default access permissions (mode 0644).

-q *key*
: Search the specified maps for *key* and write the first value found to the standard output stream. The exit status is zero when the requested information was found.

If a key value of - is specified, the program reads key values from the standard input stream and writes one line of *key: value* output for each key that was found. The exit status is zero when at least one of the requested keys was found.

-r When updating a table, do not complain about attempts to update existing entries, and make those updates anyway.

-s Retrieve all database elements, and write one line of *key: value* output for each element. The elements are printed in database order, which is not necessarily the same as the original input order. This feature is available in Postfix version 2.2 and later, and is not available for all database types.

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

-w When updating a table, do not complain about attempts to update existing entries, and ignore those attempts.

Arguments:

file_type

The database type. To find out what types are supported, use the "**postconf -m**" command.

The **postalias**(1) command can query any supported file type, but it can create only the following file types:

btree The output is a btree file, named *file_name***.db**. This is available on systems with support for **db** databases.

cdb The output is one file named *file_name***.cdb**. This is available on systems with support for **cdb** databases.

dbm The output consists of two files, named *file_name***.pag** and *file_name***.dir**. This is available on systems with support for **dbm** databases.

hash The output is a hashed file, named *file_name***.db**. This is available on systems with support for **db** databases.

sdbm The output consists of two files, named *file_name***.pag** and *file_name***.dir**. This is available on systems with support for **sdbm** databases.

When no *file_type* is specified, the software uses the database type specified via the **default_database_type** configuration parameter. The default value for this parameter depends on the host environment.

file_name

The name of the alias database source file when creating a database.

DIAGNOSTICS

Problems are logged to the standard error stream and to **syslogd**(8). No output means that no problems were detected. Duplicate entries are skipped and are flagged with a warning.

postalias(1) terminates with zero exit status in case of success (including successful "**postalias -q**" lookup) and terminates with non-zero exit status in case of failure.

ENVIRONMENT

MAIL_CONFIG

Directory with Postfix configuration files.

MAIL_VERBOSE

Enable verbose logging for debugging purposes.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

alias_database (see 'postconf -d' output)
The alias databases for **local**(8) delivery that are updated with "**newaliases**" or with "**sendmail -bi**".

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

berkeley_db_create_buffer_size (16777216)
The per-table I/O buffer size for programs that create Berkeley DB hash or btree tables.

berkeley_db_read_buffer_size (131072)
The per-table I/O buffer size for programs that read Berkeley DB hash or btree tables.

default_database_type (see 'postconf -d' output)
The default database type for use in **newaliases**(1), **postalias**(1) and **postmap**(1) commands.

syslog_facility (mail)
The syslog facility of Postfix logging.

syslog_name (postfix)
The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

STANDARDS

RFC 822 (ARPA Internet Text Messages)

SEE ALSO

aliases(5), format of alias database input file.
local(8), Postfix local delivery agent.
postconf(1), supported database types
postconf(5), configuration parameters
postmap(1), create/update/query lookup tables
newaliases(1), Sendmail compatibility interface.
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
DATABASE_README, Postfix lookup table overview

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postcat − show Postfix queue file contents

SYNOPSIS

postcat [**-oqv**] [**-c** *config_dir*] [*files*...]

DESCRIPTION

The **postcat**(1) command prints the contents of the named *files* in human-readable form. The files are expected to be in Postfix queue file format. If no *files* are specified on the command line, the program reads from standard input.

Options:

-c *config_dir*
: The **main.cf** configuration file is in the named directory instead of the default configuration directory.

-o
: Print the queue file offset of each record.

-q
: Search the Postfix queue for the named *files* instead of taking the names literally.

  Available in Postfix version 2.0 and later.

-v
: Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

DIAGNOSTICS

Problems are reported to the standard error stream.

ENVIRONMENT

MAIL_CONFIG
: Directory with Postfix configuration files.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

config_directory (see 'postconf -d' output)
: The default location of the Postfix main.cf and master.cf configuration files.

queue_directory (see 'postconf -d' output)
: The location of the Postfix top-level queue directory.

FILES

/var/spool/postfix, Postfix queue directory

SEE ALSO

postconf(5), Postfix configuration

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postconf − Postfix configuration utility

SYNOPSIS

postconf [**-dhnv**] [**-c** *config_dir*] [*parameter ...*]

postconf [**-aAmlv**] [**-c** *config_dir*]

postconf [**-ev**] [**-c** *config_dir*] [*parameter=value ...*]

postconf [**-btv**] [**-c** *config_dir*] [*template_file*]

DESCRIPTION

The **postconf**(1) command displays the actual values of configuration parameters, changes configuration parameter values, or displays other configuration information about the Postfix mail system.

Options:

-a List the available SASL server plug-in types. The SASL plug-in type is selected with the **smtpd_sasl_type** configuration parameter by specifying one of the names listed below.

cyrus This server plug-in is available when Postfix is built with Cyrus SASL support.

dovecot
This server plug-in requires the Dovecot authentication server.

This feature is available with Postfix 2.3 and later.

-A List the available SASL client plug-in types. The SASL plug-in type is selected with the **smtp_sasl_type** or **lmtp_sasl_type** configuration parameters by specifying one of the names listed below.

cyrus This client plug-in is available when Postfix is built with Cyrus SASL support.

This feature is available with Postfix 2.3 and later.

-b [*template_file*]
Display the message text that appears at the beginning of delivery status notification (DSN) messages, with $**name** expressions replaced by actual values. To override the built-in message text, specify a template file at the end of the command line, or specify a template file in main.cf with the **bounce_template_file** parameter. To force selection of the built-in message text templates, specify an empty template file name (in shell language: "").

This feature is available with Postfix 2.3 and later.

-c *config_dir*
The **main.cf** configuration file is in the named directory instead of the default configuration directory.

-d Print default parameter settings instead of actual settings.

-e Edit the **main.cf** configuration file. The file is copied to a temporary file then renamed into place. Parameters and values are specified on the command line. Use quotes in order to protect shell metacharacters and whitespace.

-h Show parameter values only, not the ‘‘name = ’’ label that normally precedes the value.

-l List the names of all supported mailbox locking methods. Postfix supports the following methods:

flock A kernel-based advisory locking method for local files only. This locking method is available on systems with a BSD compatible library.

fcntl A kernel-based advisory locking method for local and remote files.

dotlock
An application-level locking method. An application locks a file named *filename* by creating a file named *filename***.lock**. The application is expected to remove its own lock file, as well as stale lock files that were left behind after abnormal termination.

-m List the names of all supported lookup table types. In Postfix configuration files, lookup tables are specified as *type***:***name*, where *type* is one of the types listed below. The table *name* syntax depends on the lookup table type as described in the DATABASE_README document.

btree A sorted, balanced tree structure. This is available on systems with support for Berkeley DB databases.

cdb A read-optimized structure with no support for incremental updates. This is available on systems with support for CDB databases.

cidr A table that associates values with Classless Inter-Domain Routing (CIDR) patterns. This is described in **cidr_table**(5).

dbm An indexed file type based on hashing. This is available on systems with support for DBM databases.

environ
The UNIX process environment array. The lookup key is the variable name. Originally implemented for testing, someone may find this useful someday.

hash An indexed file type based on hashing. This is available on systems with support for Berkeley DB databases.

ldap (read-only)
Perform lookups using the LDAP protocol. This is described in **ldap_table**(5).

mysql (read-only)
Perform lookups using the MYSQL protocol. This is described in **mysql_table**(5).

pcre (read-only)
A lookup table based on Perl Compatible Regular Expressions. The file format is described in **pcre_table**(5).

pgsql (read-only)
Perform lookups using the PostgreSQL protocol. This is described in **pgsql_table**(5).

proxy (read-only)
A lookup table that is implemented via the Postfix **proxymap**(8) service. The table name syntax is *type***:***name*.

regexp (read-only)
A lookup table based on regular expressions. The file format is described in **regexp_table**(5).

sdbm An indexed file type based on hashing. This is available on systems with support for SDBM databases.

static (read-only)
A table that always returns its name as lookup result. For example, **static:foobar** always returns the string **foobar** as lookup result.

tcp (read-only)
Perform lookups using a simple request-reply protocol that is described in **tcp_table**(5). This feature is not included with the stable Postfix release.

unix (read-only)
A limited way to query the UNIX authentication database. The following tables are implemented:

unix:passwd.byname
The table is the UNIX password database. The key is a login name. The result is a password file entry in **passwd**(5) format.

unix:group.byname
The table is the UNIX group database. The key is a group name. The result is a group file entry in **group**(5) format.

Other table types may exist depending on how Postfix was built.

-n Print parameter settings that are not left at their built-in default value, because they are explicitly specified in main.cf.

-t [*template_file*]
Display the templates for delivery status notification (DSN) messages. To override the built-in templates, specify a template file at the end of the command line, or specify a template file in main.cf with the **bounce_template_file** parameter. To force selection of the built-in templates, specify an empty template file name (in shell language: "").

This feature is available with Postfix 2.3 and later.

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

DIAGNOSTICS

Problems are reported to the standard error stream.

ENVIRONMENT

MAIL_CONFIG
Directory with Postfix configuration files.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

bounce_template_file (empty)
Pathname of a configuration file with bounce message templates.

FILES

/etc/postfix/main.cf, Postfix configuration parameters

SEE ALSO

bounce(5), bounce template file format
postconf(5), configuration parameters

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
DATABASE_README, Postfix lookup table overview

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postdrop − Postfix mail posting utility

SYNOPSIS

postdrop [**-rv**] [**-c** *config_dir*]

DESCRIPTION

The **postdrop**(1) command creates a file in the **maildrop** directory and copies its standard input to the file.

Options:

-c *config_dir*
: The **main.cf** configuration file is in the named directory instead of the default configuration directory. See also the MAIL_CONFIG environment setting below.

-r
: Use a Postfix-internal protocol for reading the message from standard input, and for reporting status information on standard output. This is currently the only supported method.

-v
: Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose. As of Postfix 2.3, this option is available for the super-user only.

SECURITY

The command is designed to run with set-group ID privileges, so that it can write to the **maildrop** queue directory and so that it can connect to Postfix daemon processes.

DIAGNOSTICS

Fatal errors: malformed input, I/O error, out of memory. Problems are logged to **syslogd**(8) and to the standard error stream. When the input is incomplete, or when the process receives a HUP, INT, QUIT or TERM signal, the queue file is deleted.

ENVIRONMENT

MAIL_CONFIG
: Directory with the **main.cf** file. In order to avoid exploitation of set-group ID privileges, a non-standard directory is allowed only if:

  - The name is listed in the standard **main.cf** file with the **alternate_config_directories** configuration parameter.
  - The command is invoked by the super-user.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

alternate_config_directories (empty)
: A list of non-default Postfix configuration directories that may be specified with "-c config_directory" on the command line, or via the MAIL_CONFIG environment parameter.

config_directory (see 'postconf -d' output)
: The default location of the Postfix main.cf and master.cf configuration files.

import_environment (see 'postconf -d' output)
: The list of environment parameters that a Postfix process will import from a non-Postfix parent process.

queue_directory (see 'postconf -d' output)
: The location of the Postfix top-level queue directory.

syslog_facility (mail)
: The syslog facility of Postfix logging.

syslog_name (postfix)
: The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

trigger_timeout (10s)
The time limit for sending a trigger to a Postfix daemon (for example, the **pickup**(8) or **qmgr**(8) daemon).

Available in Postfix version 2.2 and later:

authorized_submit_users (static:anyone)
List of users who are authorized to submit mail with the **sendmail**(1) command (and with the privileged **postdrop**(1) helper command).

FILES

/var/spool/postfix/maildrop, maildrop queue

SEE ALSO

sendmail(1), compatibility interface
postconf(5), configuration parameters
syslogd(8), system logging

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postfix − Postfix control program

SYNOPSIS

postfix [**-Dv**] [**-c** *config_dir*] *command*

DESCRIPTION

This command is reserved for the superuser. To submit mail, use the Postfix **sendmail**(1) command.

The **postfix**(1) command controls the operation of the Postfix mail system: start or stop the **master**(8) daemon, do a health check, and other maintenance.

The **postfix**(1) command sets up a standardized environment and runs the **postfix-script** shell script to do the actual work.

The following commands are implemented:

check Warn about bad directory/file ownership or permissions, and create missing directories.

start Start the Postfix mail system. This also runs the configuration check described above.

stop Stop the Postfix mail system in an orderly fashion. If possible, running processes are allowed to terminate at their earliest convenience.

Note: in order to refresh the Postfix mail system after a configuration change, do not use the **start** and **stop** commands in succession. Use the **reload** command instead.

abort Stop the Postfix mail system abruptly. Running processes are signaled to stop immediately.

flush Force delivery: attempt to deliver every message in the deferred mail queue. Normally, attempts to deliver delayed mail happen at regular intervals, the interval doubling after each failed attempt.

Warning: flushing undeliverable mail frequently will result in poor delivery performance of all other mail.

reload Re-read configuration files. Running processes terminate at their earliest convenience.

set-permissions [*name=value ...***]**
Set the ownership and permissions of Postfix related files and directories, as specified in the **postfix-files** file.

Specify *name=value* to override and update specific main.cf configuration parameters. Use this, for example, to change the **mail_owner** or **setgid_group** setting for an already installed Postfix system.

This feature is available in Postfix 2.1 and later. With Postfix 2.0 and earlier, use "**/etc/postfix/post-install set-permissions**".

upgrade-configuration [*name=value ...***]**
Update the **main.cf** and **master.cf** files with information that Postfix needs in order to run: add or update services, and add or update configuration parameter settings.

Specify *name=value* to override and update specific main.cf configuration parameters.

This feature is available in Postfix 2.1 and later. With Postfix 2.0 and earlier, use "**/etc/postfix/post-install upgrade-configuration**".

The following options are implemented:

-c *config_dir*
Read the **main.cf** and **master.cf** configuration files in the named directory instead of the default configuration directory. Use this to distinguish between multiple Postfix instances on the same

host.

-D (with **postfix start** only)
Run each Postfix daemon under control of a debugger as specified via the **debugger_command** configuration parameter.

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

ENVIRONMENT

The **postfix**(1) command exports the following environment variables before executing the **postfix-script** file:

MAIL_CONFIG
This is set when the -c command-line option is present.

MAIL_VERBOSE
This is set when the -v command-line option is present.

MAIL_DEBUG
This is set when the -D command-line option is present.

CONFIGURATION PARAMETERS

The following **main.cf** configuration parameters are exported as environment variables with the same names:

command_directory (see 'postconf -d' output)
The location of all postfix administrative commands.

daemon_directory (see 'postconf -d' output)
The directory with Postfix support programs and daemon programs.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

queue_directory (see 'postconf -d' output)
The location of the Postfix top-level queue directory.

mail_owner (postfix)
The UNIX system account that owns the Postfix queue and most Postfix daemon processes.

setgid_group (postdrop)
The group ownership of set-gid Postfix commands and of group-writable Postfix directories.

sendmail_path (see 'postconf -d' output)
A Sendmail compatibility feature that specifies the location of the Postfix **sendmail**(1) command.

newaliases_path (see 'postconf -d' output)
Sendmail compatibility feature that specifies the location of the **newaliases**(1) command.

mailq_path (see 'postconf -d' output)
Sendmail compatibility feature that specifies where the Postfix **mailq**(1) command is installed.

html_directory (see 'postconf -d' output)
The location of Postfix HTML files that describe how to build, configure or operate a specific Postfix subsystem or feature.

manpage_directory (see 'postconf -d' output)
Where the Postfix manual pages are installed.

readme_directory (see 'postconf -d' output)
The location of Postfix README files that describe how to build, configure or operate a specific Postfix subsystem or feature.

Other configuration parameters:

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

import_environment (see 'postconf -d' output)
The list of environment parameters that a Postfix process will import from a non-Postfix parent process.

syslog_facility (mail)
The syslog facility of Postfix logging.

syslog_name (postfix)
The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

FILES

/etc/postfix/main.cf, Postfix configuration parameters
/etc/postfix/master.cf, Postfix daemon processes
/etc/postfix/postfix-files, file/directory permissions
/etc/postfix/postfix-script, administrative commands
/etc/postfix/post-install, post-installation configuration

SEE ALSO

Commands:
postalias(1), create/update/query alias database
postcat(1), examine Postfix queue file
postconf(1), Postfix configuration utility
postfix(1), Postfix control program
postkick(1), trigger Postfix daemon
postlock(1), Postfix-compatible locking
postlog(1), Postfix-compatible logging
postmap(1), Postfix lookup table manager
postqueue(1), Postfix mail queue control
postsuper(1), Postfix housekeeping
mailq(1), Sendmail compatibility interface
newaliases(1), Sendmail compatibility interface
sendmail(1), Sendmail compatibility interface

Postfix configuration:
bounce(5), Postfix bounce message templates
master(5), Postfix master.cf file syntax
postconf(5), Postfix main.cf file syntax

Table-driven mechanisms:
access(5), Postfix SMTP access control table
aliases(5), Postfix alias database
canonical(5), Postfix input address rewriting
generic(5), Postfix output address rewriting
header_checks(5), body_checks(5), Postfix content inspection
relocated(5), Users that have moved
transport(5), Postfix routing table
virtual(5), Postfix virtual aliasing

Table lookup mechanisms:
cidr_table(5), Associate CIDR pattern with value
ldap_table(5), Postfix LDAP client
mysql_table(5), Postfix MYSQL client
nisplus_table(5), Postfix NIS+ client

pcre_table(5), Associate PCRE pattern with value
pgsql_table(5), Postfix PostgreSQL client
regexp_table(5), Associate POSIX regexp pattern with value
tcp_table(5), Postfix client-server table lookup

Daemon processes:
anvil(8), Postfix connection/rate limiting
bounce(8), defer(8), trace(8), Delivery status reports
cleanup(8), canonicalize and enqueue message
discard(8), Postfix discard delivery agent
error(8), Postfix error delivery agent
flush(8), Postfix fast ETRN service
local(8), Postfix local delivery agent
master(8), Postfix master daemon
oqmgr(8), old Postfix queue manager
pickup(8), Postfix local mail pickup
pipe(8), deliver mail to non-Postfix command
proxymap(8), Postfix lookup table proxy server
qmgr(8), Postfix queue manager
qmqpd(8), Postfix QMQP server
scache(8), Postfix connection cache manager
showq(8), list Postfix mail queue
smtp(8), lmtp(8), Postfix SMTP+LMTP client
smtpd(8), Postfix SMTP server
spawn(8), run non-Postfix server
tlsmgr(8), Postfix TLS cache and randomness manager
trivial-rewrite(8), Postfix address rewriting
verify(8), Postfix address verification
virtual(8), Postfix virtual delivery agent

Other:
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
OVERVIEW, overview of Postfix commands and processes
BASIC_CONFIGURATION_README, Postfix basic configuration
ADDRESS_REWRITING_README, Postfix address rewriting
SMTPD_ACCESS_README, SMTP relay/access control
CONTENT_INSPECTION_README, Postfix content inspection
QSHAPE_README, Postfix queue analysis

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

TLS support by:
Lutz Jaenicke
Brandenburg University of Technology
Cottbus, Germany

Victor Duchovni
Morgan Stanley

SASL support originally by:
Till Franke
SuSE Rhein/Main AG
65760 Eschborn, Germany

LMTP support originally by:
Philip A. Prindeville
Mirapoint, Inc.
USA.

Amos Gouaux
University of Texas at Dallas
P.O. Box 830688, MC34
Richardson, TX 75083, USA

IPv6 support originally by:
Mark Huizer, Eindhoven University, The Netherlands
Jun-ichiro ’itojun’ Hagino, KAME project, Japan
The Linux PLD project
Dean Strik, Eindhoven University, The Netherlands

NAME

postkick − kick a Postfix service

SYNOPSIS

postkick [**-c** *config_dir*] [**-v**] *class service request*

DESCRIPTION

The **postkick**(1) command sends *request* to the specified *service* over a local transport channel. This command makes Postfix private IPC accessible for use in, for example, shell scripts.

Options:

-c *config_dir*
: Read the **main.cf** configuration file in the named directory instead of the default configuration directory.

-v
: Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

Arguments:

class
: Name of a class of local transport channel endpoints, either **public** (accessible by any local user) or **private** (administrative access only).

service
: The name of a local transport endpoint within the named class.

request
: A string. The list of valid requests is service-specific.

DIAGNOSTICS

Problems and transactions are logged to the standard error stream.

ENVIRONMENT

MAIL_CONFIG
: Directory with Postfix configuration files.

MAIL_VERBOSE
: Enable verbose logging for debugging purposes.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

config_directory (see 'postconf -d' output)
: The default location of the Postfix main.cf and master.cf configuration files.

application_event_drain_time (100s)
: How long the **postkick**(1) command waits for a request to enter the server's input buffer before giving up.

queue_directory (see 'postconf -d' output)
: The location of the Postfix top-level queue directory.

FILES

/var/spool/postfix/private, private class endpoints
/var/spool/postfix/public, public class endpoints

SEE ALSO

qmgr(8), queue manager trigger protocol
pickup(8), local pickup daemon
postconf(5), configuration parameters

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postlock − lock mail folder and execute command

SYNOPSIS

postlock [**-c** *config_dir*] **[-l** *lock_style***]**
[-v] *file command...*

DESCRIPTION

The **postlock**(1) command locks *file* for exclusive access, and executes *command*. The locking method is compatible with the Postfix UNIX-style local delivery agent.

Options:

-c *config_dir*
Read the **main.cf** configuration file in the named directory instead of the default configuration directory.

-l *lock_style*
Override the locking method specified via the **mailbox_delivery_lock** configuration parameter (see below).

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

Arguments:

file A mailbox file. The user should have read/write permission.

command...
The command to execute while *file* is locked for exclusive access. The command is executed directly, i.e. without interpretation by a shell command interpreter.

DIAGNOSTICS

The result status is 75 (EX_TEMPFAIL) when **postlock**(1) could not perform the requested operation. Otherwise, the exit status is the exit status from the command.

BUGS

With remote file systems, the ability to acquire a lock does not necessarily eliminate access conflicts. Avoid file access by processes running on different machines.

ENVIRONMENT

MAIL_CONFIG
Directory with Postfix configuration files.

MAIL_VERBOSE
Enable verbose logging for debugging purposes.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

LOCKING CONTROLS

deliver_lock_attempts (20)
The maximal number of attempts to acquire an exclusive lock on a mailbox file or **bounce**(8) logfile.

deliver_lock_delay (1s)
The time between attempts to acquire an exclusive lock on a mailbox file or **bounce**(8) logfile.

stale_lock_time (500s)
The time after which a stale exclusive mailbox lockfile is removed.

mailbox_delivery_lock (see 'postconf -d' output)
How to lock a UNIX-style **local**(8) mailbox before attempting delivery.

RESOURCE AND RATE CONTROLS

fork_attempts (5)
The maximal number of attempts to fork() a child process.

fork_delay (1s)
The delay between attempts to fork() a child process.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

SEE ALSO

postconf(5), configuration parameters

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postlog − Postfix-compatible logging utility

SYNOPSIS

postlog [**-iv**] [**-c** *config_dir*] [**-p** *priority***] [-t** *tag*] [*text...*]

DESCRIPTION

The **postlog**(1) command implements a Postfix-compatible logging interface for use in, for example, shell scripts.

By default, **postlog**(1) logs the *text* given on the command line as one record. If no *text* is specified on the command line, **postlog**(1) reads from standard input and logs each input line as one record.

Logging is sent to **syslogd**(8); when the standard error stream is connected to a terminal, logging is sent there as well.

The following options are implemented:

-c *config_dir*
: Read the **main.cf** configuration file in the named directory instead of the default configuration directory.

-i
: Include the process ID in the logging tag.

-p *priority*
: Specifies the logging severity: **info** (default), **warn**, **error**, **fatal**, or **panic**.

-t *tag*
: Specifies the logging tag, that is, the identifying name that appears at the beginning of each logging record. A default tag is used when none is specified.

-v
: Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

ENVIRONMENT

MAIL_CONFIG
: Directory with the **main.cf** file.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

config_directory (see 'postconf -d' output)
: The default location of the Postfix main.cf and master.cf configuration files.

syslog_facility (mail)
: The syslog facility of Postfix logging.

syslog_name (postfix)
: The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

SEE ALSO

postconf(5), configuration parameters
syslogd(8), syslog daemon

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704

Yorktown Heights, NY 10598, USA

NAME

postmap − Postfix lookup table management

SYNOPSIS

postmap [**-Nfinoprsvw**] [**-c** *config_dir*] [**-d** *key*] [**-q** *key*]
[*file_type*:]*file_name* ...

DESCRIPTION

The **postmap**(1) command creates or queries one or more Postfix lookup tables, or updates an existing one. The input and output file formats are expected to be compatible with:

makemap *file_type file_name* < *file_name*

If the result files do not exist they will be created with the same group and other read permissions as their source file.

While the table update is in progress, signal delivery is postponed, and an exclusive, advisory, lock is placed on the entire table, in order to avoid surprises in spectator processes.

INPUT FILE FORMAT

The format of a lookup table input file is as follows:

- A table entry has the form

  key whitespace *value*

- Empty lines and whitespace-only lines are ignored, as are lines whose first non-whitespace character is a '#'.
- A logical line starts with non-whitespace text. A line that starts with whitespace continues a logical line.

The *key* and *value* are processed as is, except that surrounding white space is stripped off. Unlike with Postfix alias databases, quotes cannot be used to protect lookup keys that contain special characters such as '#' or whitespace.

By default the lookup key is mapped to lowercase to make the lookups case insensitive; as of Postfix 2.3 this case folding happens only with tables whose lookup keys are fixed-case strings such as btree:, dbm: or hash:. With earlier versions, the lookup key is folded even with tables where a lookup field can match both upper and lower case text, such as regexp: and pcre:. This resulted in loss of information with $*number* substitutions.

COMMAND-LINE ARGUMENTS

-c *config_dir*
: Read the **main.cf** configuration file in the named directory instead of the default configuration directory.

-d *key*
: Search the specified maps for *key* and remove one entry per map. The exit status is zero when the requested information was found.

  If a key value of **-** is specified, the program reads key values from the standard input stream. The exit status is zero when at least one of the requested keys was found.

-f
: Do not fold the lookup key to lower case while creating or querying a table.

  With Postfix version 2.3 and later, this option has no effect for regular expression tables. There, case folding is controlled by appending a flag to a pattern.

-i
: Incremental mode. Read entries from standard input and do not truncate an existing database. By default, **postmap**(1) creates a new database from the entries in **file_name**.

-N Include the terminating null character that terminates lookup keys and values. By default, **postmap**(1) does whatever is the default for the host operating system.

-n Don't include the terminating null character that terminates lookup keys and values. By default, **postmap**(1) does whatever is the default for the host operating system.

-o Do not release root privileges when processing a non-root input file. By default, **postmap**(1) drops root privileges and runs as the source file owner instead.

-p Do not inherit the file access permissions from the input file when creating a new file. Instead, create a new file with default access permissions (mode 0644).

-q *key* Search the specified maps for *key* and write the first value found to the standard output stream. The exit status is zero when the requested information was found.

If a key value of **-** is specified, the program reads key values from the standard input stream and writes one line of *key value* output for each key that was found. The exit status is zero when at least one of the requested keys was found.

-r When updating a table, do not complain about attempts to update existing entries, and make those updates anyway.

-s Retrieve all database elements, and write one line of *key value* output for each element. The elements are printed in database order, which is not necessarily the same as the original input order. This feature is available in Postfix version 2.2 and later, and is not available for all database types.

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

-w When updating a table, do not complain about attempts to update existing entries, and ignore those attempts.

Arguments:

file_type

The database type. To find out what types are supported, use the "**postconf -m**" command.

The **postmap**(1) command can query any supported file type, but it can create only the following file types:

btree The output file is a btree file, named *file_name***.db**. This is available on systems with support for **db** databases.

cdb The output consists of one file, named *file_name***.cdb**. This is available on systems with support for **cdb** databases.

dbm The output consists of two files, named *file_name***.pag** and *file_name***.dir**. This is available on systems with support for **dbm** databases.

hash The output file is a hashed file, named *file_name***.db**. This is available on systems with support for **db** databases.

sdbm The output consists of two files, named *file_name***.pag** and *file_name***.dir**. This is available on systems with support for **sdbm** databases.

When no *file_type* is specified, the software uses the database type specified via the **default_database_type** configuration parameter.

file_name

The name of the lookup table source file when rebuilding a database.

DIAGNOSTICS

Problems are logged to the standard error stream and to **syslogd**(8). No output means that no problems were detected. Duplicate entries are skipped and are flagged with a warning.

postmap(1) terminates with zero exit status in case of success (including successful "**postmap -q**" lookup) and terminates with non-zero exit status in case of failure.

ENVIRONMENT

MAIL_CONFIG
Directory with Postfix configuration files.

MAIL_VERBOSE
Enable verbose logging for debugging purposes.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

berkeley_db_create_buffer_size (16777216)
The per-table I/O buffer size for programs that create Berkeley DB hash or btree tables.

berkeley_db_read_buffer_size (131072)
The per-table I/O buffer size for programs that read Berkeley DB hash or btree tables.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

default_database_type (see 'postconf -d' output)
The default database type for use in **newaliases**(1), **postalias**(1) and **postmap**(1) commands.

syslog_facility (mail)
The syslog facility of Postfix logging.

syslog_name (postfix)
The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

SEE ALSO

postalias(1), create/update/query alias database
postconf(1), supported database types
postconf(5), configuration parameters
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
DATABASE_README, Postfix lookup table overview

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postqueue − Postfix queue control

SYNOPSIS

postqueue [**-v**] [**-c** *config_dir*] **-f**
postqueue [**-v**] [**-c** *config_dir*] **-i** *queue_id*
postqueue [**-v**] [**-c** *config_dir*] **-p**
postqueue [**-v**] [**-c** *config_dir*] **-s** *site*

DESCRIPTION

The **postqueue**(1) command implements the Postfix user interface for queue management. It implements operations that are traditionally available via the **sendmail**(1) command. See the **postsuper**(1) command for queue operations that require super-user privileges such as deleting a message from the queue or changing the status of a message.

The following options are recognized:

-c *config_dir*
The **main.cf** configuration file is in the named directory instead of the default configuration directory. See also the MAIL_CONFIG environment setting below.

-f Flush the queue: attempt to deliver all queued mail.

This option implements the traditional "**sendmail -q**" command, by contacting the Postfix **qmgr**(8) daemon.

Warning: flushing undeliverable mail frequently will result in poor delivery performance of all other mail.

-i *queue_id*
Schedule immediate delivery of deferred mail with the specified queue ID.

This option implements the traditional **sendmail -qI** command, by contacting the **flush**(8) server.

This feature is available with Postfix version 2.4 and later.

-p Produce a traditional sendmail-style queue listing. This option implements the traditional **mailq** command, by contacting the Postfix **showq**(8) daemon.

Each queue entry shows the queue file ID, message size, arrival time, sender, and the recipients that still need to be delivered. If mail could not be delivered upon the last attempt, the reason for failure is shown. This mode of operation is implemented by executing the **postqueue**(1) command. The queue ID string is followed by an optional status character:

***** The message is in the **active** queue, i.e. the message is selected for delivery.

! The message is in the **hold** queue, i.e. no further delivery attempt will be made until the mail is taken off hold.

-s *site* Schedule immediate delivery of all mail that is queued for the named *site*. A numerical site must be specified as a valid RFC 2821 address literal enclosed in [], just like in email addresses. The site must be eligible for the "fast flush" service. See **flush**(8) for more information about the "fast flush" service.

This option implements the traditional "**sendmail -qR***site*" command, by contacting the Postfix **flush**(8) daemon.

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose. As of Postfix 2.3, this option is available for the super-user only.

SECURITY

This program is designed to run with set-group ID privileges, so that it can connect to Postfix daemon processes.

DIAGNOSTICS

Problems are logged to **syslogd**(8) and to the standard error stream.

ENVIRONMENT

MAIL_CONFIG

Directory with the **main.cf** file. In order to avoid exploitation of set-group ID privileges, a non-standard directory is allowed only if:

- The name is listed in the standard **main.cf** file with the **alternate_config_directories** configuration parameter.
- The command is invoked by the super-user.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

alternate_config_directories (empty)

A list of non-default Postfix configuration directories that may be specified with "-c config_directory" on the command line, or via the MAIL_CONFIG environment parameter.

config_directory (see 'postconf -d' output)

The default location of the Postfix main.cf and master.cf configuration files.

command_directory (see 'postconf -d' output)

The location of all postfix administrative commands.

fast_flush_domains ($relay_domains)

Optional list of destinations that are eligible for per-destination logfiles with mail that is queued to those destinations.

import_environment (see 'postconf -d' output)

The list of environment parameters that a Postfix process will import from a non-Postfix parent process.

queue_directory (see 'postconf -d' output)

The location of the Postfix top-level queue directory.

syslog_facility (mail)

The syslog facility of Postfix logging.

syslog_name (postfix)

The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

trigger_timeout (10s)

The time limit for sending a trigger to a Postfix daemon (for example, the **pickup**(8) or **qmgr**(8) daemon).

Available in Postfix version 2.2 and later:

authorized_flush_users (static:anyone)

List of users who are authorized to flush the queue.

authorized_mailq_users (static:anyone)

List of users who are authorized to view the queue.

FILES

/var/spool/postfix, mail queue

SEE ALSO

qmgr(8), queue manager
showq(8), list mail queue
flush(8), fast flush service
sendmail(1), Sendmail-compatible user interface
postsuper(1), privileged queue operations

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
ETRN_README, Postfix ETRN howto

LICENSE

The Secure Mailer license must be distributed with this software.

HISTORY

The postqueue command was introduced with Postfix version 1.1.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

postsuper − Postfix superintendent

SYNOPSIS

postsuper [**-psv**] [**-c** *config_dir*] [**-d** *queue_id*]
[**-h** *queue_id*] [**-H** *queue_id*]
[**-r** *queue_id*] [*directory ...*]

DESCRIPTION

The **postsuper**(1) command does maintenance jobs on the Postfix queue. Use of the command is restricted to the superuser. See the **postqueue**(1) command for unprivileged queue operations such as listing or flushing the mail queue.

By default, **postsuper**(1) performs the operations requested with the **-s** and **-p** command-line options on all Postfix queue directories - this includes the **incoming**, **active** and **deferred** directories with mail files and the **bounce**, **defer**, **trace** and **flush** directories with log files.

Options:

-c *config_dir*
The **main.cf** configuration file is in the named directory instead of the default configuration directory. See also the MAIL_CONFIG environment setting below.

-d *queue_id*
Delete one message with the named queue ID from the named mail queue(s) (default: **hold**, **incoming**, **active** and **deferred**).

If a *queue_id* of **-** is specified, the program reads queue IDs from standard input. For example, to delete all mail with exactly one recipient **user@example.com**:

```
mailq | tail +2 | grep -v '^ *(' | awk ´BEGIN { RS = "" }
  # $7=sender, $8=recipient1, $9=recipient2
  { if ($8 == "user@example.com" && $9 == "")
      print $1 }
´ | tr -d '*!' | postsuper -d -
```

Specify "**-d ALL**" to remove all messages; for example, specify "**-d ALL deferred**" to delete all mail in the **deferred** queue. As a safety measure, the word **ALL** must be specified in upper case.

Warning: Postfix queue IDs are reused. There is a very small possibility that postsuper deletes the wrong message file when it is executed while the Postfix mail system is delivering mail.

The scenario is as follows:

1) The Postfix queue manager deletes the message that **postsuper**(1) is asked to delete, because Postfix is finished with the message (it is delivered, or it is returned to the sender).

2) New mail arrives, and the new message is given the same queue ID as the message that **postsuper**(1) is supposed to delete. The probability for reusing a deleted queue ID is about 1 in 2**15 (the number of different microsecond values that the system clock can distinguish within a second).

3) **postsuper**(1) deletes the new message, instead of the old message that it should have deleted.

-h *queue_id*
Put mail "on hold" so that no attempt is made to deliver it. Move one message with the named queue ID from the named mail queue(s) (default: **incoming**, **active** and **deferred**) to the **hold**

queue.

If a *queue_id* of **-** is specified, the program reads queue IDs from standard input.

Specify "**-h ALL**" to hold all messages; for example, specify "**-h ALL deferred**" to hold all mail in the **deferred** queue. As a safety measure, the word **ALL** must be specified in upper case.

Note: while mail is "on hold" it will not expire when its time in the queue exceeds the **maximal_queue_lifetime** or **bounce_queue_lifetime** setting. It becomes subject to expiration after it is released from "hold".

-H *queue_id*
Release mail that was put "on hold". Move one message with the named queue ID from the named mail queue(s) (default: **hold**) to the **deferred** queue.

If a *queue_id* of **-** is specified, the program reads queue IDs from standard input.

Note: specify "**postsuper -r**" to release mail that was kept on hold for a significant fraction of **$maximal_queue_lifetime** or **$bounce_queue_lifetime**, or longer.

Specify "**-H ALL**" to release all mail that is "on hold". As a safety measure, the word **ALL** must be specified in upper case.

-p Purge old temporary files that are left over after system or software crashes.

-r *queue_id*
Requeue the message with the named queue ID from the named mail queue(s) (default: **hold**, **incoming**, **active** and **deferred**). To requeue multiple messages, specify multiple **-r** command-line options.

Alternatively, if a *queue_id* of **-** is specified, the program reads queue IDs from standard input.

Specify "**-r ALL**" to requeue all messages. As a safety measure, the word **ALL** must be specified in upper case.

A requeued message is moved to the **maildrop** queue, from where it is copied by the **pickup**(8) and **cleanup**(8) daemons to a new queue file. In many respects its handling differs from that of a new local submission.

- The message is not subjected to the smtpd_milters or non_smtpd_milters settings. When mail has passed through an external content filter, this would produce incorrect results with Milter applications that depend on original SMTP connection state information.
- The message is subjected again to mail address rewriting and substitution. This is useful when rewriting rules or virtual mappings have changed.

  The address rewriting context (local or remote) is the same as when the message was received.
- The message is subjected to the same content_filter settings (if any) as used for new local mail submissions. This is useful when content_filter settings have changed.

Warning: Postfix queue IDs are reused. There is a very small possibility that **postsuper**(1) requeues the wrong message file when it is executed while the Postfix mail system is running, but no harm should be done.

-s Structure check and structure repair. This should be done once before Postfix startup.

- Rename files whose name does not match the message file inode number. This operation is necessary after restoring a mail queue from a different machine, or from backup media.

- Move queue files that are in the wrong place in the file system hierarchy and remove subdirectories that are no longer needed. File position rearrangements are necessary after a change in the **hash_queue_names** and/or **hash_queue_depth** configuration parameters.

-v Enable verbose logging for debugging purposes. Multiple **-v** options make the software increasingly verbose.

DIAGNOSTICS

Problems are reported to the standard error stream and to **syslogd**(8).

postsuper(1) reports the number of messages deleted with **-d**, the number of messages requeued with **-r**, and the number of messages whose queue file name was fixed with **-s**. The report is written to the standard error stream and to **syslogd**(8).

ENVIRONMENT

MAIL_CONFIG
: Directory with the **main.cf** file.

BUGS

Mail that is not sanitized by Postfix (i.e. mail in the **maildrop** queue) cannot be placed "on hold".

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

config_directory (see 'postconf -d' output)
: The default location of the Postfix main.cf and master.cf configuration files.

hash_queue_depth (1)
: The number of subdirectory levels for queue directories listed with the hash_queue_names parameter.

hash_queue_names (deferred, defer)
: The names of queue directories that are split across multiple subdirectory levels.

queue_directory (see 'postconf -d' output)
: The location of the Postfix top-level queue directory.

syslog_facility (mail)
: The syslog facility of Postfix logging.

syslog_name (postfix)
: The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

SEE ALSO

sendmail(1), Sendmail-compatible user interface
postqueue(1), unprivileged queue operations

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

pr — print files

SYNOPSIS

pr [+*page*] [**-***column*] [**-adFmrt**] [[**-e**] [*char*] [*gap*]] [**-h** *header*] [[**-i**] [*char*] [*gap*]] [**-l** *lines*] [**-o** *offset*] [[**-s**] [*char*]] [**-T** *timefmt*] [[**-n**] [*char*] [*width*]] [**-w** *width*] [-] [*file ...*]

DESCRIPTION

The **pr** utility is a printing and pagination filter for text files. When multiple input files are specified, each is read, formatted, and written to standard output. By default, the input is separated into 66-line pages, each with

- A 5-line header with the page number, date, time, and the pathname of the file.
- A 5-line trailer consisting of blank lines.

If standard output is associated with a terminal, diagnostic messages are suppressed until the **pr** utility has completed processing.

When multiple column output is specified, text columns are of equal width. By default text columns are separated by at least one <*blank*>. Input lines that do not fit into a text column are truncated. Lines are not truncated under single column output.

OPTIONS

In the following option descriptions, column, lines, offset, page, and width are positive decimal integers and gap is a nonnegative decimal integer.

+*page*
: Begin output at page number *page* of the formatted input.

-*column*
: Produce output that is *columns* wide (default is 1) that is written vertically down each column in the order in which the text is received from the input file. The options **-e** and **-i** are assumed. This option should not be used with **-m**. When used with **-t**, the minimum number of lines is used to display the output.

-a
: Modify the effect of the **-column** option so that the columns are filled across the page in a round-robin order (e.g., when column is 2, the first input line heads column 1, the second heads column 2, the third is the second line in column 1, etc.). This option requires the use of the **-column** option.

-d
: Produce output that is double spaced. An extra <*newline*> character is output following every <newline> found in the input.

-e *[char][gap]*
: Expand each input <tab> to the next greater column position specified by the formula *n*∗*gap*+*1*, where *n* is an integer > 0. If *gap* is zero or is omitted the default is 8. All <*tab*> characters in the input are expanded into the appropriate number of <*space*>*s*. If any nondigit character, *char*, is specified, it is used as the input tab character.

-F
: Use a <*form-feed*> character for new pages, instead of the default behavior that uses a sequence of <*newline*> characters.

-h *header*
: Use the string *header* to replace the *file name* in the header line.

-i *[char][gap]*
In output, replace multiple <space>s with <tab>s whenever two or more adjacent <space>s reach column positions *gap+1*, *2***gap+1*, etc. If *gap* is zero or omitted, default <*tab*> settings at every eighth column position is used. If any nondigit character, *char*, is specified, it is used as the output <*tab*> character.

-l *lines*
Override the 66 line default and reset the page length to *lines*. If *lines* is not greater than the sum of both the header and trailer depths (in lines), the **pr** utility suppresses output of both the header and trailer, as if the **-t** option were in effect.

-m Merge the contents of multiple files. One line from each file specified by a file operand is written side by side into text columns of equal fixed widths, in terms of the number of column positions. The number of text columns depends on the number of file operands successfully opened. The maximum number of files merged depends on page width and the per process open file limit. The options **-e** and **-i** are assumed.

-n *[char][width]*
Provide *width* digit line numbering. The default for *width*, if not specified, is 5. The number occupies the first *width* column positions of each text column or each line of **-m** output. If *char* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows. The default for *char* is a <*tab*>. Line numbers longer than *width* columns are truncated.

-o *offset*
Each line of output is preceded by *offset* <*spaces*>*s*. If the **-o** option is not specified, the default is zero. The space taken is in addition to the output line width.

-r Write no diagnostic reports on failure to open a file.

-s *char*
Separate text columns by the single character *char* instead of by the appropriate number of <*space*>*s* (default for *char* is the <*tab*> character).

-T Specify an `strftime`(3) format string to be used to format the date and time information in the page header.

-t Print neither the five-line identifying header nor the five-line trailer usually supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.

-w *width*
Set the width of the line to *width* column positions for multiple text-column output only. If the **-w** option is not specified and the **-s** option is not specified, the default width is 72. If the **-w** option is not specified and the **-s** option is specified, the default width is 512.

file
A pathname of a file to be printed. If no *file* operands are specified, or if a *file* operand is '**-**', the standard input is used. The standard input is used only if no *file* operands are specified, or if a *file* operand is '**-**'.

The **-s** option does not allow the option letter to be separated from its argument, and the options **-e**, **-i**, and **-n** require that both arguments, if present, not be separated from the option letter.

ERRORS

If **pr** receives an interrupt while printing to a terminal, it flushes all accumulated error messages to the screen before terminating.

The **pr** utility exits 0 on success, and 1 if an error occurs.

Error messages are written to standard error during the printing process (if output is redirected) or after all successful file printing is complete (when printing to a terminal).

SEE ALSO

cat(1), more(1), strftime(3)

STANDARDS

The **pr** utility is IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

printenv — print out the environment

SYNOPSIS

printenv [*name*]

DESCRIPTION

printenv prints out the names and values of the variables in the environment, with one name/value pair per line. If *name* is specified, only its value is printed.

If a *name* is specified and it is not defined in the environment, **printenv** returns exit status 1, else it returns status 0.

SEE ALSO

csh(1), sh(1), environ(7)

HISTORY

The **printenv** command appeared in 3.0 BSD.

NAME

printf — formatted output

SYNOPSIS

printf *format* [*arguments ...*]

DESCRIPTION

printf formats and prints its arguments, after the first, under control of the *format*. The *format* is a character string which contains three types of objects: plain characters, which are simply copied to standard output, character escape sequences which are converted and copied to the standard output, and format specifications, each of which causes printing of the next successive *argument*.

The *arguments* after the first are treated as strings if the corresponding format is either **b**, **B**, **c**, or **s**; otherwise it is evaluated as a C constant, with the following extensions:

- A leading plus or minus sign is allowed.
- If the leading character is a single or double quote, the value is the ASCII code of the next character.

The format string is reused as often as necessary to satisfy the *arguments*. Any extra format specifications are evaluated with zero or the null string.

Character escape sequences are in backslash notation as defined in ANSI X3.159-1989 ("ANSI C89"). The characters and their meanings are as follows:

\e Write an ⟨escape⟩ character.

\a Write a ⟨bell⟩ character.

\b Write a ⟨backspace⟩ character.

\f Write a ⟨form-feed⟩ character.

\n Write a ⟨new-line⟩ character.

\r Write a ⟨carriage return⟩ character.

\t Write a ⟨tab⟩ character.

\v Write a ⟨vertical tab⟩ character.

\´ Write a ⟨single quote⟩ character.

\" Write a ⟨double quote⟩ character.

**** Write a backslash character.

*****num* Write an 8−bit character whose ASCII value is the 1−, 2−, or 3−digit octal number *num*.

\x*xx* Write an 8−bit character whose ASCII value is the 1− or 2−digit hexadecimal number *xx*.

Each format specification is introduced by the percent character ("%"). The remainder of the format specification includes, in the following order:

Zero or more of the following flags:

A '#' character specifying that the value should be printed in an "alternative form". For **b**, **c**, **d**, and **s** formats, this option has no effect. For the **o** format the precision of the number is increased to force the first character of the output string to a zero. For the **x** (**X**) format, a non-zero result has the string 0x (0X) prepended to it. For **e**, **E**, **f**, **g**, and **G** formats, the result will always contain a decimal point, even if no digits follow the point (normally, a decimal point only appears in the results of those formats if a digit follows the decimal

point). For **g** and **G** formats, trailing zeros are not removed from the result as they would otherwise be.

- A minus sign '−' which specifies *left adjustment* of the output in the indicated field;

+ A '+' character specifying that there should always be a sign placed before the number when using signed formats.

' ' A space specifying that a blank should be left before a positive number for a signed format. A '+' overrides a space if both are used;

0 A zero '0' character indicating that zero-padding should be used rather than blank-padding. A '−' overrides a '0' if both are used;

Field Width:

An optional digit string specifying a *field width*; if the output string has fewer characters than the field width it will be blank-padded on the left (or right, if the left-adjustment indicator has been given) to make up the field width (note that a leading zero is a flag, but an embedded zero is part of a field width);

Precision:

An optional period, '**.**', followed by an optional digit string giving a *precision* which specifies the number of digits to appear after the decimal point, for **e** and **f** formats, or the maximum number of characters to be printed from a string (**b**, **B**, and **s** formats); if the digit string is missing, the precision is treated as zero;

Format:

A character which indicates the type of format to use (one of **diouxXfwEgGbBcs**).

A field width or precision may be '*' instead of a digit string. In this case an *argument* supplies the field width or precision.

The format characters and their meanings are:

diouXx The *argument* is printed as a signed decimal (d or i), unsigned octal, unsigned decimal, or unsigned hexadecimal (X or x), respectively.

f The *argument* is printed in the style [−]ddd**.**ddd where the number of d's after the decimal point is equal to the precision specification for the argument. If the precision is missing, 6 digits are given; if the precision is explicitly 0, no digits and no decimal point are printed.

eE The *argument* is printed in the style [−]d**.**ddd**e**±dd where there is one digit before the decimal point and the number after is equal to the precision specification for the argument; when the precision is missing, 6 digits are produced. An upper-case E is used for an 'E' format.

gG The *argument* is printed in style **f** or in style **e** (**E**) whichever gives full precision in minimum space.

b Characters from the string *argument* are printed with backslash-escape sequences expanded. The following additional backslash-escape sequences are supported:

\c Causes **printf** to ignore any remaining characters in the string operand containing it, any remaining string operands, and any additional characters in the format operand.

\0*num* Write an 8−bit character whose ASCII value is the 1−, 2−, or 3−digit octal number *num*.

\^*c* Write the control character *c*. Generates characters '\000' through '\037', and '\177' (from '\^?').

\M-*c* Write the character *c* with the 8th bit set. Generates characters '\241' through '\376'.

\M^*c* Write the control character *c* with the 8th bit set. Generates characters '\000' through '\037', and '\177' (from '\M^?').

B Characters from the string *argument* are printed with unprintable characters backslash-escaped using the '*****c*', '**\^***c*', '**\M-***c*' or '**\M^***c*', formats described above.

c The first character of *argument* is printed.

s Characters from the string *argument* are printed until the end is reached or until the number of characters indicated by the precision specification is reached; if the precision is omitted, all characters in the string are printed.

% Print a '%'; no argument is used.

In no case does a non-existent or small field width cause truncation of a field; padding takes place only if the specified field width exceeds the actual width.

EXIT STATUS

printf exits 0 on success, 1 on failure.

SEE ALSO

echo(1), printf(3), vis(3), printf(9)

STANDARDS

The **printf** utility conforms to IEEE Std 1003.1-2001 ("POSIX.1").

Support for the floating point formats and '*' as a field width and precision are optional in POSIX.

The behaviour of the %B format and the \', \", \xxx, \e and \[M][-|^]c escape sequences are undefined in POSIX.

BUGS

Since the floating point numbers are translated from ASCII to floating-point and then back again, floating-point precision may be lost.

Hexadecimal character constants are restricted to, and should be specified as, two character constants. This is contrary to the ISO C standard but does guarantee detection of the end of the constant.

NOTES

All formats which treat the *argument* as a number first convert the *argument* from its external representation as a character string to an internal numeric representation, and then apply the format to the internal numeric representation, producing another external character string representation. One might expect the **%c** format to do likewise, but in fact it does not.

To convert a string representation of a decimal, octal, or hexadecimal number into the corresponding character, two nested **printf** invocations may be used, in which the inner invocation converts the input to an octal string, and the outer invocation uses the octal string as part of a format. For example, the following command outputs the character whose code is 0x0A, which is a newline in ASCII:

```
printf "$(printf "\\%o" "0x0A")"
```

NAME

progress — feed input to a command, displaying a progress bar

SYNOPSIS

progress [**-ez**] [**-b** *buffersize*] [**-f** *file*] [**-l** *length*] [**-p** *prefix*] *cmd* [*args ...*]

DESCRIPTION

The **progress** utility opens a pipe to *cmd* and feeds an input stream into it, while displaying a progress bar to standard output. If no filename is specified, **progress** reads from standard input. Where feasible, **progress** `fstat`(2)s the input to determine the length, so a time estimate can be calculated.

If no length is specified or determined, **progress** simply displays a count of the data and the data rate.

The options are as follows:

-b *buffersize*
: Read in buffers of the specified size (default 64k). An optional suffix (per `strsuftoll`(3)) may be given.

-e
: Display progress to standard error instead of standard output.

-f *file*
: Read from the specified *file* instead of standard input.

-l *length*
: Use the specified length for the time estimate, rather than attempting to `fstat`(2) the input. An optional suffix (per `strsuftoll`(3)) may be given.

-p *prefix*
: Print the given "prefix" text before (left of) the progress bar.

-z
: Filter the input through `gunzip`(1). If **-f** is specified, calculate the length using **gzip -l**.

EXIT STATUS

progress exits 0 on success.

EXAMPLES

The command

```
progress -zf file.tar.gz tar xf -
```

will extract the `file.tar.gz` displaying the progress bar as time passes:

```
  0% |                                |     0        0.00 KiB/s    --:-- ETA
 40% |********                        |   273 KiB  271.95 KiB/s    00:01 ETA
 81% |**********************          |   553 KiB  274.61 KiB/s    00:00 ETA
100% |*****************************|   680 KiB  264.59 KiB/s    00:00 ETA
```

If it is preferred to monitor the progress of the decompression process (unlikely), then

```
progress -f file.tar.gz tar zxf -
```

could be used.

The command

```
dd if=/dev/rwd0d ibs=64k | \
progress -l 120g dd of=/dev/rwd1d obs=64k
```

will copy the 120 GiB disk **wd0** (`/dev/rwd0d`) to **wd1** (`/dev/rwd1d`), displaying a progress bar during the operation.

SEE ALSO

`ftp`(1), `strsuftoll`(3)

HISTORY

`progress` first appeared in NetBSD 1.6.1. The dynamic progress bar display code is part of `ftp`(1).

AUTHORS

`progress` was written by John Hawkinson ⟨jhawk@NetBSD.org⟩. `ftp`(1)'s dynamic progress bar was written by Luke Mewburn.

BUGS

Since the progress bar is displayed asynchronously, it may be difficult to read some error messages, both those produced by the pipeline, as well as those produced by **`progress`** itself.

NAME

ps — process status

SYNOPSIS

ps [**-acCehjlmrsSTuvwx**] [**-k** *key*] [**-M** *core*] [**-N** *system*] [**-O** *fmt*] [**-o** *fmt*] [**-p** *pid*] [**-t** *tty*] [**-U** *username*] [**-W** *swap*]
ps [**-L**]

DESCRIPTION

ps displays a header line followed by lines containing information about your processes that have controlling terminals. The default sort order of controlling terminal and (among processes with the same controlling terminal) process ID may be changed using the **-k**, **-m**, or **-r** options.

The information displayed is selected based on a set of keywords (see the **-L** **-O** and **-o** options). The default output format includes, for each process, the process' ID, controlling terminal, CPU time (including both user and system time), state, and associated command.

The options are as follows:

-a Display information about other users' processes as well as your own.

-c Do not display full command with arguments, but only the executable name. This may be somewhat confusing; for example, all sh(1) scripts will show as "sh".

-C Change the way the CPU percentage is calculated by using a "raw" CPU calculation that ignores "resident" time (this normally has no effect).

-e Display the environment as well. The environment for other users' processes can only be displayed by the super-user.

-h Repeat the information header as often as necessary to guarantee one header per page of information.

-j Print information associated with the following keywords: *user*, *pid*, *ppid*, *pgid*, *sess*, *jobc*, *state*, *tt*, *time*, and *command*.

-k Sort the output using the space or comma separated list of keywords. Multiple sort keys may be specified, using any of the **-k**, **-m**, or **-r** options. The default sort order is equivalent to **-k** *tdev,pid*.

-L List the set of available keywords.

-l Display information associated with the following keywords: *uid*, *pid*, *ppid*, *cpu*, *pri*, *nice*, *vsz*, *rss*, *wchan*, *state*, *tt*, *time*, and *command*.

-M Extract values from the specified core file instead of the running system.

-m Sort by memory usage, equivalent to **-k** *vsz*.

-N Extract the name list from the specified system instead of the default "/netbsd". Ignored unless **-M** is specified.

-O Display information associated with the space or comma separated list of keywords specified. The **-O** option does not suppress the default display; it inserts additional keywords just after the *pid* keyword in the default display, or after the *pid* keyword (if any) in a non-default display specified before the first use of the **-O** flag. Keywords inserted by multiple **-O** options will be adjacent.

An equals sign ("=") followed by a customised header string may be appended to a keyword, as described in more detail under the **-o** option.

-o Display information associated with the space or comma separated list of keywords specified. Use of the **-o** option suppresses the set of keywords that would be displayed by default, or appends to the set of keywords specified by other options.

An equals sign ("=") followed by a customised header string may be appended to a keyword. This causes the printed header to use the specified string instead of the default header associated with the keyword.

Everything after the first equals sign is part of the customised header text, and this may include embedded spaces (" "), commas (","), or equals signs ("="). To specify multiple keywords with customised headers, use multiple **-o** or **-O** options.

If all the keywords to be displayed have customised headers, and all the customised headers are entirely empty, then the header line is not printed at all.

-p Display information associated with the specified process ID.

-r Sort by current CPU usage. This is equivalent to **-k** *%cpu*.

-S Change the way the process time is calculated by summing all exited children to their parent process.

-s Display one line for each LWP, rather than one line for each process, and display information associated with the following keywords: *uid*, *pid*, *ppid*, *cpu*, *lid*, *nlwp*, *pri*, *nice*, *vsz*, *rss*, *wchan*, *lstate*, *tt*, *time*, and *command*.

-T Display information about processes attached to the device associated with the standard input.

-t Display information about processes attached to the specified terminal device. Use an question mark ("?") for processes not attached to a terminal device and a minus sign ("-") for processes that have been revoked from their terminal device.

-U Displays processes belonging to the user whose username or uid has been given to the **-U** switch.

-u Display information associated with the following keywords: *user*, *pid*, *%cpu*, *%mem*, *vsz*, *rss*, *tt*, *state*, *start*, *time*, and *command*. The **-u** option implies the **-r** option.

-v Display information associated with the following keywords: *pid*, *state*, *time*, *sl*, *re*, *pagein*, *vsz*, *rss*, *lim*, *tsiz*, *%cpu*, *%mem*, and *command*. The **-v** option implies the **-m** option.

-W Extract swap information from the specified file instead of the default "/dev/drum". Ignored unless **-M** is specified.

-w Use 132 columns to display information, instead of the default which is your window size. If the **-w** option is specified more than once, **ps** will use as many columns as necessary without regard for your window size.

-x Also display information about processes without controlling terminals.

A complete list of the available keywords are listed below. Some of these keywords are further specified as follows:

%cpu The CPU utilization of the process; this is a decaying average over up to a minute of previous (real) time. Since the time base over which this is computed varies (since processes may be very young) it is possible for the sum of all %CPU fields to exceed 100%.

%mem The percentage of real memory used by this process.

flags The flags (in hexadecimal) associated with the process as in the include file ⟨sys/proc.h⟩:

P_ADVLOCK	0x00000001	process may hold a POSIX advisory lock
P_CONTROLT	0x00000002	process has a controlling terminal
P_INMEM	0x00000004	process is loaded into memory
P_NOCLDSTOP	0x00000008	no SIGCHLD when children stop
P_PPWAIT	0x00000010	parent is waiting for child to exec/exit
P_PROFIL	0x00000020	process has started profiling
P_SELECT	0x00000040	selecting; wakeup/waiting danger
P_SINTR	0x00000080	sleep is interruptible
P_SUGID	0x00000100	process had set id privileges since last exec
P_SYSTEM	0x00000200	system process: no sigs, stats or swapping
P_TIMEOUT	0x00000400	timing out during sleep
P_TRACED	0x00000800	process is being traced
P_WAITED	0x00001000	debugging process has waited for child
P_WEXIT	0x00002000	working on exiting
P_EXEC	0x00004000	process called execve(2)
P_OWEUPC	0x00008000	owe process an addupc() call at next ast
P_FSTRACE	0x00010000	tracing via file system
P_NOCLDWAIT	0x00020000	no zombies when children die
P_32	0x00040000	32-bit process (used on 64-bit kernels)
P_BIGLOCK	0x00080000	process needs kernel ‘‘big lock’’ to run
P_INEXEC	0x00100000	process is exec’ing and cannot be traced
P_SYSTRACE	0x00200000	process system call tracing active

lim The soft limit on memory used, specified via a call to setrlimit(2).

lstart The exact time the command started, using the “%C” format described in strftime(3).

nice The process scheduling increment (see setpriority(2)).

rss the real memory (resident set) size of the process (in 1024 byte units).

start The time the command started. If the command started less than 24 hours ago, the start time is displayed using the “%l:%M%p” format described in strftime(3). If the command started less than 7 days ago, the start time is displayed using the “%a%p” format. Otherwise, the start time is displayed using the “%e%b%y” format.

state The state is given by a sequence of letters, for example, “RWNA”. The first letter indicates the run state of the process:

D	Marks a process in disk (or other short term, uninterruptible) wait.
I	Marks a process that is idle (sleeping for longer than about 20 seconds).
R	Marks a runnable process.
S	Marks a process that is sleeping for less than about 20 seconds.
T	Marks a stopped process.
U	Marks a suspended process.
Z	Marks a dead process (a “zombie”).

Additional characters after these, if any, indicate additional state information:

+	The process is in the foreground process group of its control terminal.
-	The LWP is detached (can’t be waited for).
<	The process has raised CPU scheduling priority.

a	The process is using scheduler activations.
E	The process is trying to exit.
K	The process is a kernel thread or system process.
L	The process has pages locked in core (for example, for raw I/O).
l	The process has multiple LWPs.
N	The process has reduced CPU scheduling priority (see `setpriority`(2)).
s	The process is a session leader.
V	The process is suspended during a `vfork`(2).
W	The process is swapped out.
X	The process is being traced or debugged.

tt An abbreviation for the pathname of the controlling terminal, if any. The abbreviation consists of the two letters following "`/dev/tty`", or, for the console, "co". This is followed by a "-" if the process can no longer reach that controlling terminal (i.e., it has been revoked).

wchan The event (an address in the system) on which a process waits. When printed numerically, the initial part of the address is trimmed off and the result is printed in hex, for example, 0x80324000 prints as 324000.

When printing using the *command* keyword, a process that has exited and has a parent that has not yet waited for the process (in other words, a zombie) is listed as "⟨defunct⟩", and a process which is blocked while trying to exit is listed as "⟨exiting⟩".

ps will try to locate the processes' argument vector from the user area in order to print the command name and arguments. This method is not reliable because a process is allowed to destroy this information. The *ucomm* (accounting) keyword will always contain the real command name as contained in the process structure's *p_comm* field.

If the command vector cannot be located (usually because it has not been set, as is the case of system processes and/or kernel threads) the command name is printed within square brackets.

To indicate that the argument vector has been tampered with, **ps** will append the real command name to the output within parentheses if the basename of the first argument in the argument vector does not match the contents of the real command name.

In addition, **ps** checks for the following two situations and does not append the real command name parenthesized:

-shellname
: The login process traditionally adds a '-' in front of the shell name to indicate a login shell. **ps** will not append parenthesized the command name if it matches with the name in the first argument of the argument vector, skipping the leading '-'.

daemonname: current-activity
: Daemon processes frequently report their current activity by setting their name to be like "daemonname: current-activity". **ps** will not append parenthesized the command name, if the string preceding the ':' in the first argument of the argument vector matches the command name.

KEYWORDS

The following is a complete list of the available keywords and their meanings. Several of them have aliases (keywords which are synonyms).

%cpu	percentage CPU usage (alias *pcpu*)
%mem	percentage memory usage (alias *pmem*)

acflag	accounting flag (alias *acflg*)
comm	command (the argv[0] value)
command	command and arguments (alias *args*)
cpu	short-term CPU usage factor (for scheduling)
cpuid	CPU number the current process or lwp is running on.
ctime	accumulated CPU time of all children that have exited
egid	effective group id
egroup	group name (from egid)
emul	emulation name
etime	elapsed time since the process was started, in the form [[dd-]hh:]mm:ss
euid	effective user id
euser	user name (from euid)
flags	the process flags, in hexadecimal (alias *f*)
gid	effective group id
group	group name (from gid)
groupnames	group names (from group access list)
groups	group access list
holdcnt	number of holds on the process (if non-zero, process can't be swapped)
inblk	total blocks read (alias *inblock*)
jobc	job control count
ktrace	tracing flags
ktracep	tracing vnode
laddr	kernel virtual address of the struct lwp belonging to the LWP.
lid	ID of the LWP
lim	memory use limit
lname	descriptive name of the LWP
logname	login name of user who started the process (alias *login*)
lstart	time started
lstate	symbolic LWP state
majflt	total page faults
minflt	total page reclaims
msgrcv	total messages received (reads from pipes/sockets)
msgsnd	total messages sent (writes on pipes/sockets)
nice	nice value (alias *ni*)
nivcsw	total involuntary context switches
nlwp	number of LWPs in the process
nsigs	total signals taken (alias *nsignals*)
nswap	total swaps in/out
nvcsw	total voluntary context switches
nwchan	wait channel (as an address)
oublk	total blocks written (alias *oublock*)
p_ru	resource usage pointer (valid only for zombie)
paddr	kernel virtual address of the struct proc belonging to the process.
pagein	pageins (same as majflt)
pgid	process group number
pid	process ID
ppid	parent process ID
pri	scheduling priority
re	core residency time (in seconds; 127 = infinity)

rgid	real group ID
rlink	reverse link on run queue, or 0
rlwp	Number of LWPs on a processor or run queue
rss	resident set size
rsz	resident set size + (text size / text use count) (alias *rssize*)
ruid	real user ID
ruser	user name (from ruid)
sess	session pointer
sid	session ID
sig	pending signals (alias *pending*)
sigcatch	caught signals (alias *caught*)
sigignore	ignored signals (alias *ignored*)
sigmask	blocked signals (alias *blocked*)
sl	sleep time (in seconds; 127 = infinity)
start	time started
state	symbolic process state (alias *stat*)
stime	accumulated system CPU time
svgid	saved gid from a setgid executable
svgroup	group name (from svgid)
svuid	saved uid from a setuid executable
svuser	user name (from svuid)
tdev	control terminal device number
time	accumulated CPU time, user + system (alias *cputime*)
tpgid	control terminal process group ID
tsess	control terminal session pointer
tsiz	text size (in Kbytes)
tt	control terminal name (two letter abbreviation)
tty	full name of control terminal
uaddr	kernel virtual address of the struct user belonging to the LWP.
ucomm	name to be used for accounting
uid	effective user ID
upr	scheduling priority on return from system call (alias *usrpri*)
user	user name (from uid)
utime	accumulated user CPU time
vsz	virtual size in Kbytes (alias *vsize*)
wchan	wait channel (as a symbolic name)
xstat	exit or stop status (valid only for stopped or zombie process)

FILES

/dev	special files and device names
/dev/drum	default swap device
/var/run/dev.db	/dev name database
/var/db/kvm.db	system namelist database
/netbsd	default system namelist

SEE ALSO

kill(1), pgrep(1), pkill(1), sh(1), systrace(1), w(1), kvm(3), strftime(3), dev_mkdb(8), pstat(8)

BUGS

Since **ps** cannot run faster than the system and is run as any other scheduled process, the information it displays can never be exact.

NAME

pwd — return working directory name

SYNOPSIS

pwd [**-LP**]

DESCRIPTION

pwd writes the absolute pathname of the current working directory to the standard output.

The following options are available:

-L If the PWD environment variable is an absolute pathname that contains neither "/./" nor "/../" and references the current directory, then PWD is assumed to be the name of the current directory.

-P Print the physical path to the current working directory, with symbolic links in the path resolved.

The default for the **pwd** command is **-P**.

pwd is usually provided as a shell builtin (which may have a different default).

EXIT STATUS

The **pwd** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

cd(1), csh(1), ksh(1), sh(1), getcwd(3)

STANDARDS

The **pwd** utility is expected to be conforming to IEEE Std 1003.1 (“POSIX.1”), except that the default is **-P** not **-L**.

BUGS

In csh(1) the command **dirs** is always faster (although it can give a different answer in the rare case that the current directory or a containing directory was moved after the shell descended into it).

pwd -L relies on the file system having unique inode numbers. If this is not true (e.g., on FAT file systems) then **pwd -L** may fail to detect that PWD is incorrect.

NAME

pwhash — hashes passwords from the command line or standard input

SYNOPSIS

pwhash [**-km**] [**-b** *rounds*] [**-S** *rounds*] [**-s** *salt*] [**-p** | *string*]

DESCRIPTION

pwhash prints the encrypted form of *string* to the standard output. This is mostly useful for encrypting passwords from within scripts.

The options are as follows:

-b *rounds*
: Encrypt the string using Blowfish hashing with the specified *rounds*.

-k
: Run in makekey(8) compatible mode; a single combined key and salt are read from standard input and the DES encrypted result is written to standard output without a terminating newline.

-m
: Encrypt the string using MD5.

-p
: Prompt for a single string with echo turned off.

-S *rounds*
: Encrypt the salt with HMAC-SHA1 using the password as key and the specified *rounds* as a hint for the number of iterations.

-s *salt*
: Encrypt the string using DES, with the specified *salt*.

If no *string* is specified, **pwhash** reads one string per line from standard input, encrypting each one with the chosen algorithm from above. In the event that no specific algorithm is given as a command line option, the algorithm specified in the default class in /etc/passwd.conf will be used.

For MD5 and Blowfish a new random salt is automatically generated for each password.

Specifying the *string* on the command line should be discouraged; using the standard input is more secure.

FILES

/etc/passwd.conf

SEE ALSO

crypt(3), passwd.conf(5)

NAME

qmqp-sink − multi-threaded QMQP test server

SYNOPSIS

qmqp-sink [**-46cv**] [**-x** *time*] [**inet:**][*host*]:*port backlog*

qmqp-sink [**-46cv**] [**-x** *time*] **unix:***pathname backlog*

DESCRIPTION

qmqp-sink listens on the named host (or address) and port. It receives messages from the network and throws them away. The purpose is to measure QMQP client performance, not protocol compliance. Connections can be accepted on IPv4 or IPv6 endpoints, or on UNIX-domain sockets. IPv4 and IPv6 are the default. This program is the complement of the **qmqp-source**(1) program.

Note: this is an unsupported test program. No attempt is made to maintain compatibility between successive versions.

Arguments:

-4 Support IPv4 only. This option has no effect when Postfix is built without IPv6 support.

-6 Support IPv6 only. This option is not available when Postfix is built without IPv6 support.

-c Display a running counter that is updated whenever a delivery is completed.

-v Increase verbosity. Specify **-v -v** to see some of the QMQP conversation.

-x *time* Terminate after *time* seconds. This is to facilitate memory leak testing.

SEE ALSO

qmqp-source(1), QMQP message generator

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

qmqp-source − multi-threaded QMQP test generator

SYNOPSIS

qmqp-source [*options*] [**inet:**]*host*[:*port*]

qmqp-source [*options*] **unix:***pathname*

DESCRIPTION

qmqp-source connects to the named host and TCP port (default 628) and sends one or more messages to it, either sequentially or in parallel. The program speaks the QMQP protocol. Connections can be made to UNIX-domain and IPv4 or IPv6 servers. IPv4 and IPv6 are the default.

Note: this is an unsupported test program. No attempt is made to maintain compatibility between successive versions.

Arguments:

-4 Connect to the server with IPv4. This option has no effect when Postfix is built without IPv6 support.

-6 Connect to the server with IPv6. This option is not available when Postfix is built without IPv6 support.

-c Display a running counter that is incremented each time a delivery completes.

-C *count*
When a host sends RESET instead of SYN|ACK, try *count* times before giving up. The default count is 1. Specify a larger count in order to work around a problem with TCP/IP stacks that send RESET when the listen queue is full.

-f *from* Use the specified sender address (default: <foo@myhostname>).

-l *length*
Send *length* bytes as message payload. The length includes the message headers.

-m *message_count*
Send the specified number of messages (default: 1).

-M *myhostname*
Use the specified hostname or [address] in the default sender and recipient addresses, instead of the machine hostname.

-r *recipient_count*
Send the specified number of recipients per transaction (default: 1). Recipient names are generated by prepending a number to the recipient address.

-s *session_count*
Run the specified number of QMQP sessions in parallel (default: 1).

-t *to* Use the specified recipient address (default: <foo@myhostname>).

-R *interval*
Wait for a random period of time 0 <= n <= interval between messages. Suspending one thread does not affect other delivery threads.

-v Make the program more verbose, for debugging purposes.

-w *interval*
Wait a fixed time between messages. Suspending one thread does not affect other delivery threads.

SEE ALSO

qmqp-sink(1), QMQP message dump

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

qshape − Print Postfix queue domain and age distribution

SYNOPSIS

qshape [**-s**] [**-p**] [**-m** *min_subdomains*]
[**-b** *bucket_count*] [**-t** *bucket_time*]
[**-l**] [**-w** *terminal_width*]
[**-N** *batch_msg_count*] [**-n** *batch_top_domains*]
[**-c** *config_directory*] [*queue_name ...*]

DESCRIPTION

The **qshape** program helps the administrator understand the Postfix queue message distribution in time and by sender domain or recipient domain. The program needs read access to the queue directories and queue files, so it must run as the superuser or the **mail_owner** specified in **main.cf** (typically **postfix**).

Options:

-s Display the sender domain distribution instead of the recipient domain distribution. By default the recipient distribution is displayed. There can be more recipients than messages, but as each message has only one sender, the sender distribution is a message distribution.

-p Generate aggregate statistics for parent domains. Top level domains are not shown, nor are domains with fewer than *min_subdomains* subdomains. The names of parent domains are shown with a leading dot, (e.g. *.example.com*).

-m *min_subdomains*
When used with the **-p** option, sets the minimum subdomain count needed to show a separate line for a parent domain. The default is 5.

-b *bucket_count*
The age distribution is broken up into a sequence of geometrically increasing intervals. This option sets the number of intervals or "buckets". Each bucket has a maximum queue age that is twice as large as that of the previous bucket. The last bucket has no age limit.

-t *bucket_time*
The age limit in minutes for the first time bucket. The default value is 5, meaning that the first bucket counts messages between 0 and 5 minutes old.

-l Instead of using a geometric age sequence, use a linear age sequence, in other words simple multiples of **bucket_time**.

This feature is available in Postfix 2.2 and later.

-w *terminal_width*
The output is right justified, with the counts for the last bucket shown on the 80th column, the *terminal_width* can be adjusted for wider screens allowing more buckets to be displayed without truncating the domain names on the left. When a row for a full domain name and its counters does not fit in the specified number of columns, only the last 17 bytes of the domain name are shown with the prefix replaced by a ’+’ character. Truncated parent domain rows are shown as ’.+’ followed by the last 16 bytes of the domain name. If this is still too narrow to show the domain name and all the counters, the terminal_width limit is violated.

-N *batch_msg_count*
When the output device is a terminal, intermediate results are shown each "batch_msg_count" messages. This produces usable results in a reasonable time even when the deferred queue is large. The default is to show intermediate results every 1000 messages.

-n *batch_top_domains*
When reporting intermediate or final results to a termainal, report only the top "batch_top_domains" domains. The default limit is 20 domains.

-c *config_directory*

The **main.cf** configuration file is in the named directory instead of the default configuration directory.

Arguments:

queue_name

By default **qshape** displays the combined distribution of the incoming and active queues. To display a different set of queues, just list their directory names on the command line. Absolute paths are used as is, other paths are taken relative to the **main.cf queue_directory** parameter setting. While **main.cf** supports the use of *$variable* expansion in the definition of the **queue_directory** parameter, the **qshape** program does not. If you must use variable expansions in the **queue_directory** setting, you must specify an explicit absolute path for each queue subdirectory even if you want the default incoming and active queue distribution.

SEE ALSO

mailq(1), List all messages in the queue.
QSHAPE_README Examples and background material.

FILES

$config_directory/main.cf, Postfix installation parameters.
$queue_directory/maildrop/, local submission directory.
$queue_directory/incoming/, new message queue.
$queue_directory/hold/, messages waiting for tech support.
$queue_directory/active/, messages scheduled for delivery.
$queue_directory/deferred/, messages postponed for later delivery.

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Victor Duchovni
Morgan Stanley

NAME

qsieve, **qsafe** — generate system moduli file

SYNOPSIS

qsieve [*megabytes bits* [*initial*]]
qsafe [*trials* [*generator*]]

DESCRIPTION

The **qsieve** utility will list candidates for Sophie-Germaine primes (where q = (p-1)/2) to standard output. The list is checked against small known primes (less than 2**30). This step is both processor and memory intensive.

The *megabytes* value sets a limit for the internal sieve buffer. This should be small enough to remain entirely in memory. Swap thrashing can increase the run time from hours to days or weeks! When the *megabytes* value is zero (0), **qsieve** will select a default suitable for the *bits*.

The *bits* value sets the length of the generated possible primes (typically 768, 1024, 1536, 2048, 3072, or 4096, although others can be used for variety).

The optional *initial* value (hex) specifies the beginning of the search. Otherwise, **qsieve** generates a randomly selected number.

The **qsafe** utility will perform a Miller-Rabin primality test on the list of candidates (checking both q and p) from standard input. The result is a list of so-call "safe" primes to standard output, suitable for use as Diffie-Hellman moduli. This step is merely processor intensive.

The *trials* value sets the number of Miller-Rabin interations (typically 16 to 128).

The optional *generator* value (hex) limits testing to candidates with a specific generator (usually 2). Otherwise, **qsafe** will test each candidate and suggest a generator.

SEE ALSO

moduli(5)

HISTORY

These programs were originally developed for the Photuris project, and later the OpenSSH project.

NAME

qsubst — query-replace strings in files

SYNOPSIS

qsubst *str1* *str2* [*flags*] *file* [*file* [. . .]]

DESCRIPTION

qsubst reads its options (see below) to get a list of files. For each file on this list, it then replaces *str1* with *str2* wherever possible in that file, depending on user input (see below). The result is written back onto the original file.

For each potential substitution found, the user is prompted with a few lines before and after the line containing the string to be substituted. The string itself is displayed using the terminal's standout mode, if any. Then one character is read from the terminal. This is then interpreted as follows (this is designed to be like Emacs' query-replace-string):

space	Replace this occurrence and go on to the next one.
.	Replace this occurrence and don't change any more in this file (i.e., go on to the next file).
,	Tentatively replace this occurrence. The lines as they would look if the substitution were made are printed out. Then another character is read and it is used to decide the result as if the tentative replacement had not happened.
n	Don't change this one; just go on to the next one.
^G	Don't change this one or any others in this file, but instead simply go on to the next file.
!	Change the rest in this file without asking, then go on to the next file (at which point qsubst will start asking again).
?	Print out the current filename and ask again.

The first two arguments to qsubst are always the string to replace and the string to replace it with. The options are as follows:

-w	The search string is considered as a C symbol; it must be bounded by non-symbol characters. This option toggles. ('w' for 'word'.)
-! **-go**	
-noask	Enter ! mode automatically at the beginning of each file.
-nogo	
-ask	Negate **-go**, that is, ask as usual.
-c*N*	(Where *N* is a number.) Give *N* lines of context above and below the line with the match when prompting the user.
-CA*N*	(Where *N* is a number.) Give *N* lines of context above the line with the match when prompting the user.
-CB*N*	(Where *N* is a number.) Give *N* lines of context below the line with the match when prompting the user.
-f *filename*	The *filename* argument is one of the files qsubst should perform substitutions in.
-F *filename*	**qsubst** reads *filename* to get the names of files to perform substitutions in. The names should appear one to a line.

The default amount of context is **-c2**, that is, two lines above and two lines below the line with the match.

Arguments not beginning with a **-** sign in the options field are implicitly preceded by **-f**. Thus, **-f** is really needed only when the file name begins with a **-** sign.

qsubst reads its options in order and processes files as it gets them. This means, for example, that a **-go** will affect only files named after the **-go**.

The most context you can get is ten lines each, above and below.

str1 is limited to 512 characters; there is no limit on the size of *str2*. Neither one may contain a NUL.

NULs in the file may cause qsubst to make various mistakes.

If any other program modifies the file while qsubst is running, all bets are off.

AUTHORS

der Mouse ⟨mouse@rodents.montreal.qc.ca⟩

NAME

query-loc − to retrieve and display the location information in the DNS

SYNOPSIS

query-loc [-v]*[-d*nnn] *host*

DESCRIPTION

This manual page documents briefly the **query-loc** command.

query-loc is a program to retrieve and display the location information in the DNS.

It uses the algorithms described in RFC 1876 (and RFC 1101 to get the network names). You can find examples of networks wchich implement this scheme in the ADDRESSES file.

OPTIONS

−v Verbose mode.

−d nnn Debug mode. Displays the RFC's algorithm

BUGS

Very few hosts have location information.

AUTHOR

This manual page was written by Stephane Bortzmeyer <bortzmeyer@debian.org>.

NAME

quota — display disk usage and limits

SYNOPSIS

quota [**-g**] [**-u**] [**-v** | **-q**]
quota [**-u**] [**-v** | **-q**] *user*
quota [**-g**] [**-v** | **-q**] *group*

DESCRIPTION

quota displays users' disk usage and limits. By default only the user quotas are printed.

Options:

-g Print group quotas for the group of which the user is a member. The optional **-u** flag is equivalent to the default.

-v **quota** will display quotas on filesystems where no storage is allocated.

-q Print a more terse message, containing only information on filesystems where usage is over quota.

Specifying both **-g** and **-u** displays both the user quotas and the group quotas (for the user).

Only the super-user may use the **-u** flag and the optional *user* argument to view the limits of other users. Non-super-users can use the **-g** flag and optional *group* argument to view only the limits of groups of which they are members.

The **-q** flag takes precedence over the **-v** flag.

quota tries to report the quotas of all mounted filesystems. If the filesystem is mounted via *NFS* it will attempt to contact the `rpc.rquotad`(8) daemon on the *NFS* server. For *FFS* filesystems, quotas must be turned on in `/etc/fstab`. If **quota** exits with a non-zero status, one or more filesystems are over quota.

FILES

`quota.user` located at the filesystem root with user quotas
`quota.group` located at the filesystem root with group quotas
`/etc/fstab` to find filesystem names and locations

SEE ALSO

`quotactl`(2), `fstab`(5), `edquota`(8), `quotacheck`(8), `quotaon`(8), `repquota`(8), `rpc.rquotad`(8)

HISTORY

The **quota** command appeared in 4.2BSD.

NAME

radioctl — control radio tuners

SYNOPSIS

radioctl [**-f** *file*] [**-n**] **-a**
radioctl [**-f** *file*] [**-n**] *name*
radioctl [**-f** *file*] [**-n**] **-w** *name=value*

DESCRIPTION

The **radioctl** command displays or sets various variables that affect the radio tuner behavior. If a variable is present on the command line, **radioctl** prints the current value of this variable for the specified device. By default, **radioctl** operates on the `/dev/radio` device.

The options are as follows:

-a Print all device variables and their current values.

-w *name=value*
Attempt to set the specified variable *name* to *value*.

-f *file*
Specify an alternative radio tuner device.

-n Suppress printing of the variable name.

Values may be specified in either absolute or relative forms. The relative form is indicated by a prefix of '`+`' or '`-`' to denote an increase or decrease, respectively.

The exact set of controls that can be manipulated depends on the tuner. The general format (in both getting and setting a value) is

name = value

The *name* indicates what part of the tuner the control affects.

Write only controls:

search Only for cards that allow hardware search. Can be '`up`' or `down`.

Read-write controls:

frequency
Float value from 87.5 to 108.0.

volume Integer value from 0 to 255.

mute Mutes the card (volume is not affected), '`on`' or `off`.

mono Forces card output to mono, '`on`' or `off`. Only for cards that allow forced mono.

reference
Reference frequency. Can be 25 kHz, 50 kHz and 100 kHz. Not all cards allow to change the reference frequency.

sensitivity
Station locking sensitivity. Can be 5 mkV, 10 mkV, 30 mkV and 150 mkV. Not all cards allow to change the station locking sensitivity.

All the remaining controls (signal, stereo and card capabilities) are read-only and can be viewed using option **-a**.

ENVIRONMENT

The following environment variable affects the execution of **radioctl**:

`RADIODEVICE` The radio tuner device to use.

FILES

`/dev/radio` radio tuner device

EXAMPLES

The command

```
radioctl -a
```

can produce

```
volume=255
frequency=106.30MHz
mute=off
reference=50kHz
signal=on
stereo=on
card capabilities:
        manageable mono/stereo
```

SEE ALSO

`radio`(4)

HISTORY

radioctl command first appeared in OpenBSD 3.0 and NetBSD 1.6.

NAME

ranlib − generate index to archive.

SYNOPSIS

ranlib [**−vV**] *archive*

DESCRIPTION

ranlib generates an index to the contents of an archive and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file.

You may use **nm −s** or **nm −−print−armap** to list this index.

An archive with such an index speeds up linking to the library and allows routines in the library to call each other without regard to their placement in the archive.

The GNU **ranlib** program is another form of GNU **ar**; running **ranlib** is completely equivalent to executing **ar −s**.

OPTIONS

−v
−V
−−version
Show the version number of **ranlib**.

SEE ALSO

ar (1), *nm* (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

rcmd — backend driver for rcmd(3)

SYNOPSIS

rcmd [**-l** *username*] [**-u** *localusername*] *host command*

DESCRIPTION

rcmd executes *command* on *host*.

rcmd copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; **rcmd** normally terminates when the remote command does. The options are as follows:

-d The **-d** option turns on socket debugging (using setsockopt(2)) on the TCP sockets used for communication with the remote host.

-l By default, the remote username is the same as the local username. The **-l** option allows the remote name to be specified.

-u The **-u** option allows the local username to be specified. Only the superuser is allowed to use this option.

-n The **-n** option redirects input from the special device /dev/null (see the **BUGS** section of this manual page).

Shell metacharacters which are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine. For example, the command

```
rcmd otherhost cat remotefile >> localfile
```

appends the remote file *remotefile* to the local file *localfile*, while

```
rcmd otherhost cat remotefile ">>" other_remotefile
```

appends *remotefile* to *other_remotefile*.

FILES

/etc/hosts

SEE ALSO

rsh(1), rcmd(3), environ(7)

HISTORY

The **rcmd** command appeared in NetBSD 1.3 and is primarily derived from rsh(1). Its purpose was to create a backend driver for rcmd(3) that would allow the users of rcmd(3) to no longer require super-user privileges.

BUGS

If you are using csh(1) and put a **rcmd** in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. If no input is desired you should redirect the input of **rcmd** to /dev/null using the **-n** option.

You cannot use **rcmd** to run an interactive command (like rogue(6) or vi(1)). Use rlogin(1) instead.

The stop signal, SIGSTOP, will stop the local **rcmd** process only. This is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

NAME

rcp — remote file copy

SYNOPSIS

rcp [**-46p**] *file1 file2*
rcp [**-46pr**] *file ... directory*

DESCRIPTION

rcp copies files between machines. Each *file* or *directory* argument is either a remote file name of the form "rname@rhost:path", or a local file name (containing no ':' (colon) characters, or a '/' (slash) before any ':' (colon) characters).

The *rhost* can be an IPv4 or an IPv6 address string. Since IPv6 addresses already contain ':' (colon) characters, an IPv6 address string must be enclosed between '[' (left square bracket) and ']' (right square bracket) characters. Otherwise, the first occurrence of a ':' (colon) character would be interpreted as the separator between the *rhost* and the *path*. For example,

```
[2001:DB8::800:200C:417A]:tmp/file
```

Options:

-4 Use IPv4 addresses only.

-6 Use IPv6 addresses only.

-p The **-p** option causes **rcp** to attempt to preserve (duplicate) in its copies the modification times and modes of the source files, ignoring the *umask*. By default, the mode and owner of *file2* are preserved if it already existed; otherwise the mode of the source file modified by the umask(2) on the destination host is used.

-r If any of the source files are directories, **rcp** copies each subtree rooted at that name; in this case the destination must be a directory.

If *path* is not a full path name, it is interpreted relative to the login directory of the specified user *ruser* on *rhost*, or your current user name if no other remote user name is specified. A *path* on a remote host may be quoted (using \, ", or ´) so that the metacharacters are interpreted remotely.

rcp does not prompt for passwords; it performs remote execution via rsh(1), and requires the same authorization.

rcp handles third party copies, where neither source nor target files are on the current machine.

SEE ALSO

cp(1), ftp(1), rcmd(1), rlogin(1), rsh(1), rcmd(3), hosts.equiv(5), rhosts(5), environ(7)

HISTORY

The **rcp** utility appeared in 4.2BSD. The version of **rcp** described here has been reimplemented with Kerberos in 4.3BSD−Reno.

BUGS

Doesn't detect all cases where the target of a copy might be a file in cases where only a directory should be legal.

Is confused by any output generated by commands in a .login, .profile, or .cshrc file on the remote host.

The destination user and hostname may have to be specified as “rhost.rname” when the destination machine is running the 4.2BSD version of **rcp**.

NAME

rcp — copy file to and from remote machines

SYNOPSIS

rcp [**−45FKpxz**] [**−P** *port*] *file1 file2*
rcp [**−45FKprxz**] [**−P** *port*] *file... directory*

DESCRIPTION

rcp copies files between machines. Each file argument is either a remote file name of the form “rname@rhost:path” or a local file (containing no colon or with a slash before the first colon).

Supported options:

−4, **−5**, **−K**, **−F**, **−x**, **−z**
: These options are passed on to rsh(1).

−P *port*
: This will pass the option **−p** *port* to rsh(1).

−p
: Preserve file permissions.

−r
: Copy source directories recursively.

DIAGNOSTICS

rcp is implemented as a protocol on top of rsh(1), and thus requires a working rsh. If you intend to use Kerberos authentication, rsh needs to be Kerberos aware, else you may see more or less strange errors, such as "login incorrect", or "lost connection".

HISTORY

The **rcp** utility first appeared in 4.2BSD. This version is derived from 4.3BSD-Reno.

NAME

rcs − change RCS file attributes

SYNOPSIS

rcs *options file* ...

DESCRIPTION

rcs creates new RCS files or changes attributes of existing ones. An RCS file contains multiple revisions of text, an access list, a change log, descriptive text, and some control attributes. For **rcs** to work, the caller's login name must be on the access list, except if the access list is empty, the caller is the owner of the file or the superuser, or the **−i** option is present.

Pathnames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1). Revision numbers use the syntax described in **ci**(1).

OPTIONS

−i Create and initialize a new RCS file, but do not deposit any revision. If the RCS file has no path prefix, try to place it first into the subdirectory **./RCS**, and then into the current directory. If the RCS file already exists, print an error message.

−a*logins*
Append the login names appearing in the comma-separated list *logins* to the access list of the RCS file.

−A*oldfile*
Append the access list of *oldfile* to the access list of the RCS file.

−e[*logins*]
Erase the login names appearing in the comma-separated list *logins* from the access list of the RCS file. If *logins* is omitted, erase the entire access list.

−b[*rev*] Set the default branch to *rev*. If *rev* is omitted, the default branch is reset to the (dynamically) highest branch on the trunk.

−c*string*
Set the comment leader to *string*. An initial **ci**, or an **rcs −i** without **−c**, guesses the comment leader from the suffix of the working filename.

This option is obsolescent, since RCS normally uses the preceding **Log** line's prefix when inserting log lines during checkout (see **co**(1)). However, older versions of RCS use the comment leader instead of the **Log** line's prefix, so if you plan to access a file with both old and new versions of RCS, make sure its comment leader matches its **Log** line prefix.

−k*subst*
Set the default keyword substitution to *subst*. The effect of keyword substitution is described in **co**(1). Giving an explicit **−k** option to **co**, **rcsdiff**, and **rcsmerge** overrides this default. Beware **rcs −kv**, because **−kv** is incompatible with **co −l**. Use **rcs −kkv** to restore the normal default keyword substitution.

−l[*rev*] Lock the revision with number *rev*. If a branch is given, lock the latest revision on that branch. If *rev* is omitted, lock the latest revision on the default branch. Locking prevents overlapping changes. If someone else already holds the lock, the lock is broken as with **rcs −u** (see below).

−u[*rev*] Unlock the revision with number *rev*. If a branch is given, unlock the latest revision on that branch. If *rev* is omitted, remove the latest lock held by the caller. Normally, only the locker of a revision can unlock it. Somebody else unlocking a revision breaks the lock. This causes a mail message to be sent to the original locker. The message contains a commentary solicited from the breaker. The commentary is terminated by end-of-file or by a line containing **.** by itself.

−L Set locking to *strict*. Strict locking means that the owner of an RCS file is not exempt from locking for checkin. This option should be used for files that are shared.

−U Set locking to non-strict. Non-strict locking means that the owner of a file need not lock a revision for checkin. This option should *not* be used for files that are shared. Whether default locking is strict is determined by your system administrator, but it is normally strict.

−m*rev***:***msg*
Replace revision *rev*'s log message with *msg*.

−M Do not send mail when breaking somebody else's lock. This option is not meant for casual use; it is meant for programs that warn users by other means, and invoke **rcs −u** only as a low-level lock-breaking operation.

−n*name*[**:**[*rev*]]
Associate the symbolic name *name* with the branch or revision *rev*. Delete the symbolic name if both **:** and *rev* are omitted; otherwise, print an error message if *name* is already associated with another number. If *rev* is symbolic, it is expanded before association. A *rev* consisting of a branch number followed by a **.** stands for the current latest revision in the branch. A **:** with an empty *rev* stands for the current latest revision on the default branch, normally the trunk. For example, **rcs −n***name***: RCS/*** associates *name* with the current latest revision of all the named RCS files; this contrasts with **rcs −n***name***:$ RCS/*** which associates *name* with the revision numbers extracted from keyword strings in the corresponding working files.

−N*name*[**:**[*rev*]]
Act like **−n**, except override any previous assignment of *name*.

−o*range*
deletes ("outdates") the revisions given by *range*. A range consisting of a single revision number means that revision. A range consisting of a branch number means the latest revision on that branch. A range of the form *rev1***:***rev2* means revisions *rev1* to *rev2* on the same branch, **:***rev* means from the beginning of the branch containing *rev* up to and including *rev*, and *rev***:** means from revision *rev* to the end of the branch containing *rev*. None of the outdated revisions can have branches or locks.

−q Run quietly; do not print diagnostics.

−I Run interactively, even if the standard input is not a terminal.

−s*state*[**:***rev*]
Set the state attribute of the revision *rev* to *state*. If *rev* is a branch number, assume the latest revision on that branch. If *rev* is omitted, assume the latest revision on the default branch. Any identifier is acceptable for *state*. A useful set of states is **Exp** (for experimental), **Stab** (for stable), and **Rel** (for released). By default, **ci**(1) sets the state of a revision to **Exp**.

−t[*file*] Write descriptive text from the contents of the named *file* into the RCS file, deleting the existing text. The *file* pathname cannot begin with **−**. If *file* is omitted, obtain the text from standard input, terminated by end-of-file or by a line containing **.** by itself. Prompt for the text if interaction is possible; see **−I**. With **−i**, descriptive text is obtained even if **−t** is not given.

−t−*string*
Write descriptive text from the *string* into the RCS file, deleting the existing text.

−T Preserve the modification time on the RCS file unless a revision is removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when a change to the RCS file would mean a change to keyword strings in the working file.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* Use *zone* as the default time zone. This option has no effect; it is present for compatibility with other RCS commands.

At least one explicit option must be given, to ensure compatibility with future planned extensions to the **rcs** command.

COMPATIBILITY

The **−b***rev* option generates an RCS file that cannot be parsed by RCS version 3 or earlier.

The **−k***subst* options (except **−kkv**) generate an RCS file that cannot be parsed by RCS version 4 or earlier.

Use **rcs −V***n* to make an RCS file acceptable to RCS version *n* by discarding information that would confuse version *n*.

RCS version 5.5 and earlier does not support the **−x** option, and requires a **,v** suffix on an RCS pathname.

FILES

rcs accesses files much as **ci**(1) does, except that it uses the effective user for all accesses, it does not write the working file or its directory, and it does not even read the working file unless a revision number of **$** is specified.

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. See **ci**(1) for details.

DIAGNOSTICS

The RCS pathname and the revisions outdated are written to the diagnostic output. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

rcsintro(1), co(1), ci(1), ident(1), rcsclean(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

BUGS

A catastrophe (e.g. a system crash) can cause RCS to leave behind a semaphore file that causes later invocations of RCS to claim that the RCS file is in use. To fix this, remove the semaphore file. A semaphore file's name typically begins with **,** or ends with **_**.

The separator for revision ranges in the **−o** option used to be **−** instead of **:**, but this leads to confusion when symbolic names contain **−**. For backwards compatibility **rcs −o** still supports the old **−** separator, but it warns about this obsolete use.

Symbolic names need not refer to existing revisions or branches. For example, the **−o** option does not remove symbolic names for the outdated revisions; you must use **−n** to remove the names.

NAME

rcs2log — RCS to ChangeLog generator

SYNOPSIS

rcs2log [**-Rv**] [**-c** *changelog*] [**-h** *hostname*] [**-i** *indent*] [**-l** *length*] [**-r** *option*] [**-t** *tabwidth*] [**-u** *login<TAB>fullname<TAB>mailaddr*] [**--help**] [**--version**] [*file ...*]

DESCRIPTION

The **rcs2log** utility generates a change log prefix from RCS files (perhaps in the CVS repository) and the ChangeLog (if any).

The options are as follows:

-c *changelog*
: Output a change log prefix to *changelog* (default ChangeLog).

-h *hostname*
: Use *hostname* in change log entries (default current host).

-i *indent*
: Indent change log lines by *indent* spaces (default 8).

-l *length*
: Try to limit log lines to *length* characters (default 79).

-R
: If no *file*(s) are given and RCS is used, recurse through working directory.

-r *option*
: Pass *option* to subsidiary log command.

-t *tabwidth*
: Tab stops are every *tabwidth* characters (default 8).

-u *login<TAB>fullname<TAB>mailaddr*
: Assume *login* has *fullname* and *mailaddr*.

-v
: Append RCS revision to file names in log lines.

--help
: Output help.

--version
: Output version number.

SEE ALSO

rcs(1), rcsintro(1), rlog(1), rcsfile(5)

AUTHORS

Paul Eggert ⟨eggert@twinsun.com⟩

BUGS

Report bugs to ⟨bug-gnu-emacs@gnu.org⟩.

NAME

rcsclean − clean up working files

SYNOPSIS

rcsclean [*options*] [*file* . . .]

DESCRIPTION

rcsclean removes files that are not being worked on. **rcsclean −u** also unlocks and removes files that are being worked on but have not changed.

For each *file* given, **rcsclean** compares the working file and a revision in the corresponding RCS file. If it finds a difference, it does nothing. Otherwise, it first unlocks the revision if the **−u** option is given, and then removes the working file unless the working file is writable and the revision is locked. It logs its actions by outputting the corresponding **rcs −u** and **rm −f** commands on the standard output.

Files are paired as explained in **ci**(1). If no *file* is given, all working files in the current directory are cleaned. Pathnames matching an RCS suffix denote RCS files; all others denote working files.

The number of the revision to which the working file is compared may be attached to any of the options **−n**, **−q**, **−r**, or **−u**. If no revision number is specified, then if the **−u** option is given and the caller has one revision locked, **rcsclean** uses that revision; otherwise **rcsclean** uses the latest revision on the default branch, normally the root.

rcsclean is useful for **clean** targets in makefiles. See also **rcsdiff**(1), which prints out the differences, and **ci**(1), which normally reverts to the previous revision if a file was not changed.

OPTIONS

−k*subst*
: Use *subst* style keyword substitution when retrieving the revision for comparison. See **co**(1) for details.

−n[*rev*] Do not actually remove any files or unlock any revisions. Using this option will tell you what **rcsclean** would do without actually doing it.

−q[*rev*] Do not log the actions taken on standard output.

−r[*rev*] This option has no effect other than specifying the revision for comparison.

−T Preserve the modification time on the RCS file even if the RCS file changes because a lock is removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the lock removal would mean a change to keyword strings in the other working file.

−u[*rev*] Unlock the revision if it is locked and no difference is found.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*
: Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* Use *zone* as the time zone for keyword substitution; see **co**(1) for details.

EXAMPLES

rcsclean *.c *.h

removes all working files ending in **.c** or **.h** that were not changed since their checkout.

rcsclean

removes all working files in the current directory that were not changed since their checkout.

FILES

rcsclean accesses files much as **ci**(1) does.

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. A backslash escapes spaces within an option. The **RCSINIT** options are prepended to the argument lists of most RCS commands. Useful **RCSINIT** options include **−q**, **−V**, **−x**, and **−z**.

DIAGNOSTICS

The exit status is zero if and only if all operations were successful. Missing working files and RCS files are silently ignored.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rlog(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

BUGS

At least one *file* must be given in older Unix versions that do not provide the needed directory scanning operations.

NAME

rcsdiff − compare RCS revisions

SYNOPSIS

rcsdiff [**−k***subst*] [**−q**] [**−r***rev1* [**−r***rev2*]] [**−T**] [**−V**[*n*]] [**−x***suffixes*] [**−z***zone*] [*diff options*] *file* ...

DESCRIPTION

rcsdiff runs **diff**(1) to compare two revisions of each RCS file given.

Pathnames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

The option **−q** suppresses diagnostic output. Zero, one, or two revisions may be specified with **−r**. The option **−k***subst* affects keyword substitution when extracting revisions, as described in **co**(1); for example, **−kk −r1.1 −r1.2** ignores differences in keyword values when comparing revisions **1.1** and **1.2**. To avoid excess output from locker name substitution, **−kkvl** is assumed if (1) at most one revision option is given, (2) no **−k** option is given, (3) **−kkv** is the default keyword substitution, and (4) the working file's mode would be produced by **co −l**. See **co**(1) for details about **−T**, **−V**, **−x** and **−z**. Otherwise, all options of **diff**(1) that apply to regular files are accepted, with the same meaning as for **diff**.

If both *rev1* and *rev2* are omitted, **rcsdiff** compares the latest revision on the default branch (by default the trunk) with the contents of the corresponding working file. This is useful for determining what you changed since the last checkin.

If *rev1* is given, but *rev2* is omitted, **rcsdiff** compares revision *rev1* of the RCS file with the contents of the corresponding working file.

If both *rev1* and *rev2* are given, **rcsdiff** compares revisions *rev1* and *rev2* of the RCS file.

Both *rev1* and *rev2* may be given numerically or symbolically.

EXAMPLE

The command

rcsdiff f.c

compares the latest revision on the default branch of the RCS file to the contents of the working file **f.c**.

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. See **ci**(1) for details.

DIAGNOSTICS

Exit status is 0 for no differences during any comparison, 1 for some differences, 2 for trouble.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.5; Release Date: 1993/11/03.

SEE ALSO

ci(1), co(1), diff(1), ident(1), rcs(1), rcsintro(1), rcsmerge(1), rlog(1)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rcsfreeze − freeze a configuration of sources checked in under RCS

SYNOPSIS

rcsfreeze [*name*]

DESCRIPTION

rcsfreeze assigns a symbolic revision number to a set of RCS files that form a valid configuration.

The idea is to run **rcsfreeze** each time a new version is checked in. A unique symbolic name (**C_***number*, where *number* is increased each time **rcsfreeze** is run) is then assigned to the most recent revision of each RCS file of the main trunk.

An optional *name* argument to **rcsfreeze** gives a symbolic name to the configuration. The unique identifier is still generated and is listed in the log file but it will not appear as part of the symbolic revision name in the actual RCS files.

A log message is requested from the user for future reference.

The shell script works only on all RCS files at one time. All changed files must be checked in already. Run *rcsclean*(1) first and see whether any sources remain in the current directory.

FILES

RCS/.rcsfreeze.ver
: version number

RCS/.rcsfreeze.log
: log messages, most recent first

AUTHOR

Stephan v. Bechtolsheim

SEE ALSO

co(1), rcs(1), rcsclean(1), rlog(1)

BUGS

rcsfreeze does not check whether any sources are checked out and modified.

Although both source file names and RCS file names are accepted, they are not paired as usual with RCS commands.

Error checking is rudimentary.

rcsfreeze is just an optional example shell script, and should not be taken too seriously. See CVS for a more complete solution.

NAME

rcsintro − introduction to RCS commands

DESCRIPTION

The Revision Control System (RCS) manages multiple revisions of files. RCS automates the storing, retrieval, logging, identification, and merging of revisions. RCS is useful for text that is revised frequently, for example programs, documentation, graphics, papers, and form letters.

The basic user interface is extremely simple. The novice only needs to learn two commands: **ci**(1) and **co**(1). **ci**, short for "check in", deposits the contents of a file into an archival file called an RCS file. An RCS file contains all revisions of a particular file. **co**, short for "check out", retrieves revisions from an RCS file.

Functions of RCS

- Store and retrieve multiple revisions of text. RCS saves all old revisions in a space efficient way. Changes no longer destroy the original, because the previous revisions remain accessible. Revisions can be retrieved according to ranges of revision numbers, symbolic names, dates, authors, and states.
- Maintain a complete history of changes. RCS logs all changes automatically. Besides the text of each revision, RCS stores the author, the date and time of check-in, and a log message summarizing the change. The logging makes it easy to find out what happened to a module, without having to compare source listings or having to track down colleagues.
- Resolve access conflicts. When two or more programmers wish to modify the same revision, RCS alerts the programmers and prevents one modification from corrupting the other.
- Maintain a tree of revisions. RCS can maintain separate lines of development for each module. It stores a tree structure that represents the ancestral relationships among revisions.
- Merge revisions and resolve conflicts. Two separate lines of development of a module can be coalesced by merging. If the revisions to be merged affect the same sections of code, RCS alerts the user about the overlapping changes.
- Control releases and configurations. Revisions can be assigned symbolic names and marked as released, stable, experimental, etc. With these facilities, configurations of modules can be described simply and directly.
- Automatically identify each revision with name, revision number, creation time, author, etc. The identification is like a stamp that can be embedded at an appropriate place in the text of a revision. The identification makes it simple to determine which revisions of which modules make up a given configuration.
- Minimize secondary storage. RCS needs little extra space for the revisions (only the differences). If intermediate revisions are deleted, the corresponding deltas are compressed accordingly.

Getting Started with RCS

Suppose you have a file **f.c** that you wish to put under control of RCS. If you have not already done so, make an RCS directory with the command

mkdir RCS

Then invoke the check-in command

ci f.c

This command creates an RCS file in the **RCS** directory, stores **f.c** into it as revision 1.1, and deletes **f.c**. It also asks you for a description. The description should be a synopsis of the contents of the file. All later check-in commands will ask you for a log entry, which should summarize the changes that you made.

Files in the RCS directory are called RCS files; the others are called working files. To get back the working file **f.c** in the previous example, use the check-out command

co f.c

This command extracts the latest revision from the RCS file and writes it into **f.c**. If you want to edit **f.c**, you must lock it as you check it out with the command

co −l f.c

You can now edit **f.c**.

Suppose after some editing you want to know what changes that you have made. The command

rcsdiff f.c

tells you the difference between the most recently checked-in version and the working file. You can check the file back in by invoking

ci f.c

This increments the revision number properly.

If **ci** complains with the message

ci error: no lock set by *your name*

then you have tried to check in a file even though you did not lock it when you checked it out. Of course, it is too late now to do the check-out with locking, because another check-out would overwrite your modifications. Instead, invoke

rcs −l f.c

This command will lock the latest revision for you, unless somebody else got ahead of you already. In this case, you'll have to negotiate with that person.

Locking assures that you, and only you, can check in the next update, and avoids nasty problems if several people work on the same file. Even if a revision is locked, it can still be checked out for reading, compiling, etc. All that locking prevents is a *check-in* by anybody but the locker.

If your RCS file is private, i.e., if you are the only person who is going to deposit revisions into it, strict locking is not needed and you can turn it off. If strict locking is turned off, the owner of the RCS file need not have a lock for check-in; all others still do. Turning strict locking off and on is done with the commands

rcs −U f.c and **rcs −L f.c**

If you don't want to clutter your working directory with RCS files, create a subdirectory called **RCS** in your working directory, and move all your RCS files there. RCS commands will look first into that directory to find needed files. All the commands discussed above will still work, without any modification. (Actually, pairs of RCS and working files can be specified in three ways: (a) both are given, (b) only the working file is given, (c) only the RCS file is given. Both RCS and working files may have arbitrary path prefixes; RCS commands pair them up intelligently.)

To avoid the deletion of the working file during check-in (in case you want to continue editing or compiling), invoke

ci −l f.c or **ci −u f.c**

These commands check in **f.c** as usual, but perform an implicit check-out. The first form also locks the checked in revision, the second one doesn't. Thus, these options save you one check-out operation. The first form is useful if you want to continue editing, the second one if you just want to read the file. Both update the identification markers in your working file (see below).

You can give **ci** the number you want assigned to a checked in revision. Assume all your revisions were numbered 1.1, 1.2, 1.3, etc., and you would like to start release 2. The command

ci −r2 f.c or **ci −r2.1 f.c**

assigns the number 2.1 to the new revision. From then on, **ci** will number the subsequent revisions with 2.2, 2.3, etc. The corresponding **co** commands

co −r2 f.c and **co −r2.1 f.c**

retrieve the latest revision numbered 2.*x* and the revision 2.1, respectively. **co** without a revision number selects the latest revision on the *trunk*, i.e. the highest revision with a number consisting of two fields. Numbers with more than two fields are needed for branches. For example, to start a branch at revision 1.3, invoke

ci −r1.3.1 f.c

This command starts a branch numbered 1 at revision 1.3, and assigns the number 1.3.1.1 to the new revision. For more information about branches, see **rcsfile**(5).

Automatic Identification

RCS can put special strings for identification into your source and object code. To obtain such identification, place the marker

Id

into your text, for instance inside a comment. RCS will replace this marker with a string of the form

$Id: *filename revision date time author state* **$**

With such a marker on the first page of each module, you can always see with which revision you are working. RCS keeps the markers up to date automatically. To propagate the markers into your object code, simply put them into literal character strings. In C, this is done as follows:

```
static char rcsid[] = "$Id$";
```

The command **ident** extracts such markers from any file, even object code and dumps. Thus, **ident** lets you find out which revisions of which modules were used in a given program.

You may also find it useful to put the marker **Log** into your text, inside a comment. This marker accumulates the log messages that are requested during check-in. Thus, you can maintain the complete history of your file directly inside it. There are several additional identification markers; see **co**(1) for details.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rlog(1)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rcsmerge − merge RCS revisions

SYNOPSIS

rcsmerge [*options*] *file*

DESCRIPTION

rcsmerge incorporates the changes between two revisions of an RCS file into the corresponding working file.

Pathnames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

At least one revision must be specified with one of the options described below, usually **−r**. At most two revisions may be specified. If only one revision is specified, the latest revision on the default branch (normally the highest branch on the trunk) is assumed for the second revision. Revisions may be specified numerically or symbolically.

rcsmerge prints a warning if there are overlaps, and delimits the overlapping regions as explained in **merge**(1). The command is useful for incorporating changes into a checked-out revision.

OPTIONS

−A Output conflicts using the **−A** style of **diff3**(1), if supported by **diff3**. This merges all changes leading from *file2* to *file3* into *file1*, and generates the most verbose output.

−E, **−e** These options specify conflict styles that generate less information than **−A**. See **diff3**(1) for details. The default is **−E**. With **−e**, **rcsmerge** does not warn about conflicts.

−k*subst*
Use *subst* style keyword substitution. See **co**(1) for details. For example, **−kk −r1.1 −r1.2** ignores differences in keyword values when merging the changes from **1.1** to **1.2**. It normally does not make sense to merge binary files as if they were text, so **rcsmerge** refuses to merge files if **−kb** expansion is used.

−p[*rev*] Send the result to standard output instead of overwriting the working file.

−q[*rev*] Run quietly; do not print diagnostics.

−r[*rev*] Merge with respect to revision *rev*. Here an empty *rev* stands for the latest revision on the default branch, normally the head.

−T This option has no effect; it is present for compatibility with other RCS commands.

−V Print RCS's version number.

−V*n* Emulate RCS version *n*. See **co**(1) for details.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* Use *zone* as the time zone for keyword substitution. See **co**(1) for details.

EXAMPLES

Suppose you have released revision 2.8 of **f.c**. Assume furthermore that after you complete an unreleased revision 3.4, you receive updates to release 2.8 from someone else. To combine the updates to 2.8 and your changes between 2.8 and 3.4, put the updates to 2.8 into file f.c and execute

rcsmerge −p −r2.8 −r3.4 f.c >f.merged.c

Then examine **f.merged.c**. Alternatively, if you want to save the updates to 2.8 in the RCS file, check them in as revision 2.8.1.1 and execute **co −j**:

ci −r2.8.1.1 f.c
co −r3.4 −j2.8:2.8.1.1 f.c

As another example, the following command undoes the changes between revision 2.4 and 2.8 in your currently checked out revision in **f.c**.

rcsmerge −r2.8 −r2.4 f.c

Note the order of the arguments, and that **f.c** will be overwritten.

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. See **ci**(1) for details.

DIAGNOSTICS

Exit status is 0 for no overlaps, 1 for some overlaps, 2 for trouble.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

ci(1), co(1), ident(1), merge(1), rcs(1), rcsdiff(1), rcsintro(1), rlog(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

NAME

rdist — remote file distribution program

SYNOPSIS

rdist [**-bDhinqRvwy**] [**-d** *var=value*] [**-f** *distfile*] [**-m** *host*] [*name ...*]
rdist [**-bDhinqRvwy**] **-c** *name ...* [login@]*host*[:dest]

DESCRIPTION

rdist is a program to maintain identical copies of files over multiple hosts. It preserves the owner, group, mode, and mtime of files if possible and can update programs that are executing. **rdist** reads commands from *distfile* to direct the updating of files and/or directories.

Options specific to the first SYNOPSIS form:

- If *distfile* is '**-**', the standard input is used.

-f *distfile*
Use the specified *distfile*.

If either the **-f** or '**-**' option is not specified, the program looks first for "distfile", then "Distfile" to use as the input. If no names are specified on the command line, **rdist** will update all of the files and directories listed in *distfile*. Otherwise, the argument is taken to be the name of a file to be updated or the label of a command to execute. If label and file names conflict, it is assumed to be a label. These may be used together to update specific files using specific commands.

Options specific to the second SYNOPSIS form:

-c Forces **rdist** to interpret the remaining arguments as a small *distfile*.

The equivalent distfile is as follows.

```
(name ...) -> [login@]host
        install [dest];
```

Options common to both forms:

-b Binary comparison. Perform a binary comparison and update files if they differ rather than comparing dates and sizes.

-d *var=value*
Define *var* to have *value*. The **-d** option is used to define or override variable definitions in the *distfile*. *Value* can be the empty string, one name, or a list of names surrounded by parentheses and separated by tabs and/or spaces.

-D Turn on debugging.

-h Follow symbolic links. Copy the file that the link points to rather than the link itself.

-i Ignore unresolved links. **rdist** will normally try to maintain the link structure of files being transferred and warn the user if all the links cannot be found.

-m *host* Limit which machines are to be updated. Multiple **-m** arguments can be given to limit updates to a subset of the hosts listed in the *distfile*.

-n Print the commands without executing them. This option is useful for debugging *distfile*.

-q Quiet mode. Files that are being modified are normally printed on standard output. The **-q** option suppresses this.

-R Remove extraneous files. If a directory is being updated, any files that exist on the remote host that do not exist in the master directory are removed. This is useful for maintaining truly identical copies of directories.

-v Verify that the files are up to date on all the hosts. Any files that are out of date will be displayed but no files will be changed nor any mail sent.

-w Whole mode. The whole file name is appended to the destination directory name. Normally, only the last component of a name is used when renaming files. This will preserve the directory structure of the files being copied instead of flattening the directory structure. For example, renaming a list of files such as (dir1/f1 dir2/f2) to dir3 would create files dir3/dir1/f1 and dir3/dir2/f2 instead of dir3/f1 and dir3/f2.

-y Younger mode. Files are normally updated if their *mtime* and *size* (see stat(2)) disagree. The **-y** option causes **rdist** not to update files that are younger than the master copy. This can be used to prevent newer copies on other hosts from being replaced. A warning message is printed for files which are newer than the master copy.

Distfile contains a sequence of entries that specify the files to be copied, the destination hosts, and what operations to perform to do the updating. Each entry has one of the following formats.

```
<variable name> '=' <name list>
[label:]<source list> '->' <destination list> <command list>
[label:]<source list> '::' <time_stamp file> <command list>
```

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for making lists of files that have been changed since some given date. The *source list* specifies a list of files and/or directories on the local host which are to be used as the master copy for distribution. The *destination list* is the list of hosts to which these files are to be copied. Each file in the source list is added to a list of changes if the file is out of date on the host which is being updated (second format) or the file is newer than the time stamp file (third format).

Labels are optional. They are used to identify a command for partial updates.

Newlines, tabs, and blanks are only used as separators and are otherwise ignored. Comments begin with '#' and end with a newline.

Variables to be expanded begin with '$' followed by one character or a name enclosed in curly braces (see the examples at the end).

The source and destination lists have the following format:

```
<name>
```

or

```
'(' <zero or more names separated by white-space> ')'
```

The shell meta-characters '[', ']', '{', '}', '*', and '?' are recognized and expanded (on the local host only) in the same way as csh(1). They can be escaped with a backslash. The '~' character is also expanded in the same way as csh(1) but is expanded separately on the local and destination hosts. When the **-w** option is used with a file name that begins with '~', everything except the home directory is appended to the destination name. File names which do not begin with '/' or '~' use the destination user's home directory as the root directory for the rest of the file name.

The command list consists of zero or more commands of the following format.

'install' <options> opt_dest_name ';'

‘notify’	<name list>	‘;’
‘except’	<name list>	‘;’
‘except_pat’	<pattern list>	‘;’
‘special’	<name list>	string ‘;’

The **install** command is used to copy out of date files and/or directories. Each source file is copied to each host in the destination list. Directories are recursively copied in the same way. *Opt_dest_name* is an optional parameter to rename files. If no **install** command appears in the command list or the destination name is not specified, the source file name is used. Directories in the path name will be created if they do not exist on the remote host. To help prevent disasters, a non-empty directory on a target host will never be replaced with a regular file or a symbolic link. However, under the ‘−R’ option a non-empty directory will be removed if the corresponding filename is completely absent on the master host. The *options* are ‘−R’, ‘−h’, ‘−i’, ‘−v’, ‘−w’, ‘−y’, and ‘−b’ and have the same semantics as options on the command line except they only apply to the files in the source list. The login name used on the destination host is the same as the local host unless the destination name is of the format ‘‘login@host".

The **notify** command is used to mail the list of files updated (and any errors that may have occurred) to the listed names. If no ‘@’ appears in the name, the destination host is appended to the name (e.g., name1@host, name2@host, ...).

The **except** command is used to update all of the files in the source list **except** for the files listed in *name list*. This is usually used to copy everything in a directory except certain files.

The **except_pat** command is like the **except** command except that *pattern list* is a list of regular expressions (see ed(1) for details). If one of the patterns matches some string within a file name, that file will be ignored. Note that since ‘\’ is a quote character, it must be doubled to become part of the regular expression. Variables are expanded in *pattern list* but not shell file pattern matching characters. To include a ‘$’, it must be escaped with ‘\’.

The **special** command is used to specify sh(1) commands that are to be executed on the remote host after the file in *name list* is updated or installed. If the *name list* is omitted then the shell commands will be executed for every file updated or installed. The shell variable ‘FILE’ is set to the current filename before executing the commands in *string*. *String* starts and ends with ‘"’ and can cross multiple lines in *distfile*. Multiple commands to the shell should be separated by ‘;’. Commands are executed in the user’s home directory on the host being updated. The *special* command can be used to rebuild private databases, etc. after a program has been updated.

The following is a small example:

```
HOSTS = ( matisse root@arpa )

FILES = ( /bin /lib /usr/bin /usr/games
/usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
/usr/lib /usr/man/man? /usr/ucb /usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )

${FILES} -> ${HOSTS}
install -R ;
except /usr/lib/${EXLIB} ;
except /usr/games/lib ;
special /usr/lib/sendmail "/usr/lib/sendmail -bz" ;

srcs:
```

```
/usr/src/bin -> arpa
except_pat ( \\.o\$ /SCCS\$ ) ;

IMAGEN = (ips dviimp catdvi)

imagen:
/usr/local/${IMAGEN} -> arpa
install /usr/local/lib ;
notify ralph ;

${FILES} :: stamp.cory
notify root@cory ;
```

FILES

`distfile`	input command file
`/tmp/rdist*`	temporary file for update lists

DIAGNOSTICS

A complaint about mismatch of rdist version numbers may really stem from some problem with starting your shell, e.g., you are in too many groups.

SEE ALSO

`csh`(1), `sh`(1), `stat`(2)

HISTORY

The **`rdist`** command appeared in 4.3BSD.

BUGS

Source files must reside on the local host where **`rdist`** is executed.

There is no easy way to have a special command executed after all files in a directory have been updated.

Variable expansion only works for name lists; there should be a general macro facility.

`rdist` aborts on files which have a negative mtime (before Jan 1, 1970).

There should be a ‘force’ option to allow replacement of non-empty directories by regular files or symlinks. A means of updating file modes and owners of otherwise identical files is also needed.

NAME

readelf − Displays information about ELF files.

SYNOPSIS

readelf [**−a**|**−−all**]
[**−h**|**−−file−header**]
[**−l**|**−−program−headers**|**−−segments**]
[**−S**|**−−section−headers**|**−−sections**]
[**−g**|**−−section−groups**]
[**−e**|**−−headers**]
[**−s**|**−−syms**|**−−symbols**]
[**−n**|**−−notes**]
[**−r**|**−−relocs**]
[**−u**|**−−unwind**]
[**−d**|**−−dynamic**]
[**−V**|**−−version−info**]
[**−A**|**−−arch−specific**]
[**−D**|**−−use−dynamic**]
[**−x** <number>|**−−hex−dump=**<number>]
[**−w[liaprmfFsoR]**|
−−debug−dump[=line,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames−interp,=str,=loc,=Ranges]]
[**−I**|**−histogram**]
[**−v**|**−−version**]
[**−W**|**−−wide**]
[**−H**|**−−help**]
elffile...

DESCRIPTION

readelf displays information about one or more ELF format object files. The options control what particular information to display.

elffile... are the object files to be examined. 32−bit and 64−bit ELF files are supported, as are archives containing ELF files.

This program performs a similar function to **objdump** but it goes into more detail and it exists independently of the BFD library, so if there is a bug in BFD then readelf will not be affected.

OPTIONS

The long and short forms of options, shown here as alternatives, are equivalent. At least one option besides **−v** or **−H** must be given.

−a
−−all
Equivalent to specifiying **−−file−header**, **−−program−headers**, **−−sections**, **−−symbols**, **−−relocs**, **−−dynamic**, **−−notes** and **−−version−info**.

−h
−−file−header
Displays the information contained in the ELF header at the start of the file.

−l
−−program−headers
−−segments
Displays the information contained in the file's segment headers, if it has any.

−S
−−sections

−−section−headers
Displays the information contained in the file's section headers, if it has any.

−g
−−section−groups
Displays the information contained in the file's section groups, if it has any.

−s
−−symbols
−−syms
Displays the entries in symbol table section of the file, if it has one.

−e
−−headers
Display all the headers in the file. Equivalent to **−h −l −S**.

−n
−−notes
Displays the contents of the NOTE segments and/or sections, if any.

−r
−−relocs
Displays the contents of the file's relocation section, if it has one.

−u
−−unwind
Displays the contents of the file's unwind section, if it has one. Only the unwind sections for IA64 ELF files are currently supported.

−d
−−dynamic
Displays the contents of the file's dynamic section, if it has one.

−V
−−version−info
Displays the contents of the version sections in the file, it they exist.

−A
−−arch−specific
Displays architecture-specific information in the file, if there is any.

−D
−−use−dynamic
When displaying symbols, this option makes **readelf** use the symbol table in the file's dynamic section, rather than the one in the symbols section.

−x <number>
−−hex−dump=<number>
Displays the contents of the indicated section as a hexadecimal dump.

−w[liaprmfFsoR]
−−debug−dump[=line,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames−interp,=str,=loc,=Ranges]
Displays the contents of the debug sections in the file, if any are present. If one of the optional letters or words follows the switch then only data found in those specific sections will be dumped.

−I
−−histogram
Display a histogram of bucket list lengths when displaying the contents of the symbol tables.

−v

−−version
Display the version number of readelf.

−W
−−wide
Don’t break output lines to fit into 80 columns. By default **readelf** breaks section header and segment listing lines for 64−bit ELF files, so that they fit into 80 columns. This option causes **readelf** to print each section header resp. each segment one a single line, which is far more readable on terminals wider than 80 columns.

−H
−−help
Display the command line options understood by **readelf**.

SEE ALSO

objdump (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

rev — reverse order of characters of lines in a file

SYNOPSIS

rev [*file*]

DESCRIPTION

The **rev** utility copies the specified files to the standard output, reversing the order of characters in every line. If no files are specified, the standard input is read.

NAME

rfcomm_sppd — RFCOMM Serial Port Profile daemon

SYNOPSIS

rfcomm_sppd [**-d** *device*] [**-m** *mode*] [**-s** *service*] [**-t** *tty*]
{ **-a** *address* | **-c** *channel*}

DESCRIPTION

The **rfcomm_sppd** utility is a Serial Port Profile daemon, providing serial access over an RFCOMM connection to a remote device. **rfcomm_sppd** can work in client or server mode.

In client mode, **rfcomm_sppd** initiates an RFCOMM connection to the *service* at the remote *address*.

In server mode, **rfcomm_sppd** registers the *service* with the local SDP server and listens on the specified RFCOMM *channel* for an incoming connection.

The options are as follows:

-a *address* Client mode. Specify the address of the remote RFCOMM device. The address can be specified as BD_ADDR or name. If given as a name, then the **rfcomm_sppd** utility will attempt to resolve the name via bt_gethostbyname(3).

-c *channel* Server mode. Specify the RFCOMM channel number to listen on. **rfcomm_sppd** will register the service with the local sdpd(8) daemon. Note that registering services with sdpd(8) is a privileged operation.

-d *device* Use the local device with the specified address. The device can be specified by BD_ADDR or device name. See btconfig(8) for a list of available devices. If no *device* is specified, the connection will be set up on a system determined device.

-m *mode* Set connection link mode. Supported modes are:

auth	require devices be paired.
encrypt	auth, plus enable encryption.
secure	encryption, plus change of link key.

-s *service* This is the service class that will be searched for on the remote device. If no *service* is given, the default "Serial Port" service class will be used. Known service classes are:

DUN	Dialup Networking
LAN	LAN access using PPP
SP	Serial Port

In client mode, the service class may be given as a channel number, for instances where the remote device does not provide Service Discovery.

-t *tty* Slave pseudo tty name. If this option is given, **rfcomm_sppd** will detach from the controlling process after the Bluetooth connection is made, and operate over the named pty(4) pair. Otherwise, stdin/stdout will be used.

EXIT STATUS

The **rfcomm_sppd** utility exits 0 on success, and >0 if an error occurs.

FILES

`/dev/pty[p-sP-S][0-9a-v]` master pseudo terminals
`/dev/tty[p-sP-S][0-9a-v]` slave pseudo terminals

EXAMPLES

```
rfcomm_sppd -a 00:01:02:03:04:05 -s 1 -t /dev/ttyp1
```

Will open an RFCOMM connection to the server at `00:01:02:03:04:05` on channel `1`. Once the connection has been established, **`rfcomm_sppd`** will detach and `/dev/ttyp1` can be used to communicate with the remote serial port on the server, e.g. with the use of

```
cu -l /dev/ttyp1
```

In order to use **`rfcomm_sppd`** to automatically create a secured link for `pppd`(8), use

```
pty "rfcomm_sppd -a 00:01:02:03:04:05 -s DUN -m secure"
```

in your `pppd`(8) configuration file.

SEE ALSO

`bluetooth`(3), `bluetooth`(4), `pty`(4), `btconfig`(8), `pppd`(8), `sdpd`(8)

HISTORY

The **`rfcomm_sppd`** program first appeared in FreeBSD and was ported to NetBSD 4.0 by Iain Hibbert under the sponsorship of Itronix, Inc.

AUTHORS

Maksim Yevmenkin ⟨m_evmenkin@yahoo.com⟩,
Iain Hibbert

BUGS

Please report if found.

NAME

rlog − print log messages and other information about RCS files

SYNOPSIS

rlog [*options*] *file* . . .

DESCRIPTION

rlog prints information about RCS files.

Pathnames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

rlog prints the following information for each RCS file: RCS pathname, working pathname, head (i.e., the number of the latest revision on the trunk), default branch, access list, locks, symbolic names, suffix, total number of revisions, number of revisions selected for printing, and descriptive text. This is followed by entries for the selected revisions in reverse chronological order for each branch. For each revision, **rlog** prints revision number, author, date/time, state, number of lines added/deleted (with respect to the previous revision), locker of the revision (if any), and log message. All times are displayed in Coordinated Universal Time (UTC) by default; this can be overridden with **−z**. Without options, **rlog** prints complete information. The options below restrict this output.

−L Ignore RCS files that have no locks set. This is convenient in combination with **−h**, **−l**, and **−R**.

−R Print only the name of the RCS file. This is convenient for translating a working pathname into an RCS pathname.

−h Print only the RCS pathname, working pathname, head, default branch, access list, locks, symbolic names, and suffix.

−t Print the same as **−h**, plus the descriptive text.

−N Do not print the symbolic names.

−b Print information about the revisions on the default branch, normally the highest branch on the trunk.

−d*dates*
Print information about revisions with a checkin date/time in the ranges given by the semicolon-separated list of *dates*. A range of the form *d1*<*d2* or *d2*>*d1* selects the revisions that were deposited between *d1* and *d2* exclusive. A range of the form <*d* or *d*> selects all revisions earlier than *d*. A range of the form *d*< or >*d* selects all revisions dated later than *d*. If < or > is followed by = then the ranges are inclusive, not exclusive. A range of the form *d* selects the single, latest revision dated *d* or earlier. The date/time strings *d*, *d1*, and *d2* are in the free format explained in **co**(1). Quoting is normally necessary, especially for < and >. Note that the separator is a semicolon.

−l[*lockers*]
Print information about locked revisions only. In addition, if the comma-separated list *lockers* of login names is given, ignore all locks other than those held by the *lockers*. For example, **rlog −L −R −lwft RCS/*** prints the name of RCS files locked by the user **wft**.

−r[*revisions*]
prints information about revisions given in the comma-separated list *revisions* of revisions and ranges. A range *rev1***:***rev2* means revisions *rev1* to *rev2* on the same branch, **:***rev* means revisions from the beginning of the branch up to and including *rev*, and *rev***:** means revisions starting with *rev* to the end of the branch containing *rev*. An argument that is a branch means all revisions on that branch. A range of branches means all revisions on the branches in that range. A branch followed by a **.** means the latest revision in that branch. A bare **−r** with no *revisions* means the latest revision on the default branch, normally the trunk.

−s*states*
prints information about revisions whose state attributes match one of the states given in the comma-separated list *states*.

−w[*logins*]
prints information about revisions checked in by users with login names appearing in the comma-separated list *logins*. If *logins* is omitted, the user's login is assumed.

−T This option has no effect; it is present for compatibility with other RCS commands.

−V Print RCS's version number.

−V*n*
Emulate RCS version *n* when generating logs. See **co**(1) for more.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

rlog prints the intersection of the revisions selected with the options **−d**, **−l**, **−s**, and **−w**, intersected with the union of the revisions selected by **−b** and **−r**.

−z*zone* specifies the date output format, and specifies the default time zone for *date* in the **−d***dates* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

EXAMPLES

```
rlog −L −R RCS/*
rlog −L −h RCS/*
rlog −L −l RCS/*
rlog RCS/*
```

The first command prints the names of all RCS files in the subdirectory **RCS** that have locks. The second command prints the headers of those files, and the third prints the headers plus the log messages of the locked revisions. The last command prints complete information.

ENVIRONMENT

RCSINIT
options prepended to the argument list, separated by spaces. See **ci**(1) for details.

DIAGNOSTICS

The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: ; Release Date: .

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

BUGS

The separator for revision ranges in the **−r** option used to be − instead of **:**, but this leads to confusion when symbolic names contain −. For backwards compatibility **rlog −r** still supports the old − separator, but it warns about this obsolete use.

NAME

rlogin — remote login

SYNOPSIS

rlogin [**-468Ed**] [**-e** *char*] [**-l** *username*] [**-p** *port*] *host*
rlogin [**-468Ed**] [**-e** *char*] [**-p** *port*] *username@host*

DESCRIPTION

rlogin starts a terminal session on a remote host *host*.

rlogin first attempts to use the standard Berkeley *rhosts* authorization mechanism. The options are as follows:

-4 Use IPv4 addresses only.

-6 Use IPv6 addresses only.

-8 The **-8** option allows an eight-bit input data path at all times; otherwise parity bits are stripped except when the remote side's stop and start characters are other than '^S/^Q'.

-E The **-E** option stops any character from being recognized as an escape character. When used with the **-8** option, this provides a completely transparent connection.

-d The **-d** option turns on socket debugging (see `setsockopt`(2)) on the TCP sockets used for communication with the remote host.

-e *char* The **-e** option allows user specification of the escape character, which is "~" by default. This specification may be as a literal character, or as an octal value in the form \nnn.

-l *username* the **-l** option specifies an alternate *username* for the remote login. If this option is not specified, your local username will be used.

-p *port* Uses the given *port* instead of the one assigned to the service "login". May be given either as symbolic name or as number.

A line of the form "⟨escape char⟩." disconnects from the remote host. Similarly, the line "⟨escape char⟩^Z" will suspend the **rlogin** session, and "⟨escape char⟩⟨delayed-suspend char⟩" suspends the send portion of the rlogin, but allows output from the remote system. By default, the tilde ("~") character is the escape character, and normally control-Y ("^Y") is the delayed-suspend character.

All echoing takes place at the remote site, so that (except for delays) the **rlogin** is transparent. Flow control via ^S/^Q and flushing of input and output on interrupts are handled properly.

ENVIRONMENT

The following environment variable is used by **rlogin**:

`TERM` Determines the user's terminal type.

SEE ALSO

`rcmd`(1), `rsh`(1), `rcmd`(3), `hosts.equiv`(5), `rhosts`(5), `environ`(7)

HISTORY

The **rlogin** command appeared in 4.2BSD.

BUGS

More of the environment should be propagated.

NAME

rm — remove directory entries

SYNOPSIS

rm [**-f** | **-i**] [**-dPRrvW**] *file ...*

DESCRIPTION

The **rm** utility attempts to remove the non-directory type files specified on the command line. If the permissions of the file do not permit writing, and the standard input device is a terminal, the user is prompted (on the standard error output) for confirmation.

The options are as follows:

-d Attempt to remove directories as well as other types of files.

-f Attempt to remove the files without prompting for confirmation, regardless of the file's permissions. If the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error. The **-f** option overrides any previous **-i** options.

-i Request confirmation before attempting to remove each file, regardless of the file's permissions, or whether or not the standard input device is a terminal. The **-i** option overrides any previous **-f** options.

-P Overwrite regular files before deleting them. Files are overwritten three times, first with the byte pattern 0xff, then 0x00, and then with random data, before they are deleted. Some care is taken to ensure that the data are actually written to disk, but this cannot be guaranteed, even on traditional filesystems; on log-structured filesystems or if any block-journaling scheme is in use, this option is completely useless. If the file cannot be overwritten, it will not be removed.

-R Attempt to remove the file hierarchy rooted in each file argument. The **-R** option implies the **-d** option. If the **-i** option is specified, the user is prompted for confirmation before each directory's contents are processed (as well as before the attempt is made to remove the directory). If the user does not respond affirmatively, the file hierarchy rooted in that directory is skipped.

-r Equivalent to **-R**.

-v Cause **rm** to be verbose, showing files as they are processed.

-W Attempts to undelete the named files. Currently, this option can only be used to recover files covered by whiteouts.

The **rm** utility removes symbolic links, not the files referenced by the links.

It is an error to attempt to remove the files "." and "..".

EXIT STATUS

The **rm** utility exits 0 if all of the named files or file hierarchies were removed, or if the **-f** option was specified and all of the existing files or file hierarchies were removed. If an error occurs, **rm** exits with a value >0.

EXAMPLES

rm uses `getopt`(3) standard argument processing. Removing filenames that begin with a dash (e.g., *-file*) in the current directory which might otherwise be taken as option flags to **rm** can be accomplished as follows:

```
rm -- -file
```

or

```
rm ./-file
```

SEE ALSO

rmdir(1), undelete(2), unlink(2), fts(3), getopt(3), symlink(7)

BUGS

The **-P** option assumes that the underlying file system is a fixed-block file system. FFS is a fixed-block file system, LFS is not. In addition, only regular files are overwritten, other types of files are not. Recent research indicates that as many as 35 overwrite passes with carefully chosen data patterns may be necessary to actually prevent recovery of data from a magnetic disk. Thus the **-P** option is likely both insufficient for its design purpose and far too costly for default operation. However, it will at least prevent the recovery of data from FFS volumes with fsdb(8).

COMPATIBILITY

The **rm** utility differs from historical implementations in that the **-f** option only masks attempts to remove non-existent files instead of masking a large variety of errors.

Also, historical BSD implementations prompted on the standard output, not the standard error output.

STANDARDS

The **rm** utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible. The **-v** option is an extension.

The **-P** option attempts to conform to U.S. DoD 5220-22.M, "National Industrial Security Program Operating Manual" ("NISPOM") as updated by Change 2 and the July 23, 2003 "Clearing & Sanitization Matrix". However, unlike earlier revisions of NISPOM, the 2003 matrix imposes requirements which make it clear that the standard does not and can not apply to the erasure of individual files, in particular requirements relating to spare sector management for an entire magnetic disk. *Because these requirements are not met, the* **-P** *option does not conform to the standard*.

NAME

rmdir — remove directories

SYNOPSIS

rmdir [**-p**] *directory ...*

DESCRIPTION

The rmdir utility removes the directory entry specified by each *directory* argument, provided it is empty.

Arguments are processed in the order given. In order to remove both a parent directory and a subdirectory of that parent, the subdirectory must be specified first so the parent directory is empty when **rmdir** tries to remove it.

The following option is available:

-p Each *directory* argument is treated as a pathname of which all components will be removed, if they are empty, starting with the last most component. (See rm(1) for fully non-discriminant recursive removal.)

EXIT STATUS

The **rmdir** utility exits with one of the following values:

0 Each directory entry specified by a dir operand referred to an empty directory and was removed successfully.

>0 An error occurred.

SEE ALSO

rm(1)

STANDARDS

The **rmdir** utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

rpcgen — Remote Procedure Call (RPC) protocol compiler

SYNOPSIS

rpcgen *infile*
rpcgen [**-A**] [**-a**] [**-b**] [**-C**] [**-D** *name* [=value]] [**-I**] [**-i** *size*] [**-K** *secs*] [**-L**] [**-M**]
[**-N**] [**-T**] [**-v**] [**-Y** *pathname*] *infile*
rpcgen **-c** | **-h** | **-l** | **-m** | **-t** | **-Sc** | **-Ss** [**-o** *outfile*] [*infile*]
rpcgen [**-s** *nettype*] [**-o** *outfile*] [*infile*]
rpcgen [**-n** *netid*] [**-o** *outfile*] [*infile*]

DESCRIPTION

rpcgen is a tool that generates C code to implement an RPC protocol. The input to **rpcgen** is a language similar to C known as RPC Language (Remote Procedure Call Language). **rpcgen** is normally used as in the first synopsis where it takes an input file and generates up to four output files. If the *infile* is named `proto.x`, then **rpcgen** will generate a header file in `proto.h`, XDR routines in `proto_xdr.c`, server-side stubs in `proto_svc.c`, and client-side stubs in `proto_clnt.c`. With the **-T** option, it will also generate the RPC dispatch table in `proto_tbl.i`. With the **-Sc** option, it will also generate sample code which would illustrate how to use the remote procedures on the client side. This code would be created in `proto_client.c`. With the **-Ss** option, it will also generate a sample server code which would illustrate how to write the remote procedures. This code would be created in `proto_server.c`.

The server created can be started both by the port monitors (for example, *inetd* or *listen*) or by itself. When it is started by a port monitor, it creates servers only for the transport for which the file descriptor 0 was passed. The name of the transport must be specified by setting up the environmental variable `PM_TRANSPORT`. When the server generated by **rpcgen** is executed, it creates server handles for all the transports specified in `NETPATH` environment variable, or if it is unset, it creates server handles for all the visible transports from `/etc/netconfig` file.

Note: the transports are chosen at run time and not at compile time. When the server is self-started, it backgrounds itself by default. A special define symbol `RPC_SVC_FG` can be used to run the server process in foreground. The second synopsis provides special features which allow for the creation of more sophisticated RPC servers. These features include support for user provided `#defines` and RPC dispatch tables. The entries in the RPC dispatch table contain:

+ pointers to the service routine corresponding to that procedure,
+ a pointer to the input and output arguments,
+ the size of these routines

A server can use the dispatch table to check authorization and then to execute the service routine; a client library may use it to deal with the details of storage management and XDR data conversion.

The other three synopses shown above are used when one does not want to generate all the output files, but only a particular one. Some examples of their usage is described in the **EXAMPLES** section below. When **rpcgen** is executed with the **-s** option, it creates servers for that particular class of transports. When executed with the **-n** option, it creates a server for the transport specified by *netid*. If *infile* is not specified, **rpcgen** accepts the standard input.

The C preprocessor, `cpp`(1) is run on the input file before it is actually interpreted by **rpcgen** For each type of output file, **rpcgen** defines a special preprocessor symbol for use by the **rpcgen** programmer:

`RPC_HDR` defined when compiling into header files

`RPC_XDR` defined when compiling into XDR routines

`RPC_SVC` defined when compiling into server-side stubs

`RPC_CLNT` defined when compiling into client-side stubs

`RPC_TBL` defined when compiling into RPC dispatch tables

Any line beginning with '%' is passed directly into the output file, uninterpreted by **rpcgen**.

For every data type referred to in *infile* **rpcgen** assumes that there exists a routine with the string "xdr_" prepended to the name of the data type. If this routine does not exist in the RPC/XDR library, it must be provided. Providing an undefined data type allows customization of XDR routines.

OPTIONS

-a Generate all the files including sample code for client and server side.

-A Generate an **svc_caller**() function.

-b Compile stubs in "backwards compatible" mode, disabling support for transport-independent RPC. The **-b** should always be specified when generating files for NetBSD, since there is no transport-independent RPC support in NetBSD.

-c Compile into XDR routines.

-C Generate code in ANSI C. This option also generates code that could be compiled with the C++ compiler.

-D *name*[=*value*]
Define a symbol `name`. Equivalent to the `#define` directive in the source. If no `value` is given, `value` is defined as 1. This option may be specified more than once.

-h Compile into C data-definitions (a header file). The **-T** option can be used in conjunction to produce a header file which supports RPC dispatch tables.

-i *size*
Size to decide when to start generating inline code. The default size is 3.

-I Support `inetd`(8) in the server side stubs. Servers generated using this flag can either be standalone or started from `inetd`(8). If a server is started as standalone, then it places itself in the background, unless `RCP_SVC_FG` is defined, or the server is compiled without **-I**.

-K *secs*
By default, services created using **rpcgen** wait 120 seconds after servicing a request before exiting. That interval can be changed using the **-K** flag. To create a server that exits immediately upon servicing a request, "**-K** 0" can be used. To create a server that never exits, the appropriate argument is "**-K** -1".

When monitoring for a server, some port monitors, like the AT&T System V.4 UNIX utility **listen**, *always* spawn a new process in response to a service request. If it is known that a server will be used with such a monitor, the server should exit immediately on completion. For such servers, **rpcgen** should be used with "**-K** -1".

-l Compile into client-side stubs. `inetd`(8).

-I Compile stubs meant for use in programs started by `inetd`(8).

-L Server errors will be sent to syslog instead of stderr.

-m Compile into server-side stubs, but do not generate a **main**() routine. This option is useful for doing callback-routines and for users who need to write their own **main**() routine to do initialization.

-M Generate thread-safe stubs. This alters the calling pattern of client and server stubs so that storage for results is allocated by the caller. Note that all components for a particular service (stubs, client and service wrappers, etc.) must be built either with or without the **-M** flag.

-N Use the newstyle of **rpcgen**. This allows procedures to have multiple arguments. It also uses the style of parameter passing that closely resembles C. So, when passing an argument to a remote procedure you do not have to pass a pointer to the argument but the argument itself. This behaviour is different from the oldstyle of **rpcgen** generated code. The newstyle is not the default case because of backward compatibility.

-n *netid*
Compile into server-side stubs for the transport specified by *netid*. There should be an entry for *netid* in the netconfig database. This option may be specified more than once, so as to compile a server that serves multiple transports.

-o *outfile*
Specify the name of the output file. If none is specified, standard output is used (**-c -h -l -m -n -s** modes only)

-n *netid*
Specify the transport for the server-side stubs. *netid* should be defined in netconfig(5). This option can be repeated in order to support more than one transport.

-s *nettype*
Compile into server-side stubs for all the transports belonging to the class *nettype*. The supported classes are *netpath*, *visible*, *circuit_n*, *circuit_v*, *datagram_n*, *datagram_v*, *tcp*, and *udp* [see rpc(3) for the meanings associated with these classes. *Note*: BSD currently supports only the *tcp* and *udp* classes]. This option may be specified more than once. *Note*: the transports are chosen at run time and not at compile time.

-Sc Generate sample code to show the use of remote procedure and how to bind to the server before calling the client side stubs generated by **rpcgen**.

-Ss Generate skeleton code for the remote procedures on the server side. You would need to fill in the actual code for the remote procedures.

-t Compile into RPC dispatch table.

-T Generate the code to support RPC dispatch tables.

-v Display the version number.

-Y *pathname*
Specify the directory where **rpcgen** looks for the C pre-processor.

The options **-c**, **-h**, **-l**, **-m**, **-s**, and **-t** are used exclusively to generate a particular type of file, while the options **-D** and **-T** are global and can be used with the other options.

ENVIRONMENT

If the RPCGEN_CPP environment variable is set, its value is used as the pathname of the C preprocessor to be run on the input file.

NOTES

The RPC Language does not support nesting of structures. As a work-around, structures can be declared at the top-level, and their name used inside other structures in order to achieve the same effect.

Name clashes can occur when using program definitions, since the apparent scoping does not really apply. Most of these can be avoided by giving unique names for programs, versions, procedures and types.

The server code generated with **-n** option refers to the transport indicated by *netid* and hence is very site specific.

EXAMPLES

The command

```
$ rpcgen -T prot.x
```

generates the five files: `prot.h`, `prot_clnt.c`, `prot_svc.c`, `prot_xdr.c` and `prot_tbl.i`.

The following example sends the C data-definitions (header file) to standard output.

```
$ rpcgen -h prot.x
```

To send the test version of the `-DTEST`, server side stubs for all the transport belonging to the class *datagram_n* to standard output, use:

```
$ rpcgen -s datagram_n -DTEST prot.x
```

To create the server side stubs for the transport indicated by *netid tcp*, use:

```
$ rpcgen -n tcp -o prot_svc.c prot.x
```

SEE ALSO

`cpp`(1), `inetd`(8)

HISTORY

The **-M** option was first implemented in RedHat Linux, and was reimplemented by Charles M. Hannum in NetBSD 1.6.

NAME

rs — reshape a data array

SYNOPSIS

rs [**-CcSs** [*x*]] [**-GgKkw** *N*] [**-EeHhjmnTty**] [*rows* [*cols*]]

DESCRIPTION

rs reads the standard input, interpreting each line as a row of blank-separated entries in an array, transforms the array according to the options, and writes it on the standard output. With no arguments it transforms stream input into a columnar format convenient for terminal viewing.

The shape of the input array is deduced from the number of lines and the number of columns on the first line. If that shape is inconvenient, a more useful one might be obtained by skipping some of the input with the **-k** option. Other options control interpretation of the input columns.

The shape of the output array is influenced by the *rows* and *cols* specifications, which should be positive integers. If only one of them is a positive integer, **rs** computes a value for the other which will accommodate all of the data. When necessary, missing data are supplied in a manner specified by the options and surplus data are deleted. There are options to control presentation of the output columns, including transposition of the rows and columns.

The options are described below.

-C [*x*] Output columns are delimited by the single character *x*. A missing *x* is taken to be '^I'.

-c [*x*] Input columns are delimited by the single character *x*. A missing *x* is taken to be '^I'.

-e Consider each line of input as an array entry.

-G *N* The gutter width (inter-column space) has *N* percent of the maximum column width added to it.

-g *N* The gutter width (inter-column space), normally 2, is taken to be *N*.

-H Like **-h**, but also print the length of each line.

-h Print the shape of the input array and do nothing else. The shape is just the number of lines and the number of entries on the first line.

-j Right adjust entries within columns.

-K *N* Like **-k**, but print the ignored lines.

-k *N* Ignore the first *N* lines of input.

-m Do not trim excess delimiters from the ends of the output array.

-n On lines having fewer entries than the first line, use null entries to pad out the line. Normally, missing entries are taken from the next line of input.

-S [*x*] Like **-C**, but padded strings of *x* are delimiters.

-s [*x*] Like **-c**, but maximal strings of *x* are delimiters.

-T Print the pure transpose of the input, ignoring any *rows* or *cols* specification.

-t Fill in the rows of the output array using the columns of the input array, that is, transpose the input while honoring any *rows* and *cols* specifications.

-w *N* The width of the display, normally 80, is taken to be the positive integer *N*.

-y If there are too few entries to make up the output dimensions, pad the output by recycling the input from the beginning. Normally, the output is padded with blanks.

-z Adapt column widths to fit the largest entries appearing in them.

With no arguments, **rs** transposes its input, and assumes one array entry per input line unless the first non-ignored line is longer than the display width. Option letters which take numerical arguments interpret a missing number as zero unless otherwise indicated.

EXAMPLES

rs can be used as a filter to convert the stream output of certain programs (e.g., spell(1), du(1), file(1), look(1), nm(1), who(1), and wc(1)) into a convenient "window" format, as in

```
who | rs
```

This function has been incorporated into the ls(1) program, though for most programs with similar output **rs** suffices.

To convert stream input into vector output and back again, use

```
rs 1 0 | rs 0 1
```

A 10 by 10 array of random numbers from 1 to 100 and its transpose can be generated with

```
jot -r 100 | rs 10 10 | tee array | rs -T > tarray
```

In the editor vi(1), a file consisting of a multi-line vector with 9 elements per line can undergo insertions and deletions, and then be neatly reshaped into 9 columns with

```
:1,$!rs 0 9
```

Finally, to sort a database by the first line of each 4-line field, try

```
rs -eC 0 4 | sort | rs -c 0 1
```

SEE ALSO

jot(1), pr(1), sort(1), vi(1)

BUGS

Handles only two dimensional arrays.

The algorithm currently reads the whole file into memory, so files that do not fit in memory will not be reshaped.

Fields cannot be defined yet on character positions.

Re-ordering of columns is not yet possible.

There are too many options.

NAME

rsh — remote shell

SYNOPSIS

rsh [**-45FGKdefnuxz**][**-U** string][**-p** *port*][**-l** *username*][**-P** *N*|*O*]*host* *[command]*

DESCRIPTION

rsh authenticates to the rshd(8) daemon on the remote *host*, and then executes the specified *command*.

rsh copies its standard input to the remote command, and the standard output and error of the remote command to its own.

Valid options are:

-4, **--krb4**
: The **-4** option requests Kerberos 4 authentication. Normally all supported authentication mechanisms will be tried, but in some cases more explicit control is desired.

-5, **--krb5**
: The **-5** option requests Kerberos 5 authentication. This is analogous to the **-4** option.

-K, **--broken**
: The **-K** option turns off all Kerberos authentication. The security in this mode relies on reserved ports. The long name is an indication of how good this is.

-n, **--no-input**
: The **-n** option directs the input from the /dev/null device (see the **BUGS** section of this manual page).

-d
: Enable setsockopt(2) socket debugging.

-e, **--no-stderr**
: Don't use a separate socket for the stderr stream. This can be necessary if rsh-ing through a NAT bridge.

-x, **--encrypt**
: The **-x** option enables encryption for all data exchange. This is only valid for Kerberos authenticated connections (see the **BUGS** section for limitations).

-z
: The opposite of **-x**. This is the default, and is mainly useful if encryption has been enabled by default, for instance in the appdefaults section of /etc/krb5.conf when using Kerberos 5.

-f, **--forward**
: Forward Kerberos 5 credentials to the remote host. Also settable via appdefaults (see krb5.conf).

-F, **--forwardable**
: Make the forwarded credentials re-forwardable. Also settable via appdefaults (see krb5.conf).

-l *string*, **--user=***string*
: By default the remote username is the same as the local. The **-l** option or the username@host format allow the remote name to be specified.

-n, **--no-input**
: Direct input from /dev/null (see the **BUGS** section).

-p *number-or-service*, **--port=***number-or-service*
Connect to this port instead of the default (which is 514 when using old port based authentication, 544 for Kerberos 5 and non-encrypted Kerberos 4, and 545 for encrytpted Kerberos 4; subject of course to the contents of `/etc/services`).

-P *N|O|1|2*, **--protocol=***N|O|1|2*
Specifies the protocol version to use with Kerberos 5. *N* and *2* select protocol version 2, while *O* and *1* select version 1. Version 2 is believed to be more secure, and is the default. Unless asked for a specific version, **rsh** will try both. This behaviour may change in the future.

-u, **--unique**
Make sure the remote credentials cache is unique, that is, don't reuse any existing cache. Mutually exclusive to **-U**.

-U `string`, **--tkfile=**`string`
Name of the remote credentials cache. Mutually exclusive to **-u**.

-x, **--encrypt**
The **-x** option enables encryption for all data exchange. This is only valid for Kerberos authenticated connections (see the **BUGS** section for limitations).

-z The opposite of **-x**. This is the default, but encryption can be enabled when using Kerberos 5, by setting the `libdefaults/encrypt` option in `krb5.conf`(5).

EXAMPLES

Care should be taken when issuing commands containing shell meta characters. Without quoting, these will be expanded on the local machine.

The following command:

```
rsh otherhost cat remotefile > localfile
```

will write the contents of the remote `remotefile` to the local `localfile`, but:

```
rsh otherhost 'cat remotefile > remotefile2'
```

will write it to the remote `remotefile2`.

FILES

`/etc/hosts`

SEE ALSO

`rlogin`(1), `krb_realmofhost`(3), `krb_sendauth`(3), `hosts.equiv`(5), `krb5.conf`(5), `rhosts`(5), `kerberos`(8) `rshd`(8)

HISTORY

The **rsh** command appeared in 4.2BSD.

AUTHORS

This implementation of **rsh** was written as part of the Heimdal Kerberos 5 implementation.

BUGS

Some shells (notably `csh`(1)) will cause **rsh** to block if run in the background, unless the standard input is directed away from the terminal. This is what the **-n** option is for.

The **-x** options enables encryption for the session, but for both Kerberos 4 and 5 the actual command is sent unencrypted, so you should not send any secret information in the command line (which is probably a bad

idea anyway, since the command line can usually be read with tools like ps(1)). Forthermore in Kerberos 4 the command is not even integrity protected, so anyone with the right tools can modify the command.

NAME

rsh — remote shell

SYNOPSIS

rsh [**-46dn**] [**-l** *username*] [**-p** *port*] *host* [command]
rsh [**-46dn**] [**-p** *port*] *username@host* [command]

DESCRIPTION

rsh executes *command* on *host*.

rsh copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; **rsh** normally terminates when the remote command does. The options are as follows:

-4 Use IPv4 addresses only.

-6 Use IPv6 addresses only.

-d The **-d** option turns on socket debugging (using `setsockopt`(2)) on the TCP sockets used for communication with the remote host.

-l *username* By default, the remote username is the same as the local username. The **-l** option or the *username@host* format allow the remote name to be specified.

-n The **-n** option redirects input from the special device `/dev/null` (see the **BUGS** section of this manual page).

-p *port* Uses the given `port` instead of the one assigned to the service "shell". May be given either as symbolic name or as number. If no command is given, note that `rlogin`(1) is started, which may need a different daemon (`rlogind`(8) instead of `rshd`(8)) running on the server; you want to pass the `rshd`(8) port number in that case.

If no *command* is specified, you will be logged in on the remote host using `rlogin`(1).

Shell metacharacters which are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine. For example, the command

```
rsh otherhost cat remotefile >> localfile
```

appends the remote file *remotefile* to the local file *localfile*, while

```
rsh otherhost cat remotefile ">>" other_remotefile
```

appends *remotefile* to *other_remotefile*.

FILES

`/etc/hosts`

SEE ALSO

`rcmd`(1), `rlogin`(1), `rcmd`(3), `hosts.equiv`(5), `rhosts`(5), `environ`(7)

HISTORY

The **rsh** command appeared in 4.2BSD.

BUGS

If you are using `csh`(1) and put a **rsh** in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. If no input is desired you should redi-

rect the input of **rsh** to /dev/null using the **-n** option.

You cannot run an interactive command (like rogue(6) or vi(1)) using **rsh**; use rlogin(1) instead.

Stop signals stop the local **rsh** process only; this is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

NAME

run—Simulator front-end

SYNOPSIS

run [**−v**] [**−p** *freq*] [**−m** *memory*] [**−-sysroot** *filepath*] *program*

DESCRIPTION

Use ‘ **run** *program* ’ to execute a binary by interpreting machine instructions on your host computer.

run is the same emulator used by GDB’s ‘ **target sim** ’ command. You can run it directly by executing **run** if you just want to see your program execute, and do not need any debugger functionality. You can also use **run** to generate profiling information for analysis with **gprof**.

OPTIONS

−v Verbose output. Display the name of the program to run before execution; after execution, display the number of instructions executed, the number of machine cycles emulated, the number of pipeline stalls, the real time taken, the emulated execution time taken, and a summary of how much profiling information was generated.

−p *freq* Generate profile information (for use with **gprof**). *freq* is the profiling frequency. Write the profiling information to a file called **gmon.out**.

−m *memory*
Set the memory size for the emulated machine to two to the power *memory*. The default value is 19, emulating a board with 524288 bytes of memory.

−-sysroot *filepath*
Prepend *filepath* to all simulator system calls that pass absolute file paths. Change working directory to *filepath* at program start. Not all simulators support this option; those that don’t, will ignore it.

SEE ALSO

‘ **gprof** ’ entry in **info**; ‘ **gdb** ’ entry in **info**; *Using GDB: A Guide to the GNU Source-Level Debugger*, Richard M. Stallman and Roland H. Pesch.

COPYING

NAME

runidn − A script to allow applications to use internationalized domain names.

SYNOPSIS

runidn [**-e** *local-codeset*] *program-name* [*args..*]

DESCRIPTION

runidn enables applications to use internationalized domain names without recompilation. Just add ‘‘runidn’’ before the application-name, and the application can handle non-ASCII domain names. For example, you can do:

```
% runidn telnet non-ASCII-hostname
```

Before using runidn, you should set up properties related to internationalized DNS by configuring idnkit's configuration file **idn.conf**. See idn.conf(5) which describes the configuration.

OPTION

The following option is available:

−e *local-codeset*

Specify the application's local codeset. If the option is not specified, **runidn** guesses the codeset from the current locale. See the ‘‘NOTE’’ section for more details about local codeset.

IMPLEMENTATION

runidn is a small shell script that sets up an environment variable called ‘‘LD_PRELOAD’’, so that an application dynamically links a shared library ‘‘libidnkitres’’ before any other shared libraries.

The library ‘‘libidnkitres’’ provides a special version of resolver functions which implement features for handling internationalized domain names. **runidn** replaces the following functions with the special version:

```
gethostbyname
gethostbyname2
gethostbyaddr
gethostbyname_r
gethostbyname2_r
gethostbyaddr_r
getipnodebyname
getipnodebyaddr
freehostent
getaddrinfo
freeaddrinfo
getnameinfo
```

By overriding them in the standard libraries with the special version provided by ‘‘libidnkitres’’, **runidn** enables applications to use internationalized domain names.

- These API functions accept non-ASCII domain names encoded in the local codeset that the application is using. Also the result from these APIs may contain non-ASCII domain names.
- The normalization and codeset conversion between application's local codeset and the codeset used in DNS protocol data are handled automatically, so users/applications need not worry about them.

Properties of internationalized DNS (such as the normalization or the codeset used on DNS protocol data) can be configured with the idnkit's configuration file (**idn.conf**). See idn.conf(5) for details.

NOTE

Unless **−e** option is specified, **runidn** tries to guess the application's local codeset from the application's current locale. However, sometimes it cannot guess the codeset correctly, for example if the application does not set the locale appropriately by calling ‘setlocale()’. In that case, you can explicitly specify the local codeset by setting an environment variable ‘‘IDN_LOCAL_CODESET’’. See the section ‘‘LOCAL CODESET’’ in idn.conf(5) for details.

The idea of using ‘‘LD_PRELOAD’’ to replace some functions in the standard library was taken from ‘‘runsocks’’ script distributed as part of SOCKS5 reference implementation.

BUGS

There are many cases where **runidn** does not work.

Your system must support ‘‘LD_PRELOAD’’ mechanism in the first place.

Due to security reasons, ‘‘LD_PRELOAD’’ mechanism is disabled for setuid programs in any sane systems. So **runidn** does not work for setuid programs such as ping or rsh.

If your application uses a function other than the ones runidn supports for name resolution, you lose.

SEE ALSO

idn.conf(5), runsocks(1)

NAME

rup — remote status display

SYNOPSIS

rup [**-dhlt**] [*host ...*]

DESCRIPTION

rup displays a summary of the current system status of a particular *host* or all hosts on the local network. The output shows the current time of day, how long the system has been up, and the load averages. The load average numbers give the number of jobs in the run queue averaged over 1, 5 and 15 minutes.

The following options are available:

-d For each host, report what it's local time is. This is useful for checking time synchronization on a network.

-h Sort the display alphabetically by host name.

-l Sort the display by load average.

-t Sort the display by up time.

The `rpc.rstatd`(8) daemon must be running on the remote host for this command to work. **rup** uses an RPC protocol defined in /usr/include/rpcsvc/rstat.x.

EXAMPLES

```
example% rup otherhost
otherhost      up 6 days, 16:45,  load average: 0.20, 0.23, 0.18
example%
```

DIAGNOSTICS

rup: RPC: Program not registered

The `rpc.rstatd`(8) daemon has not been started on the remote host.

rup: RPC: Timed out

A communication error occurred. Either the network is excessively congested, or the `rpc.rstatd`(8) daemon has terminated on the remote host.

rup: RPC: Port mapper failure - RPC: Timed out

The remote host is not running the portmapper (see `rpcbind`(8)), and cannot accommodate any RPC-based services. The host may be down.

SEE ALSO

`ruptime`(1), `rpc.rstatd`(8), `rpcbind`(8)

HISTORY

The **rup** command appeared in SunOS.

NAME

ruptime — show host status of local machines

SYNOPSIS

ruptime [**-alrtu**]

DESCRIPTION

ruptime gives a status line like uptime(1) for each machine on the local network; these are formed from packets broadcast by each host on the network via the rwhod(8) daemon. The default broadcast time by the hosts is every three minutes.

Machines for which no status report has been received for 11 minutes are shown as being down.

The options are as follows:

-a Users idle an hour or more are not counted unless the **-a** flag is given.

-l Sort by load average.

-r Reverses the sort order.

-t Sort by uptime.

-u Sort by number of users.

The default listing is sorted by host name.

FILES

/var/rwho/whod.* data files

SEE ALSO

rup(1), rwho(1), uptime(1), rwhod(8)

HISTORY

ruptime appeared in 4.2 BSD.

NAME

rusers — who is logged in to machines on local network

SYNOPSIS

rusers [**-al**] [*host ...*]

DESCRIPTION

The **rusers** command produces output similar to who(1), but for the list of hosts or all machines on the local network. For each host responding to the rusers query, the hostname with the names of the users currently logged on is printed on each line. The rusers command will wait for one minute to catch late responders.

The following options are available:

-a Print all machines responding even if no one is currently logged in.

-l Print a long format listing. This includes the user name, host name, tty that the user is logged in to, the date and time the user logged in, the idle time (in minutes), and the remote host they logged in from (if applicable).

DIAGNOSTICS

rusers: RPC: Program not registered
: The rpc.rusersd(8) daemon has not been started on the remote host.

rusers: RPC: Timed out
: A communication error occurred. Either the network is excessively congested, or the rpc.rusersd(8) daemon has terminated on the remote host.

rusers: RPC: Port mapper failure - RPC: Timed out
: The remote host is not running the portmapper (see rpcbind(8)), and cannot accommodate any RPC-based services. The host may be down.

SEE ALSO

rwho(1), users(1), who(1), rpc.rusersd(8), rpcbind(8)

HISTORY

The **rusers** command appeared in SunOS.

BUGS

The sorting options are not implemented.

NAME

rwall — send a message to users logged on a host

SYNOPSIS

rwall *host* [*file*]

DESCRIPTION

The **rwall** command sends a message to the users logged into the specified host. The message to be sent can be typed in and terminated with EOF or it can be in a *file*.

DIAGNOSTICS

rwall: RPC: Program not registered

The rpc.rwalld(8) daemon has not been started on the remote host.

rwall: RPC: Timed out

A communication error occurred. Either the network is excessively congested, or the rpc.rwalld(8) daemon has terminated on the remote host.

rwall: RPC: Port mapper failure - RPC: Timed out

The remote host is not running the portmapper (see rpcbind(8)), and cannot accommodate any RPC-based services. The host may be down.

SEE ALSO

wall(1), rpc.rwalld(8), rpcbind(8)

HISTORY

The **rwall** command appeared in SunOS.

NAME

rwho — who is logged in on local machines

SYNOPSIS

rwho [**-aHq**]

DESCRIPTION

The **rwho** command produces output similar to who(1), but for all machines on the local network. If no report has been received from a machine for 11 minutes then **rwho** assumes the machine is down, and does not report the users last known to be logged into that machine.

If a user hasn't typed to the system for a minute or more, then **rwho** reports this idle time.

-a Include all users. By default, if a user hasn't typed to the system for an hour or more, then the user will be omitted from the output.

-H Write column headings above the regular output.

-q "Quick mode": List only the names and the number of users currently logged on. When this option is used, all other options are ignored.

FILES

/var/rwho/whod.* information about other machines

SEE ALSO

finger(1), rup(1), ruptime(1), rusers(1), who(1), rwhod(8)

HISTORY

The **rwho** command appeared in 4.3BSD.

BUGS

This is unwieldy when the number of machines on the local net is large.

NAME

rxtelnet — start a telnet and forward X-connections.

SYNOPSIS

rxtelnet [**-l** *username*] [**-k**] [**-t** *telnet_args*] [**-x** *xterm_args*] [**-K** *kx_args*] [**-w** *term_emulator*] [**-b** *telnet_program*] [**-n**] [**-v**] *host* [*port*]

DESCRIPTION

The **rxtelnet** program starts an **xterm** window with a telnet to host *host*. From this window you will also be able to run X clients that will be able to connect securely to your X server. If *port* is given, that port will be used instead of the default.

If setting up the X forwarding failes, **rxtelnet** will still telnet in to the remote host, but without X forwarding.

The supported options are:

-l Log in on the remote host as user *username*.

-k Disables keep-alives.

-t Send *telnet_args* as arguments to **telnet**.

-x Send *xterm_args* as arguments to **xterm**.

-X Send *kx_args* as arguments to **kx**.

-w Use *term_emulator* instead of xterm.

-b Use *telnet_program* instead of telnet.

-n Do not start any terminal emulator.

-v Be verbose.

EXAMPLE

To login from host *foo* (where your display is) to host *bar*, you might do the following.

1. On foo: **rxtelnet** *bar*
2. You will get a new window with a **telnet** to *bar*. In this window you will be able to start X clients.

SEE ALSO

kx(1), rxterm(1), telnet(1), tenletxr(1), kxd(8)

NAME

rxterm — start a secure remote xterm

SYNOPSIS

rxterm [**-l** *username*] [**-k**] [**-r** *rsh_args*] [**-x** *xterm_args*] [**-K** *kx_args*] [**-w** *term_emulator*] [**-b** *rsh_program*] *host* [*port*]

DESCRIPTION

The **rxterm** program starts an **xterm** window on host *host*. From this window you will also be able to run X clients that will be able to connect securely to your X server. If *port* is given, that port will be used instead of the default.

The supported options are:

-l Log in on the remote host as user *username*.

-k Disable keep-alives.

-r Send *rsh_args* as arguments to **rsh**.

-x Send *xterm_args* as arguments to **xterm**.

-X Send *kx_args* as arguments to **kx**.

-w Use *term_emulator* instead of xterm.

-b Use *rsh_program* instead of rsh.

-v Be verbose.

EXAMPLE

To login from host *foo* (where your display is) to host *bar*, you might do the following.

1. On foo: **rxterm** *bar*
2. You will get a new window running an **xterm** on host *bar*. In this window you will be able to start X clients.

SEE ALSO

kx(1), rsh(1), rxtelnet(1), tenletxr(1), kxd(8)

; Sample input data file

;#server ns.sector93.ie ;#port 53 ;#maxwait 10

1-host.com ANY 1-linknet.com A 1.0-127.35.195.200.in-addr.arpa PTR 1.0-63.236.210.200.in-addr.arpa PTR 1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.int PTR 1.0.0.0.in-addr.arpa PTR 1.0.0.10.in-addr.arpa PTR

NAME

scp — secure copy (remote file copy program)

SYNOPSIS

scp [**-1246BCpqrv**] [**-c** *cipher*] [**-F** *ssh_config*] [**-i** *identity_file*] [**-l** *limit*] [**-o** *ssh_option*] [**-P** *port*] [**-S** *program*] [[*user*@]*host1*:]*file1 . . .* [[*user*@]*host2*:]*file2*

DESCRIPTION

scp copies files between hosts on a network. It uses ssh(1) for data transfer, and uses the same authentication and provides the same security as ssh(1). Unlike rcp(1), **scp** will ask for passwords or passphrases if they are needed for authentication.

File names may contain a user and host specification to indicate that the file is to be copied to/from that host. Local file names can be made explicit using absolute or relative pathnames to avoid **scp** treating file names containing ':' as host specifiers. Copies between two remote hosts are also permitted.

The options are as follows:

-1 Forces **scp** to use protocol 1.

-2 Forces **scp** to use protocol 2.

-4 Forces **scp** to use IPv4 addresses only.

-6 Forces **scp** to use IPv6 addresses only.

-B Selects batch mode (prevents asking for passwords or passphrases).

-C Compression enable. Passes the **-C** flag to ssh(1) to enable compression.

-c *cipher*
Selects the cipher to use for encrypting the data transfer. This option is directly passed to ssh(1).

-F *ssh_config*
Specifies an alternative per-user configuration file for **ssh**. This option is directly passed to ssh(1).

-i *identity_file*
Selects the file from which the identity (private key) for RSA authentication is read. This option is directly passed to ssh(1).

-l *limit*
Limits the used bandwidth, specified in Kbit/s.

-o *ssh_option*
Can be used to pass options to **ssh** in the format used in ssh_config(5). This is useful for specifying options for which there is no separate **scp** command-line flag. For full details of the options listed below, and their possible values, see ssh_config(5).

AddressFamily
BatchMode
BindAddress
ChallengeResponseAuthentication
CheckHostIP
Cipher
Ciphers

Compression
CompressionLevel
ConnectionAttempts
ConnectTimeout
ControlMaster
ControlPath
GlobalKnownHostsFile
GSSAPIAuthentication
GSSAPIDelegateCredentials
HashKnownHosts
Host
HostbasedAuthentication
HostKeyAlgorithms
HostKeyAlias
HostName
IdentityFile
IdentitiesOnly
KbdInteractiveDevices
LogLevel
MACs
NoHostAuthenticationForLocalhost
NumberOfPasswordPrompts
PasswordAuthentication
Port
PreferredAuthentications
Protocol
ProxyCommand
PubkeyAuthentication
RekeyLimit
RhostsRSAAuthentication
RSAAuthentication
SendEnv
ServerAliveInterval
ServerAliveCountMax
SmartcardDevice
StrictHostKeyChecking
TCPKeepAlive
UsePrivilegedPort
User
UserKnownHostsFile
VerifyHostKeyDNS

-P *port*
: Specifies the port to connect to on the remote host. Note that this option is written with a capital 'P', because **-p** is already reserved for preserving the times and modes of the file in `rcp`(1).

-p
: Preserves modification times, access times, and modes from the original file.

-q
: Quiet mode: disables the progress meter as well as warning and diagnostic messages from `ssh`(1).

-r
: Recursively copy entire directories.

-S *program*
Name of *program* to use for the encrypted connection. The program must understand ssh(1) options.

-v Verbose mode. Causes **scp** and ssh(1) to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.

The **scp** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

rcp(1), sftp(1), ssh(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), ssh_config(5), sshd(8)

HISTORY

scp is based on the rcp(1) program in BSD source code from the Regents of the University of California.

AUTHORS

Timo Rinne ⟨tri@iki.fi⟩
Tatu Ylonen ⟨ylo@cs.hut.fi⟩

NAME

screenblank — screen saver daemon for wscons and FBIO machines

SYNOPSIS

screenblank [**-k** | **-m**] [**-d** *inactivity-timeout*] [**-e** *wakeup-delay*]
[**-f** *framebuffer*] [**-i** *input-device*]
screenblank { **-b** | **-u** }

DESCRIPTION

screenblank disables the framebuffer if the keyboard and mouse are idle for a period of time, and re-enables the framebuffer when keyboard or mouse activity resumes.

When killed with a SIGINT, SIGHUP, or SIGTERM, **screenblank** will re-enable the framebuffer. The pid can be found in the file `/var/run/screenblank.pid`.

The options are as follows:

-b Overriding the other options, simply try (once) to blank the framebuffer, then exit.

-d *inactivity-timeout*
Wait the number of seconds specified by *inactivity-timeout*, expressed in the format "xxx.xxx", before disabling the framebuffer due to inactivity. The default is 600 seconds (10 minutes).

-e *wakeup-delay*
Wait the number of seconds specified by *wakeup-delay*, expressed in the format "xxx.xxx", before re-enabling the framebuffer once activity resumes. The default is .25 seconds.

-f *framebuffer*
Use the framebuffer device *framebuffer* instead of the default `/dev/fb`.

-i *input-device*
Add *input-device* to the list of devices to monitor for activity.

-k Do not check the keyboard for activity.

-m Do not check the mouse for activity.

-u Overriding the other options, simply try (once) to unblank the framebuffer, then exit.

Note that the **-k** and **-m** flags are mutually exclusive.

FILES

/dev/kbd	The keyboard device.
/dev/mouse	The mouse device.
/dev/console	The console device.
/dev/fb	The default framebuffer.
/dev/wskbd	The keyboard for wscons machines.
/dev/wsmouse	The mouse device for wscons machines.
/dev/ttyE0	The console device for wscons machines.
/var/run/screenblank.pid	File containing the pid of **screenblank**.

NAME

script — make typescript of terminal session

SYNOPSIS

script [**-adpr**] [*file*]

DESCRIPTION

script makes a typescript of everything printed on your terminal. It is useful for students who need a hardcopy record of an interactive session as proof of an assignment, as the typescript file can be printed out later with lpr(1).

If the argument *file* is given, **script** saves all dialogue in *file*. If no file name is given, the typescript is saved in the file typescript.

Option:

-a Append the output to *file* or typescript, retaining the prior contents.

-d Don't sleep between records when playing back a timestamped session.

-p Play back a recorded session in real time.

-r Record a session with input, output, and timestamping.

The script ends when the forked shell exits (a *control-D* to exit the Bourne shell (sh(1)), and *exit*, *logout* or *control-d* (if *ignoreeof* is not set) for the C-shell, csh(1)).

Certain interactive commands, such as vi(1), create garbage in the typescript file. **script** works best with commands that do not manipulate the screen, the results are meant to emulate a hardcopy terminal.

ENVIRONMENT

The following environment variable is used by **script**:

SHELL If the variable SHELL exists, the shell forked by **script** will be that shell. If SHELL is not set, the Bourne shell is assumed. (Most shells set this variable automatically).

SEE ALSO

csh(1) (for the *history* mechanism).

HISTORY

The **script** command appeared in 3.0 BSD.

BUGS

script places **everything** in the log file, including linefeeds and backspaces. This is not what the naive user expects.

NAME

sdiff − side-by-side merge of file differences

SYNOPSIS

sdiff [*OPTION*]... *FILE1 FILE2*

DESCRIPTION

Side-by-side merge of file differences.

−o FILE **−−output**=*FILE*
Operate interactively, sending output to FILE.

−i −−ignore−case
Consider upper- and lower-case to be the same.

−E −−ignore−tab−expansion
Ignore changes due to tab expansion.

−b −−ignore−space−change
Ignore changes in the amount of white space.

−W −−ignore−all−space
Ignore all white space.

−B −−ignore−blank−lines
Ignore changes whose lines are all blank.

−I RE **−−ignore−matching−lines**=*RE*
Ignore changes whose lines all match RE.

−−strip−trailing−cr
Strip trailing carriage return on input.

−a −−text
Treat all files as text.

−w NUM **−−width**=*NUM*
Output at most NUM (default 130) columns per line.

−l −−left−column
Output only the left column of common lines.

−s −−suppress−common−lines
Do not output common lines.

−t −−expand−tabs
Expand tabs to spaces in output.

−d −−minimal
Try hard to find a smaller set of changes.

−H −−speed−large−files
Assume large files and many scattered small changes.

−−diff−program=*PROGRAM*
Use PROGRAM to compare files.

−v −−version
Output version info.

−−help Output this help.

If a FILE is ‘-’, read standard input.

AUTHOR

Written by Thomas Lord.

REPORTING BUGS

Report bugs to <bug-gnu-utils@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **sdiff** is maintained as a Texinfo manual. If the **info** and **sdiff** programs are properly installed at your site, the command

info diff

should give you access to the complete manual.

NAME

sdiff — side-by-side diff

SYNOPSIS

sdiff [**-abdilstW**] [**-I** *regexp*] [**-o** *outfile*] [**-w** *width*] *file1 file2*

DESCRIPTION

Shows a side-by-side comparison of two files.

The options are:

-l Only print the left column for identical lines.

-o *outfile*
Interactively merge *file1* and *file2* into *outfile*. In this mode, the user is prompted for each set of differences. See EDITOR and VISUAL, below, for details of which editor, if any, is invoked.

The commands are as follows:

l Choose left set of diffs.

r Choose right set of diffs.

s Silent mode – identical lines are not printed.

v Verbose mode – identical lines are printed.

e Start editing an empty file, which will be merged into *outfile* upon exiting the editor.

e l Start editing file with left set of diffs.

e r Start editing file with right set of diffs.

e b Start editing file with both sets of diffs.

q Quit **sdiff**.

-s Skip identical lines.

-w *width*
Print a maximum of *width* characters on each line. The default is 130 characters.

Options passed to diff(1) are:

-a Treat *file1* and *file2* as text files.

-b Ignore trailing blank spaces.

-d Minimize diff size.

-I *regexp*
Ignore line changes matching *regexp*. All lines in the change must match *regexp* for the change to be ignored.

-i Do a case-insensitive comparison.

-t Expand tabs to spaces.

-W Ignore all spaces (the **-w** flag is passed to diff(1)).

ENVIRONMENT

`EDITOR`, `VISUAL`
: Specifies an editor to use with the **`-o`** option. If both `EDITOR` and `VISUAL` are set, `VISUAL` takes precedence. If neither `EDITOR` nor `VISUAL` are set, the default is `vi`(1).

`TMPDIR`
: Specifies a directory for temporary files to be created. The default is `/tmp`.

SEE ALSO

`diff`(1), `diff3`(1), `vi`(1), `re_format`(7)

AUTHORS

`sdiff` was written from scratch for the public domain by Ray Lai ⟨ray@cyth.net⟩.

CAVEATS

Although undocumented, **`sdiff`** supports all options supported by GNU sdiff. Some options require GNU diff.

Tabs are treated as anywhere from one to eight characters wide, depending on the current column. Terminals that treat tabs as eight characters wide will look best.

BUGS

`sdiff` may not work with binary data.

NAME

sdpquery — SDP query utility

SYNOPSIS

sdpquery -h
sdpquery [**-d** *device*] **-a** *address command* [*parameters ...*]
sdpquery [**-c** *path*] **-l** *command* [*parameters ...*]

DESCRIPTION

The **sdpquery** utility attempts to query the specified Service Discovery Protocol (SDP) server. Remote SDP servers are identified by their address. Connection to the local SDP server is made via the control socket. The **sdpquery** utility uses Service Search Attribute Requests and prints results to standard output and error messages to standard error.

The options are as follows:

-a *address* Connect to the remote device with the specified address. The address can be specified as BD_ADDR or a name. If a name was specified, the **sdpquery** utility attempts to resolve the name via `bt_gethostbyname`(3). If no remote address is given, **sdpquery** attempts to contact a local SDP server via the control socket.

-c *path* Specify path to the control socket. The default path is `/var/run/sdp`.

-d *device* Connect from the local device with the specifed address. The device can be specified by BD_ADDR or device name. See `btconfig`(8) for a list of devices available. If no device is specified, the **sdpquery** utility will use the best available.

-l Query the local SDP server via the control socket.

-h Display usage message and exit.

The currently supported commands in **sdpquery** are:

Browse [*group*] Browse for services. The *group* parameter is a 16-bit UUID of the group to browse. If omitted, the Public Browse Group. is used.

Search *service* Search for the *service*. The *service* parameter is a 16-bit UUID of the service to search for. For the following services it is possible to use service name instead of service UUID:

CIP	Common ISDN Access
CTP	Cordless Telephony
DUN	DialUp Networking
FAX	Fax
FTRN	OBEX File Transfer
GN	Group ad-hoc Network
HID	Human Interface Device
HF	Handsfree
HSET	Headset
LAN	LAN Access Using PPP
NAP	Network Access Point
OPUSH	OBEX Object Push
PANU	Personal Area Networking User
SP	Serial Port

EXIT STATUS

The **sdpquery** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

bluetooth(3), sdp(3), btconfig(8), sdpd(8)

HISTORY

The **sdpquery** command first appeared in FreeBSD 5.3 as **sdpcontrol**. It was ported to NetBSD 4.0 under its present name by Iain Hibbert under the sponsorship of Itronix, Inc.

AUTHORS

Maksim Yevmenkin ⟨m_evmenkin@yahoo.com⟩
Iain Hibbert for Itronix, Inc.

CAVEATS

The **sdpquery** utility only requests the following attributes from the SDP server:

1. Service Record Handle
2. Service Class ID List
3. Protocol Descriptor List
4. Bluetooth Profile Descriptor List

NAME

sed — stream editor

SYNOPSIS

sed [**-aEn**] *command* [*file ...*]
sed [**-aEn**] [**-e** *command*] [**-f** *command_file*] [*file ...*]

DESCRIPTION

The **sed** utility reads the specified files, or the standard input if no files are specified, modifying the input as specified by a list of commands. The input is then written to the standard output.

A single command may be specified as the first argument to **sed**. Multiple commands may be specified by using the **-e** or **-f** options. All commands are applied to the input in the order they are specified regardless of their origin.

The following options are available:

-a The files listed as parameters for the "w" functions are created (or truncated) before any processing begins, by default. The **-a** option causes **sed** to delay opening each file until a command containing the related "w" function is applied to a line of input.

-E Enables the use of extended regular expressions instead of the usual basic regular expression syntax.

-e *command*
Append the editing commands specified by the *command* argument to the list of commands.

-f *command_file*
Append the editing commands found in the file *command_file* to the list of commands. The editing commands should each be listed on a separate line.

-n By default, each line of input is echoed to the standard output after all of the commands have been applied to it. The **-n** option suppresses this behavior.

The form of a **sed** command is as follows:

```
[address[,address]]function[arguments]
```

Whitespace may be inserted before the first address and the function portions of the command.

Normally, **sed** cyclically copies a line of input, not including its terminating newline character, into a *pattern space*, (unless there is something left after a "D" function), applies all of the commands with addresses that select that pattern space, copies the pattern space to the standard output, appending a newline, and deletes the pattern space.

Some of the functions use a *hold space* to save all or part of the pattern space for subsequent retrieval.

SED ADDRESSES

An address is not required, but if specified must be a number (that counts input lines cumulatively across input files), a dollar ("$") character that addresses the last line of input, or a context address (which consists of a regular expression preceded and followed by a delimiter).

A command line with no addresses selects every pattern space.

A command line with one address selects all of the pattern spaces that match the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less

than or equal to the line number first selected, only that line is selected.) Starting at the first line following the selected range, **sed** starts looking again for the first address.

Editing commands can be applied to non-selected pattern spaces by use of the exclamation character ("!") function.

SED REGULAR EXPRESSIONS

The **sed** regular expressions are basic regular expressions (BRE's, see `re_format`(7) for more information). In addition, **sed** has the following two additions to BRE's:

1. In a context address, any character other than a backslash ("\") or newline character may be used to delimit the regular expression by prefixing the first use of that delimiter with a backslash. Also, putting a backslash character before the delimiting character causes the character to be treated literally. For example, in the context address \xabc\xdefx, the RE delimiter is an "x" and the second "x" stands for itself, so that the regular expression is "abcxdef".

2. The escape sequence \n matches a newline character embedded in the pattern space. You can't, however, use a literal newline character in an address or in the substitute command.

One special feature of **sed** regular expressions is that they can default to the last regular expression used. If a regular expression is empty, i.e. just the delimiter characters are specified, the last regular expression encountered is used instead. The last regular expression is defined as the last regular expression used as part of an address or substitute command, and at run-time, not compile-time. For example, the command "/abc/s//XXX/" will substitute "XXX" for the pattern "abc".

SED FUNCTIONS

In the following list of commands, the maximum number of permissible addresses for each command is indicated by [0addr], [1addr], or [2addr], representing zero, one, or two addresses.

The argument *text* consists of one or more lines. To embed a newline in the text, precede it with a backslash. Other backslashes in text are deleted and the following character taken literally.

The "r" and "w" functions take an optional file parameter, which should be separated from the function letter by white space. Each file given as an argument to **sed** is created (or its contents truncated) before any input processing begins.

The "b", "r", "s", "t", "w", "y", "!", and ":" functions all accept additional arguments. The following synopses indicate which arguments have to be separated from the function letters by white space characters.

Two of the functions take a function-list. This is a list of **sed** functions separated by newlines, as follows:

```
{ function
  function
  ...
  function
}
```

The "{" can be preceded by white space and can be followed by white space. The function can be preceded by white space. The terminating "}" must be preceded by a newline (and optionally white space).

[2addr] function-list
: Execute function-list only when the pattern space is selected.

[1addr]a\
text

Write *text* to standard output immediately before each attempt to read a line of input, whether by executing the “N” function or by beginning a new cycle.

[2addr]b[label]

Branch to the “:” function with the specified label. If the label is not specified, branch to the end of the script.

[2addr]c\
text

Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, *text* is written to the standard output.

[2addr]d Delete the pattern space and start the next cycle.

[2addr]D

Delete the initial segment of the pattern space through the first newline character and start the next cycle.

[2addr]g Replace the contents of the pattern space with the contents of the hold space.

[2addr]G

Append a newline character followed by the contents of the hold space to the pattern space.

[2addr]h Replace the contents of the hold space with the contents of the pattern space.

[2addr]H

Append a newline character followed by the contents of the pattern space to the hold space.

[1addr]i\
text

Write *text* to the standard output.

[2addr]l (The letter ell.) Write the pattern space to the standard output in a visually unambiguous form. This form is as follows:

backslash	\\
alert	\a
form-feed	\f
newline	\n
carriage-return	\r
tab	\t
vertical tab	\v

Nonprintable characters are written as three-digit octal numbers (with a preceding backslash) for each byte in the character (most significant byte first). Long lines are folded, with the point of folding indicated by displaying a backslash followed by a newline. The end of each line is marked with a “$”.

[2addr]n Write the pattern space to the standard output if the default output has not been suppressed, and replace the pattern space with the next line of input. (Does not begin a new cycle.)

[2addr]N
Append the next line of input to the pattern space, using an embedded newline character to separate the appended material from the original contents. Note that the current line number changes.

[2addr]p Write the pattern space to standard output.

[2addr]P Write the pattern space, up to the first newline character to the standard output.

[1addr]q Branch to the end of the script and quit without starting a new cycle.

[1addr]r file
Copy the contents of *file* to the standard output immediately before the next attempt to read a line of input. If *file* cannot be read for any reason, it is silently ignored and no error condition is set.

[2addr]s/regular expression/replacement/flags
Substitute the replacement string for the first instance of the regular expression in the pattern space. Any character other than backslash or newline can be used instead of a slash to delimit the RE and the replacement. Within the RE and the replacement, the RE delimiter itself can be used as a literal character if it is preceded by a backslash.

An ampersand (“&”) appearing in the replacement is replaced by the string matching the RE. The special meaning of “&” in this context can be suppressed by preceding it by a backslash. The string “\#”, where “#” is a digit, is replaced by the text matched by the corresponding backreference expression (see `re_format`(7)).

A line can be split by substituting a newline character into it. To specify a newline character in the replacement string, precede it with a backslash.

The value of *flags* in the substitute function is zero or more of the following:

0 ... 9 Make the substitution only for the N’th occurrence of the regular expression in the pattern space.

g Make the substitution for all non-overlapping matches of the regular expression, not just the first one.

p Write the pattern space to standard output if a replacement was made. If the replacement string is identical to that which it replaces, it is still considered to have been a replacement.

w *file* Append the pattern space to *file* if a replacement was made. If the replacement string is identical to that which it replaces, it is still considered to have been a replacement.

[2addr]t [label]
Branch to the “:” function bearing the label if any substitutions have been made since the most recent reading of an input line or execution of a “t” function. If no label is specified, branch to the end of the script.

[2addr]w *file*
Append the pattern space to the *file*.

[2addr]x Swap the contents of the pattern and hold spaces.

[2addr]y/string1/string2/
Replace all occurrences of characters in *string1* in the pattern space with the corresponding characters from *string2*. Any character other than a backslash or newline can be used instead of a slash to delimit the strings. Within *string1* and *string2*, a backslash followed by any character other than a newline is that literal character, and a backslash followed by an ‘‘n’’ is replaced by a newline character.

[2addr]!function
[2addr]!function-list
Apply the function or function-list only to the lines that are *not* selected by the address(es).

[0addr]:label
This function does nothing; it bears a label to which the “b” and “t” commands may branch.

[1addr]= Write the line number to the standard output followed by a newline character.

[0addr] Empty lines are ignored.

[0addr]# The “#” and the remainder of the line are ignored (treated as a comment), with the single exception that if the first two characters in the file are “#n”, the default output is suppressed. This is the same as specifying the **-n** option on the command line.

The **sed** utility exits 0 on success and >0 if an error occurs.

SEE ALSO

awk(1), ed(1), grep(1), regex(3), re_format(7)

STANDARDS

The **sed** function is expected to be a superset of the IEEE Std 1003.2 (“POSIX.2”) specification.

HISTORY

A **sed** command appeared in Version 7 AT&T UNIX.

NAME

sendmail − Postfix to Sendmail compatibility interface

SYNOPSIS

sendmail [*option ...*] [*recipient ...*]

mailq
sendmail -bp

newaliases
sendmail -I

DESCRIPTION

The Postfix **sendmail**(1) command implements the Postfix to Sendmail compatibility interface. For the sake of compatibility with existing applications, some Sendmail command-line options are recognized but silently ignored.

By default, Postfix **sendmail**(1) reads a message from standard input until EOF or until it reads a line with only a **.** character, and arranges for delivery. Postfix **sendmail**(1) relies on the **postdrop**(1) command to create a queue file in the **maildrop** directory.

Specific command aliases are provided for other common modes of operation:

mailq List the mail queue. Each entry shows the queue file ID, message size, arrival time, sender, and the recipients that still need to be delivered. If mail could not be delivered upon the last attempt, the reason for failure is shown. This mode of operation is implemented by executing the **postqueue**(1) command.

newaliases
Initialize the alias database. If no input file is specified (with the **-oA** option, see below), the program processes the file(s) specified with the **alias_database** configuration parameter. If no alias database type is specified, the program uses the type specified with the **default_database_type** configuration parameter. This mode of operation is implemented by running the **postalias**(1) command.

Note: it may take a minute or so before an alias database update becomes visible. Use the "**postfix reload**" command to eliminate this delay.

These and other features can be selected by specifying the appropriate combination of command-line options. Some features are controlled by parameters in the **main.cf** configuration file.

The following options are recognized:

-Am (ignored)

-Ac (ignored)
Postfix sendmail uses the same configuration file regardless of whether or not a message is an initial submission.

-B *body_type*
The message body MIME type: **7BIT** or **8BITMIME**.

-bd Go into daemon mode. This mode of operation is implemented by executing the "**postfix start**" command.

-bh (ignored)

-bH (ignored)
Postfix has no persistent host status database.

-bi Initialize alias database. See the **newaliases** command above.

-bm Read mail from standard input and arrange for delivery. This is the default mode of operation.

-bp List the mail queue. See the **mailq** command above.

-bs Stand-alone SMTP server mode. Read SMTP commands from standard input, and write responses to standard output. In stand-alone SMTP server mode, mail relaying and other access controls are disabled by default. To enable them, run the process as the **mail_owner** user.

This mode of operation is implemented by running the **smtpd**(8) daemon.

-bv Do not collect or deliver a message. Instead, send an email report after verifying each recipient address. This is useful for testing address rewriting and routing configurations.

This feature is available in Postfix version 2.1 and later.

-C *config_file*

-C *config_dir*
The path name of the Postfix **main.cf** file, or of its parent directory. This information is ignored with Postfix versions before 2.3.

With all Postfix versions, you can specify a directory pathname with the MAIL_CONFIG environment variable to override the location of configuration files.

-F *full_name*
Set the sender full name. This overrides the NAME environment variable, and is used only with messages that have no **From:** message header.

-f *sender*
Set the envelope sender address. This is the address where delivery problems are sent to. With Postfix versions before 2.1, the **Errors-To:** message header overrides the error return address.

-G Gateway (relay) submission, as opposed to initial user submission. Either do not rewrite addresses at all, or update incomplete addresses with the domain information specified with **remote_header_rewrite_domain**.

This option is ignored before Postfix version 2.3.

-h *hop_count* (ignored)
Hop count limit. Use the **hopcount_limit** configuration parameter instead.

-I Initialize alias database. See the **newaliases** command above.

-i When reading a message from standard input, don´t treat a line with only a **.** character as the end of input.

-L *label* (ignored)
The logging label. Use the **syslog_name** configuration parameter instead.

-m (ignored)
Backwards compatibility.

-N *dsn* (default: ’delay, failure’)
Delivery status notification control. Specify either a comma-separated list with one or more of **failure** (send notification when delivery fails), **delay** (send notification when delivery is delayed), or **success** (send notification when the message is delivered); or specify **never** (don’t send any notifications at all).

This feature is available in Postfix 2.3 and later.

-n (ignored)
Backwards compatibility.

-oA*alias_database*
Non-default alias database. Specify *pathname* or *type*:*pathname*. See **postalias**(1) for details.

-O *option*=*value* (ignored)
Backwards compatibility.

-o7 (ignored)

-o8 (ignored)
To send 8-bit or binary content, use an appropriate MIME encapsulation and specify the appropriate **-B** command-line option.

-oi When reading a message from standard input, don´t treat a line with only a **.** character as the end of input.

-om (ignored)
The sender is never eliminated from alias etc. expansions.

-o *x value* (ignored)
Set option *x* to *value*. Use the equivalent configuration parameter in **main.cf** instead.

-r *sender*
Set the envelope sender address. This is the address where delivery problems are sent to. With Postfix versions before 2.1, the **Errors-To:** message header overrides the error return address.

-R *return_limit* (ignored)
Limit the size of bounced mail. Use the **bounce_size_limit** configuration parameter instead.

-q Attempt to deliver all queued mail. This is implemented by executing the **postqueue**(1) command.

Warning: flushing undeliverable mail frequently will result in poor delivery performance of all other mail.

-q*interval* (ignored)
The interval between queue runs. Use the **queue_run_delay** configuration parameter instead.

-qI*queueid*
Schedule immediate delivery of mail with the specified queue ID. This option is implemented by executing the **postqueue**(1) command, and is available with Postfix version 2.4 and later.

-qR*site* Schedule immediate delivery of all mail that is queued for the named *site*. This option accepts only *site* names that are eligible for the "fast flush" service, and is implemented by executing the **postqueue**(1) command. See **flush**(8) for more information about the "fast flush" service.

-qS*site* This command is not implemented. Use the slower "**sendmail -q**" command instead.

-t Extract recipients from message headers. These are added to any recipients specified on the command line.

With Postfix versions prior to 2.1, this option requires that no recipient addresses are specified on the command line.

-U (ignored)
Initial user submission.

-V *envid*
Specify the envelope ID for notification by servers that support DSN.

This feature is available in Postfix 2.3 and later.

-XV (Postfix 2.2 and earlier: **-V**)
Variable Envelope Return Path. Given an envelope sender address of the form *owner-listname@origin*, each recipient *user@domain* receives mail with a personalized envelope sender address.

By default, the personalized envelope sender address is *owner-listname+user=domain@origin*. The default + and = characters are configurable with the **default_verp_delimiters** configuration parameter.

-XV*xy* (Postfix 2.2 and earlier: **-V***xy*)
As **-XV**, but uses *x* and *y* as the VERP delimiter characters, instead of the characters specified with the **default_verp_delimiters** configuration parameter.

-v Send an email report of the first delivery attempt (Postfix versions 2.1 and later). Mail delivery always happens in the background. When multiple **-v** options are given, enable verbose logging for debugging purposes.

-X *log_file* (ignored)
Log mailer traffic. Use the **debug_peer_list** and **debug_peer_level** configuration parameters instead.

SECURITY

By design, this program is not set-user (or group) id. However, it must handle data from untrusted, possibly remote, users. Thus, the usual precautions need to be taken against malicious inputs.

DIAGNOSTICS

Problems are logged to **syslogd**(8) and to the standard error stream.

ENVIRONMENT

MAIL_CONFIG
Directory with Postfix configuration files.

MAIL_VERBOSE (value does not matter)
Enable verbose logging for debugging purposes.

MAIL_DEBUG (value does not matter)
Enable debugging with an external command, as specified with the **debugger_command** configuration parameter.

NAME The sender full name. This is used only with messages that have no **From:** message header. See also the **-F** option above.

CONFIGURATION PARAMETERS

The following **main.cf** parameters are especially relevant to this program. The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

TROUBLE SHOOTING CONTROLS

The DEBUG_README file gives examples of how to trouble shoot a Postfix system.

debugger_command (empty)
The external command to execute when a Postfix daemon program is invoked with the -D option.

debug_peer_level (2)
The increment in verbose logging level when a remote client or server matches a pattern in the debug_peer_list parameter.

debug_peer_list (empty)
Optional list of remote client or server hostname or network address patterns that cause the verbose logging level to increase by the amount specified in $debug_peer_level.

ACCESS CONTROLS

Available in Postfix version 2.2 and later:

authorized_flush_users (static:anyone)
List of users who are authorized to flush the queue.

authorized_mailq_users (static:anyone)
List of users who are authorized to view the queue.

authorized_submit_users (static:anyone)
List of users who are authorized to submit mail with the **sendmail**(1) command (and with the privileged **postdrop**(1) helper command).

RESOURCE AND RATE CONTROLS

bounce_size_limit (50000)
The maximal amount of original message text that is sent in a non-delivery notification.

fork_attempts (5)
The maximal number of attempts to fork() a child process.

fork_delay (1s)
The delay between attempts to fork() a child process.

hopcount_limit (50)
The maximal number of Received: message headers that is allowed in the primary message headers.

queue_run_delay (version dependent)
The time between deferred queue scans by the queue manager.

FAST FLUSH CONTROLS

The ETRN_README file describes configuration and operation details for the Postfix "fast flush" service.

fast_flush_domains ($relay_domains)
Optional list of destinations that are eligible for per-destination logfiles with mail that is queued to those destinations.

VERP CONTROLS

The VERP_README file describes configuration and operation details of Postfix support for variable envelope return path addresses.

default_verp_delimiters (+=)
The two default VERP delimiter characters.

verp_delimiter_filter (-=+)
The characters Postfix accepts as VERP delimiter characters on the Postfix **sendmail**(1) command line and in SMTP commands.

MISCELLANEOUS CONTROLS

alias_database (see 'postconf -d' output)
The alias databases for **local**(8) delivery that are updated with "**newaliases**" or with "**sendmail -bi**".

command_directory (see 'postconf -d' output)
The location of all postfix administrative commands.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_directory (see 'postconf -d' output)
The directory with Postfix support programs and daemon programs.

default_database_type (see 'postconf -d' output)
The default database type for use in **newaliases**(1), **postalias**(1) and **postmap**(1) commands.

delay_warning_time (0h)
The time after which the sender receives the message headers of mail that is still queued.

enable_errors_to (no)
Report mail delivery errors to the address specified with the non-standard Errors-To: message header, instead of the envelope sender address (this feature is removed with Postfix version 2.2, is turned off by default with Postfix version 2.1, and is always turned on with older Postfix versions).

mail_owner (postfix)
The UNIX system account that owns the Postfix queue and most Postfix daemon processes.

queue_directory (see 'postconf -d' output)
The location of the Postfix top-level queue directory.

remote_header_rewrite_domain (empty)
Don't rewrite message headers from remote clients at all when this parameter is empty; otherwise, rewrite message headers and append the specified domain name to incomplete addresses.

syslog_facility (mail)
The syslog facility of Postfix logging.

syslog_name (postfix)
The mail system name that is prepended to the process name in syslog records, so that "smtpd" becomes, for example, "postfix/smtpd".

FILES

/var/spool/postfix, mail queue
/etc/postfix, configuration files

SEE ALSO

pickup(8), mail pickup daemon
qmgr(8), queue manager
smtpd(8), SMTP server
flush(8), fast flush service
postsuper(1), queue maintenance
postalias(1), create/update/query alias database
postdrop(1), mail posting utility
postfix(1), mail system control
postqueue(1), mail queue control
syslogd(8), system logging

README_FILES

Use "**postconf readme_directory**" or
"**postconf html_directory**" to locate this information.
DEBUG_README, Postfix debugging howto
ETRN_README, Postfix ETRN howto
VERP_README, Postfix VERP howto

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

seq — print sequences of numbers

SYNOPSIS

seq [**-w**] [**-f** *format*] [**-s** *string*] [**-t** *string*] [*first* [*incr*]] *last*

DESCRIPTION

The **seq** utility prints a sequence of numbers, one per line (default), from *first* (default 1), to near *last* as possible, in increments of *incr* (default 1). When *first* is larger than *last* the default *incr* is -1.

All numbers are interpreted as floating point.

Normally integer values are printed as decimal integers.

The **seq** utility accepts the following options:

-f *format* Use a `printf`(3) style *format* to print each number. Only the **E**, **e**, **f**, **G**, **g**, and **%** conversion characters are valid, along with any optional flags and an optional numeric mimimum field width or precision. The *format* can contain character escape sequences in backslash notation as defined in ANSI X3.159-1989 ("ANSI C89"). The default is **%g**.

-s *string* Use *string* to separate numbers. The *string* can contain character escape sequences in backslash notation as defined in ANSI X3.159-1989 ("ANSI C89"). The default is **\n**.

-t *string* Use *string* to terminate sequence of numbers. The *string* can contain character escape sequences in backslash notation as defined in ANSI X3.159-1989 ("ANSI C89"). This option is useful when the default separator does not contain a **\n**.

-w Equalize the widths of all numbers by padding with zeros as necessary. This option has no effect with the **-f** option. If any sequence numbers will be printed in exponential notation, the default conversion is changed to **%e**.

The **seq** utility exits 0 on success and non-zero if an error occurs.

EXAMPLES

```
# seq 1 3
1
2
3

# seq 3 1
3
2
1

# seq -w 0 .05 .1
0.00
0.05
0.10
```

SEE ALSO

`jot`(1), `printf`(1), `printf`(3)

HISTORY

The **seq** command first appeared in Plan 9 from Bell Labs. A **seq** command appeared in NetBSD 3.0. This command was based on the command of the same name in Plan 9 from Bell Labs and the GNU core utilities. The GNU **seq** command first appeared in the 1.13 shell utilities release.

BUGS

The **-w** option does not handle the transition from pure floating point to exponent representation very well. The **seq** command is not bug for bug compatible with the Plan 9 from Bell Labs or GNU versions of **seq**.

NAME

sftp — secure file transfer program

SYNOPSIS

sftp [**-1Cv**] [**-B** *buffer_size*] [**-b** *batchfile*] [**-F** *ssh_config*] [**-o** *ssh_option*]
[**-P** *sftp_server_path*] [**-R** *num_requests*] [**-S** *program*]
[**-s** *subsystem* | *sftp_server*] *host*
sftp [[*user*@]*host*[:*file* [*file*]]]
sftp [[*user*@]*host*[:*dir*[/]]]
sftp -b *batchfile* [*user*@]*host*

DESCRIPTION

sftp is an interactive file transfer program, similar to ftp(1), which performs all operations over an encrypted ssh(1) transport. It may also use many features of ssh, such as public key authentication and compression. **sftp** connects and logs into the specified *host*, then enters an interactive command mode.

The second usage format will retrieve files automatically if a non-interactive authentication method is used; otherwise it will do so after successful interactive authentication.

The third usage format allows **sftp** to start in a remote directory.

The final usage format allows for automated sessions using the **-b** option. In such cases, it is necessary to configure non-interactive authentication to obviate the need to enter a password at connection time (see sshd(8) and ssh-keygen(1) for details). The options are as follows:

-1 Specify the use of protocol version 1.

-B *buffer_size*
Specify the size of the buffer that **sftp** uses when transferring files. Larger buffers require fewer round trips at the cost of higher memory consumption. The default is 32768 bytes.

-b *batchfile*
Batch mode reads a series of commands from an input *batchfile* instead of *stdin*. Since it lacks user interaction it should be used in conjunction with non-interactive authentication. A *batchfile* of '−' may be used to indicate standard input. **sftp** will abort if any of the following commands fail: **get**, **put**, **rename**, **ln**, **rm**, **mkdir**, **chdir**, **ls**, **lchdir**, **chmod**, **chown**, **chgrp**, **lpwd** and **lmkdir**. Termination on error can be suppressed on a command by command basis by prefixing the command with a '−' character (for example, **-rm /tmp/blah***).

-C Enables compression (via ssh's **-C** flag).

-F *ssh_config*
Specifies an alternative per-user configuration file for ssh(1). This option is directly passed to ssh(1).

-o *ssh_option*
Can be used to pass options to **ssh** in the format used in ssh_config(5). This is useful for specifying options for which there is no separate **sftp** command-line flag. For example, to specify an alternate port use: **sftp -oPort=24**. For full details of the options listed below, and their possible values, see ssh_config(5).

AddressFamily
BatchMode
BindAddress

ChallengeResponseAuthentication
CheckHostIP
Cipher
Ciphers
Compression
CompressionLevel
ConnectionAttempts
ConnectTimeout
ControlMaster
ControlPath
GlobalKnownHostsFile
GSSAPIAuthentication
GSSAPIDelegateCredentials
HashKnownHosts
Host
HostbasedAuthentication
HostKeyAlgorithms
HostKeyAlias
HostName
IdentityFile
IdentitiesOnly
KbdInteractiveDevices
LogLevel
MACs
NoHostAuthenticationForLocalhost
NumberOfPasswordPrompts
PasswordAuthentication
Port
PreferredAuthentications
Protocol
ProxyCommand
PubkeyAuthentication
RekeyLimit
RhostsRSAAuthentication
RSAAuthentication
SendEnv
ServerAliveInterval
ServerAliveCountMax
SmartcardDevice
StrictHostKeyChecking
TCPKeepAlive
UsePrivilegedPort
User
UserKnownHostsFile
VerifyHostKeyDNS

-P *sftp_server_path*
Connect directly to a local sftp server (rather than via ssh(1)). This option may be useful in debugging the client and server.

-R *num_requests*
Specify how many requests may be outstanding at any one time. Increasing this may slightly improve file transfer speed but will increase memory usage. The default is 16 outstanding requests.

-S *program*
Name of the *program* to use for the encrypted connection. The program must understand ssh(1) options.

-s *subsystem* | *sftp_server*
Specifies the SSH2 subsystem or the path for an sftp server on the remote host. A path is useful for using **sftp** over protocol version 1, or when the remote sshd(8) does not have an sftp subsystem configured.

-v Raise logging level. This option is also passed to ssh.

INTERACTIVE COMMANDS

Once in interactive mode, **sftp** understands a set of commands similar to those of ftp(1). Commands are case insensitive. Pathnames that contain spaces must be enclosed in quotes. Any special characters contained within pathnames that are recognized by glob(3) must be escaped with backslashes ('\').

bye Quit **sftp**.

cd *path*
Change remote directory to *path*.

chgrp *grp path*
Change group of file *path* to *grp*. *path* may contain glob(3) characters and may match multiple files. *grp* must be a numeric GID.

chmod *mode path*
Change permissions of file *path* to *mode*. *path* may contain glob(3) characters and may match multiple files.

chown *own path*
Change owner of file *path* to *own*. *path* may contain glob(3) characters and may match multiple files. *own* must be a numeric UID.

exit Quit **sftp**.

get [**-P**] *remote-path* [*local-path*]
Retrieve the *remote-file* and store it on the local machine. If the local path name is not specified, it is given the same name it has on the remote machine. *remote-path* may contain glob(3) characters and may match multiple files. If it does and *local-path* is specified, then *local-path* must specify a directory. If the **-P** flag is specified, then full file permissions and access times are copied too.

help Display help text.

lcd *path*
Change local directory to *path*.

lls [*ls-options* [*path*]]
Display local directory listing of either *path* or current directory if *path* is not specified. *ls-options* may contain any flags supported by the local system's ls(1) command. *path* may contain glob(3) characters and may match multiple files.

lmkdir *path*
Create local directory specified by *path*.

ln *oldpath newpath*
Create a symbolic link from *oldpath* to *newpath*.

lpwd Print local working directory.

ls [**-1aflnrSt**] [*path*]
Display a remote directory listing of either *path* or the current directory if *path* is not specified. *path* may contain glob(3) characters and may match multiple files.

The following flags are recognized and alter the behaviour of **ls** accordingly:

-1 Produce single columnar output.

-a List files beginning with a dot ('.').

-f Do not sort the listing. The default sort order is lexicographical.

-l Display additional details including permissions and ownership information.

-n Produce a long listing with user and group information presented numerically.

-r Reverse the sort order of the listing.

-S Sort the listing by file size.

-t Sort the listing by last modification time.

lumask *umask*
Set local umask to *umask*.

mkdir *path*
Create remote directory specified by *path*.

progress
Toggle display of progress meter.

put [**-P**] *local-path* [*remote-path*]
Upload *local-path* and store it on the remote machine. If the remote path name is not specified, it is given the same name it has on the local machine. *local-path* may contain glob(3) characters and may match multiple files. If it does and *remote-path* is specified, then *remote-path* must specify a directory. If the **-P** flag is specified, then the file's full permission and access time are copied too.

pwd Display remote working directory.

quit Quit **sftp**.

rename *oldpath newpath*
Rename remote file from *oldpath* to *newpath*.

rm *path*
Delete remote file specified by *path*.

rmdir *path*
Remove remote directory specified by *path*.

symlink *oldpath newpath*
Create a symbolic link from *oldpath* to *newpath*.

version
Display the **sftp** protocol version.

! *command*
Execute *command* in local shell.

! Escape to local shell.

? Synonym for help.

SEE ALSO

`ftp`(1), `ls`(1), `scp`(1), `ssh`(1), `ssh-add`(1), `ssh-keygen`(1), `glob`(3), `ssh_config`(5), `sftp-server`(8), `sshd`(8)

T. Ylonen and S. Lehtinen, *SSH File Transfer Protocol*, draft-ietf-secsh-filexfer-00.txt, January 2001, work in progress material.

NAME

sh — command interpreter (shell)

SYNOPSIS

sh [**−aCefnuvxIimqVEb**] [**+aCefnuvxIimqVEb**] [**−o** *option_name*] [**+o** *option_name*]
[*command_file* [*argument ...*]]
sh −c [**−aCefnuvxIimqVEb**] [**+aCefnuvxIimqVEb**] [**−o** *option_name*]
[**+o** *option_name*] *command_string* [*command_name* [*argument ...*]]
sh −s [**−aCefnuvxIimqVEb**] [**+aCefnuvxIimqVEb**] [**−o** *option_name*]
[**+o** *option_name*] [*argument ...*]

DESCRIPTION

sh is the standard command interpreter for the system. The current version of **sh** is in the process of being changed to conform with the POSIX 1003.2 and 1003.2a specifications for the shell. This version has many features which make it appear similar in some respects to the Korn shell, but it is not a Korn shell clone (see ksh(1)). Only features designated by POSIX, plus a few Berkeley extensions, are being incorporated into this shell. This man page is not intended to be a tutorial or a complete specification of the shell.

Overview

The shell is a command that reads lines from either a file or the terminal, interprets them, and generally executes other commands. It is the program that is running when a user logs into the system (although a user can select a different shell with the chsh(1) command). The shell implements a language that has flow control constructs, a macro facility that provides a variety of features in addition to data storage, along with built in history and line editing capabilities. It incorporates many features to aid interactive use and has the advantage that the interpretative language is common to both interactive and non-interactive use (shell scripts). That is, commands can be typed directly to the running shell or can be put into a file and the file can be executed directly by the shell.

Invocation

If no arguments are present and if the standard input of the shell is connected to a terminal (or if the **−i** flag is set), and the **−c** option is not present, the shell is considered an interactive shell. An interactive shell generally prompts before each command and handles programming and command errors differently (as described below). When first starting, the shell inspects argument 0, and if it begins with a dash '-', the shell is also considered a login shell. This is normally done automatically by the system when the user first logs in. A login shell first reads commands from the files /etc/profile and .profile if they exist. If the environment variable ENV is set on entry to a shell, or is set in the .profile of a login shell, the shell next reads commands from the file named in ENV. Therefore, a user should place commands that are to be executed only at login time in the .profile file, and commands that are executed for every shell inside the ENV file. To set the ENV variable to some file, place the following line in your .profile of your home directory

```
ENV=$HOME/.shinit; export ENV
```

substituting for ".shinit" any filename you wish. Since the ENV file is read for every invocation of the shell, including shell scripts and non-interactive shells, the following paradigm is useful for restricting commands in the ENV file to interactive invocations. Place commands within the "case" and "esac" below (these commands are described later):

```
case $- in *i*)
     # commands for interactive use only
     ...
```

```
esac
```

If command line arguments besides the options have been specified, then the shell treats the first argument as the name of a file from which to read commands (a shell script), and the remaining arguments are set as the positional parameters of the shell ($1, $2, etc). Otherwise, the shell reads commands from its standard input.

Argument List Processing

All of the single letter options have a corresponding name that can be used as an argument to the **-o** option. The set **-o** name is provided next to the single letter option in the description below. Specifying a dash "-" turns the option on, while using a plus "+" disables the option. The following options can be set from the command line or with the **set** built-in (described later).

-a *allexport*	Export all variables assigned to.
-c	Read commands from the *command_string* operand instead of from the standard input. Special parameter 0 will be set from the *command_name* operand and the positional parameters ($1, $2, etc.) set from the remaining argument operands.
-C *noclobber*	Don't overwrite existing files with ">".
-e *errexit*	If not interactive, exit immediately if any untested command fails. The exit status of a command is considered to be explicitly tested if the command is used to control an **if**, **elif**, **while**, or **until**; or if the command is the left hand operand of an "&&" or "||" operator.
-f *noglob*	Disable pathname expansion.
-n *noexec*	If not interactive, read commands but do not execute them. This is useful for checking the syntax of shell scripts.
-u *nounset*	Write a message to standard error when attempting to expand a variable that is not set, and if the shell is not interactive, exit immediately.
-v *verbose*	The shell writes its input to standard error as it is read. Useful for debugging.
-x *xtrace*	Write each command to standard error (preceded by a '+ ') before it is executed. Useful for debugging.
-q *quietprofile*	If the **-v** or **-x** options have been set, do not apply them when reading initialization files, these being /etc/profile, .profile, and the file specified by the ENV environment variable.
-I *ignoreeof*	Ignore EOFs from input when interactive.
-i *interactive*	Force the shell to behave interactively.
-m *monitor*	Turn on job control (set automatically when interactive).
-s *stdin*	Read commands from standard input (set automatically if no file arguments are present). This option has no effect when set after the shell has already started running (i.e. with **set**).
-V *vi*	Enable the built-in vi(1) command line editor (disables **-E** if it has been set). (See the **Command Line Editing** section below.)
-E *emacs*	Enable the built-in emacs style command line editor (disables **-V** if it has been set). (See the **Command Line Editing** section below.)

-b *notify*	Enable asynchronous notification of background job completion. (UNIMPLEMENTED for 4.4alpha)
cdprint	Make an interactive shell always print the new directory name when changed by the **cd** command.
tabcomplete	Enables filename completion in the command line editor. Typing a tab character will extend the current input word to match a filename. If more than one filename matches it is only extended to be the common prefix. Typing a second tab character will list all the matching names.

Lexical Structure

The shell reads input in terms of lines from a file and breaks it up into words at whitespace (blanks and tabs), and at certain sequences of characters that are special to the shell called "operators". There are two types of operators: control operators and redirection operators (their meaning is discussed later). Following is a list of operators:

Control operators:

```
& && ( ) ; ;; | || <newline>
```

Redirection operators:

```
< > >| << >> <& >& <<- <>
```

Quoting

Quoting is used to remove the special meaning of certain characters or words to the shell, such as operators, whitespace, or keywords. There are three types of quoting: matched single quotes, matched double quotes, and backslash.

Backslash

A backslash preserves the literal meaning of the following character, with the exception of ⟨newline⟩. A backslash preceding a ⟨newline⟩ is treated as a line continuation.

Single Quotes

Enclosing characters in single quotes preserves the literal meaning of all the characters (except single quotes, making it impossible to put single-quotes in a single-quoted string).

Double Quotes

Enclosing characters within double quotes preserves the literal meaning of all characters except dollar sign (`$`), backquote (`‘`), and backslash (`\`). The backslash inside double quotes is historically weird, and serves to quote only the following characters:

```
$ ‘ " \ <newline>.
```

Otherwise it remains literal.

Reserved Words

Reserved words are words that have special meaning to the shell and are recognized at the beginning of a line and after a control operator. The following are reserved words:

!	elif	fi	while	case
else	for	then	{	}
do	done	until	if	esac

Their meaning is discussed later.

Aliases

An alias is a name and corresponding value set using the **alias** built-in command. Whenever a reserved word may occur (see above), and after checking for reserved words, the shell checks the word to see if it matches an alias. If it does, it replaces it in the input stream with its value. For example, if there is an alias called "lf" with the value "ls -F", then the input:

`lf foobar` ⟨return⟩

would become

`ls -F foobar` ⟨return⟩

Aliases provide a convenient way for naive users to create shorthands for commands without having to learn how to create functions with arguments. They can also be used to create lexically obscure code. This use is discouraged.

Commands

The shell interprets the words it reads according to a language, the specification of which is outside the scope of this man page (refer to the BNF in the POSIX 1003.2 document). Essentially though, a line is read and if the first word of the line (or after a control operator) is not a reserved word, then the shell has recognized a simple command. Otherwise, a complex command or some other special construct may have been recognized.

Simple Commands

If a simple command has been recognized, the shell performs the following actions:

1. Leading words of the form "name=value" are stripped off and assigned to the environment of the simple command. Redirection operators and their arguments (as described below) are stripped off and saved for processing.
2. The remaining words are expanded as described in the section called "Expansions", and the first remaining word is considered the command name and the command is located. The remaining words are considered the arguments of the command. If no command name resulted, then the "name=value" variable assignments recognized in item 1 affect the current shell.
3. Redirections are performed as described in the next section.

Redirections

Redirections are used to change where a command reads its input or sends its output. In general, redirections open, close, or duplicate an existing reference to a file. The overall format used for redirection is:

`[n]` *redir-op* *`file`*

where *redir-op* is one of the redirection operators mentioned previously. Following is a list of the possible redirections. The `[n]` is an optional number, as in '3' (not '`[3]`'), that refers to a file descriptor.

[n]> file	Redirect standard output (or n) to file.
[n]>| file	Same, but override the **-C** option.
[n]>> file	Append standard output (or n) to file.
[n]< file	Redirect standard input (or n) from file.
[n1]<&n2	Duplicate standard input (or n1) from file descriptor n2.
[n]<&-	Close standard input (or n).

[n1]>&n2 Duplicate standard output (or n1) to n2.

[n]>&- Close standard output (or n).

[n]<> file Open file for reading and writing on standard input (or n).

The following redirection is often called a “here-document”.

```
[n]<< delimiter
     here-doc-text ...
delimiter
```

All the text on successive lines up to the delimiter is saved away and made available to the command on standard input, or file descriptor n if it is specified. If the delimiter as specified on the initial line is quoted, then the here-doc-text is treated literally, otherwise the text is subjected to parameter expansion, command substitution, and arithmetic expansion (as described in the section on “Expansions”). If the operator is “<<-” instead of “<<”, then leading tabs in the here-doc-text are stripped.

Search and Execution

There are three types of commands: shell functions, built-in commands, and normal programs -- and the command is searched for (by name) in that order. They each are executed in a different way.

When a shell function is executed, all of the shell positional parameters (except $0, which remains unchanged) are set to the arguments of the shell function. The variables which are explicitly placed in the environment of the command (by placing assignments to them before the function name) are made local to the function and are set to the values given. Then the command given in the function definition is executed. The positional parameters are restored to their original values when the command completes. This all occurs within the current shell.

Shell built-ins are executed internally to the shell, without spawning a new process.

Otherwise, if the command name doesn’t match a function or built-in, the command is searched for as a normal program in the file system (as described in the next section). When a normal program is executed, the shell runs the program, passing the arguments and the environment to the program. If the program is not a normal executable file (i.e., if it does not begin with the "magic number" whose ASCII representation is "#!", so `execve`(2) returns `ENOEXEC` then) the shell will interpret the program in a subshell. The child shell will reinitialize itself in this case, so that the effect will be as if a new shell had been invoked to handle the ad-hoc shell script, except that the location of hashed commands located in the parent shell will be remembered by the child.

Note that previous versions of this document and the source code itself misleadingly and sporadically refer to a shell script without a magic number as a "shell procedure".

Path Search

When locating a command, the shell first looks to see if it has a shell function by that name. Then it looks for a built-in command by that name. If a built-in command is not found, one of two things happen:

1. Command names containing a slash are simply executed without performing any searches.
2. The shell searches each entry in `PATH` in turn for the command. The value of the `PATH` variable should be a series of entries separated by colons. Each entry consists of a directory name. The current directory may be indicated implicitly by an empty directory name, or explicitly by a single period.

Command Exit Status

Each command has an exit status that can influence the behavior of other shell commands. The paradigm is that a command exits with zero for normal or success, and non-zero for failure, error, or a false indication. The man page for each command should indicate the various exit codes and what they mean. Additionally,

the built-in commands return exit codes, as does an executed shell function.

If a command consists entirely of variable assignments then the exit status of the command is that of the last command substitution if any, otherwise 0.

Complex Commands

Complex commands are combinations of simple commands with control operators or reserved words, together creating a larger complex command. More generally, a command is one of the following:

- simple command
- pipeline
- list or compound-list
- compound command
- function definition

Unless otherwise stated, the exit status of a command is that of the last simple command executed by the command.

Pipelines

A pipeline is a sequence of one or more commands separated by the control operator |. The standard output of all but the last command is connected to the standard input of the next command. The standard output of the last command is inherited from the shell, as usual.

The format for a pipeline is:

```
[!] command1 [| command2 ...]
```

The standard output of command1 is connected to the standard input of command2. The standard input, standard output, or both of a command is considered to be assigned by the pipeline before any redirection specified by redirection operators that are part of the command.

If the pipeline is not in the background (discussed later), the shell waits for all commands to complete.

If the reserved word ! does not precede the pipeline, the exit status is the exit status of the last command specified in the pipeline. Otherwise, the exit status is the logical NOT of the exit status of the last command. That is, if the last command returns zero, the exit status is 1; if the last command returns greater than zero, the exit status is zero.

Because pipeline assignment of standard input or standard output or both takes place before redirection, it can be modified by redirection. For example:

```
$ command1 2>&1 | command2
```

sends both the standard output and standard error of command1 to the standard input of command2.

A ; or ⟨newline⟩ terminator causes the preceding AND-OR-list (described next) to be executed sequentially; a & causes asynchronous execution of the preceding AND-OR-list.

Note that unlike some other shells, each process in the pipeline is a child of the invoking shell (unless it is a shell built-in, in which case it executes in the current shell -- but any effect it has on the environment is wiped).

Background Commands -- &

If a command is terminated by the control operator ampersand (&), the shell executes the command asynchronously -- that is, the shell does not wait for the command to finish before executing the next command.

The format for running a command in background is:

```
command1 & [command2 & ...]
```

If the shell is not interactive, the standard input of an asynchronous command is set to `/dev/null`.

Lists -- Generally Speaking

A list is a sequence of zero or more commands separated by newlines, semicolons, or ampersands, and optionally terminated by one of these three characters. The commands in a list are executed in the order they are written. If command is followed by an ampersand, the shell starts the command and immediately proceed onto the next command; otherwise it waits for the command to terminate before proceeding to the next one.

Short-Circuit List Operators

"&&" and "||" are AND-OR list operators. "&&" executes the first command, and then executes the second command if and only if the exit status of the first command is zero. "||" is similar, but executes the second command if and only if the exit status of the first command is nonzero. "&&" and "||" both have the same priority. Note that these operators are left-associative, so "true || echo bar && echo baz" writes "baz" and nothing else. This is not the way it works in C. Also, if you forget the left-hand side (for example when continuing lines but forgetting to use a backslash) it defaults to a true statement. This behavior is not useful and should not be relied upon.

Flow-Control Constructs -- if, while, for, case

The syntax of the if command is

```
if list
then list
[ elif list
then    list ] ...
[ else list ]
fi
```

The syntax of the while command is

```
while list
do   list
done
```

The two lists are executed repeatedly while the exit status of the first list is zero. The until command is similar, but has the word until in place of while, which causes it to repeat until the exit status of the first list is zero.

The syntax of the for command is

```
for variable in word ...
do   list
done
```

The words are expanded, and then the list is executed repeatedly with the variable set to each word in turn. do and done may be replaced with "{" and "}".

The syntax of the break and continue command is

```
break [ num ]
continue [ num ]
```

Break terminates the num innermost for or while loops. Continue continues with the next iteration of the innermost loop. These are implemented as built-in commands.

The syntax of the case command is

```
case word in
pattern) list ;;
...
esac
```

The pattern can actually be one or more patterns (see **Shell Patterns** described later), separated by "|" characters.

Grouping Commands Together

Commands may be grouped by writing either

```
(list)
```

or

```
{ list; }
```

The first of these executes the commands in a subshell. Built-in commands grouped into a (list) will not affect the current shell. The second form does not fork another shell so is slightly more efficient. Grouping commands together this way allows you to redirect their output as though they were one program:

```
{ echo -n " hello " ; echo " world" ; } > greeting
```

Note that "}" must follow a control operator (here, ";") so that it is recognized as a reserved word and not as another command argument.

Functions

The syntax of a function definition is

```
name () command
```

A function definition is an executable statement; when executed it installs a function named name and returns an exit status of zero. The command is normally a list enclosed between "{" and "}".

Variables may be declared to be local to a function by using a local command. This should appear as the first statement of a function, and the syntax is

```
local [variable | -] ...
```

"Local" is implemented as a built-in command.

When a variable is made local, it inherits the initial value and exported and read-only flags from the variable with the same name in the surrounding scope, if there is one. Otherwise, the variable is initially unset. The shell uses dynamic scoping, so that if you make the variable x local to function f, which then calls function g, references to the variable x made inside g will refer to the variable x declared inside f, not to the global variable named x.

The only special parameter that can be made local is "-". Making "-" local causes any shell options that are changed via the set command inside the function to be restored to their original values when the function returns.

The syntax of the return command is

```
return [exitstatus]
```

It terminates the currently executing function. Return is implemented as a built-in command.

Variables and Parameters

The shell maintains a set of parameters. A parameter denoted by a name is called a variable. When starting up, the shell turns all the environment variables into shell variables. New variables can be set using the form

```
name=value
```

Variables set by the user must have a name consisting solely of alphabetics, numerics, and underscores - the first of which must not be numeric. A parameter can also be denoted by a number or a special character as explained below.

Positional Parameters

A positional parameter is a parameter denoted by a number (n > 0). The shell sets these initially to the values of its command line arguments that follow the name of the shell script. The **set** built-in can also be used to set or reset them.

Special Parameters

A special parameter is a parameter denoted by one of the following special characters. The value of the parameter is listed next to its character.

* Expands to the positional parameters, starting from one. When the expansion occurs within a double-quoted string it expands to a single field with the value of each parameter separated by the first character of the `IFS` variable, or by a ⟨space⟩ if `IFS` is unset.

@ Expands to the positional parameters, starting from one. When the expansion occurs within double-quotes, each positional parameter expands as a separate argument. If there are no positional parameters, the expansion of @ generates zero arguments, even when @ is double-quoted. What this basically means, for example, is if $1 is “abc” and $2 is “def ghi”, then "$@" expands to the two arguments:

```
"abc" "def ghi"
```

\# Expands to the number of positional parameters.

? Expands to the exit status of the most recent pipeline.

- (Hyphen.) Expands to the current option flags (the single-letter option names concatenated into a string) as specified on invocation, by the set built-in command, or implicitly by the shell.

$ Expands to the process ID of the invoked shell. A subshell retains the same value of $ as its parent.

! Expands to the process ID of the most recent background command executed from the current shell. For a pipeline, the process ID is that of the last command in the pipeline.

0 (Zero.) Expands to the name of the shell or shell script.

Word Expansions

This clause describes the various expansions that are performed on words. Not all expansions are performed on every word, as explained later.

Tilde expansions, parameter expansions, command substitutions, arithmetic expansions, and quote removals that occur within a single word expand to a single field. It is only field splitting or pathname expansion that can create multiple fields from a single word. The single exception to this rule is the expansion of the special parameter @ within double-quotes, as was described above.

The order of word expansion is:

1. Tilde Expansion, Parameter Expansion, Command Substitution, Arithmetic Expansion (these all occur at the same time).
2. Field Splitting is performed on fields generated by step (1) unless the `IFS` variable is null.
3. Pathname Expansion (unless set **`-f`** is in effect).
4. Quote Removal.

The $ character is used to introduce parameter expansion, command substitution, or arithmetic evaluation.

Tilde Expansion (substituting a user's home directory)

A word beginning with an unquoted tilde character (~) is subjected to tilde expansion. All the characters up to a slash (/) or the end of the word are treated as a username and are replaced with the user's home directory. If the username is missing (as in `~/foobar`), the tilde is replaced with the value of the *HOME* variable (the current user's home directory).

Parameter Expansion

The format for parameter expansion is as follows:

```
${expression}
```

where expression consists of all characters until the matching "}". Any "}" escaped by a backslash or within a quoted string, and characters in embedded arithmetic expansions, command substitutions, and variable expansions, are not examined in determining the matching "}".

The simplest form for parameter expansion is:

```
${parameter}
```

The value, if any, of parameter is substituted.

The parameter name or symbol can be enclosed in braces, which are optional except for positional parameters with more than one digit or when parameter is followed by a character that could be interpreted as part of the name. If a parameter expansion occurs inside double-quotes:

1. Pathname expansion is not performed on the results of the expansion.
2. Field splitting is not performed on the results of the expansion, with the exception of the special rules for @.

In addition, a parameter expansion can be modified by using one of the following formats.

${parameter:-word}	Use Default Values. If parameter is unset or null, the expansion of word is substituted; otherwise, the value of parameter is substituted.
${parameter:=word}	Assign Default Values. If parameter is unset or null, the expansion of word is assigned to parameter. In all cases, the final value of parameter is substituted. Only variables, not positional parameters or special parameters, can be assigned in this way.
${parameter:?[word]}	Indicate Error if Null or Unset. If parameter is unset or null, the expansion of word (or a message indicating it is unset if word is omitted) is written to standard error and the shell exits with a nonzero exit status. Otherwise, the value of parameter is substituted. An interactive shell need not exit.
${parameter:+word}	Use Alternative Value. If parameter is unset or null, null is substituted; otherwise, the expansion of word is substituted.

In the parameter expansions shown previously, use of the colon in the format results in a test for a parameter that is unset or null; omission of the colon results in a test for a parameter that is only unset.

${#parameter} String Length. The length in characters of the value of parameter.

The following four varieties of parameter expansion provide for substring processing. In each case, pattern matching notation (see **Shell Patterns**), rather than regular expression notation, is used to evaluate the patterns. If parameter is * or @, the result of the expansion is unspecified. Enclosing the full parameter expansion string in double-quotes does not cause the following four varieties of pattern characters to be quoted, whereas quoting characters within the braces has this effect.

${parameter%word} Remove Smallest Suffix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the smallest portion of the suffix matched by the pattern deleted.

${parameter%%word} Remove Largest Suffix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the largest portion of the suffix matched by the pattern deleted.

${parameter#word} Remove Smallest Prefix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the smallest portion of the prefix matched by the pattern deleted.

${parameter##word} Remove Largest Prefix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the largest portion of the prefix matched by the pattern deleted.

Command Substitution

Command substitution allows the output of a command to be substituted in place of the command name itself. Command substitution occurs when the command is enclosed as follows:

```
$(command)
```

or (“backquoted” version):

```
‘command‘
```

The shell expands the command substitution by executing command in a subshell environment and replacing the command substitution with the standard output of the command, removing sequences of one or more ⟨newline⟩s at the end of the substitution. (Embedded ⟨newline⟩s before the end of the output are not removed; however, during field splitting, they may be translated into ⟨space⟩s, depending on the value of `IFS` and quoting that is in effect.)

Arithmetic Expansion

Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion is as follows:

```
$((expression))
```

The expression is treated as if it were in double-quotes, except that a double-quote inside the expression is not treated specially. The shell expands all tokens in the expression for parameter expansion, command substitution, and quote removal.

Next, the shell treats this as an arithmetic expression and substitutes the value of the expression.

Arithmetic expressions use a syntax similar to that of the C language, and are evaluated using the `intmax_t` data type (this is an extension to POSIX, which requires only `long` arithmetic). Shell variables may be referenced by name inside an arithmetic expression, without needing a “$” sign.

White Space Splitting (Field Splitting)

After parameter expansion, command substitution, and arithmetic expansion the shell scans the results of expansions and substitutions that did not occur in double-quotes for field splitting and multiple fields can result.

The shell treats each character of the `IFS` as a delimiter and use the delimiters to split the results of parameter expansion and command substitution into fields.

Non-whitespace characters in `IFS` are treated strictly as parameter terminators. So adjacent non-whitespace `IFS` characters will produce empty parameters.

If `IFS` is unset it is assumed to contain space, tab, and newline.

Pathname Expansion (File Name Generation)

Unless the **-f** flag is set, file name generation is performed after word splitting is complete. Each word is viewed as a series of patterns, separated by slashes. The process of expansion replaces the word with the names of all existing files whose names can be formed by replacing each pattern with a string that matches the specified pattern. There are two restrictions on this: first, a pattern cannot match a string containing a slash, and second, a pattern cannot match a string starting with a period unless the first character of the pattern is a period. The next section describes the patterns used for both Pathname Expansion and the **case** command.

Shell Patterns

A pattern consists of normal characters, which match themselves, and meta-characters. The meta-characters are "!", "*", "?", and "[". These characters lose their special meanings if they are quoted. When command or variable substitution is performed and the dollar sign or back quotes are not double quoted, the value of the variable or the output of the command is scanned for these characters and they are turned into meta-characters.

An asterisk ("*") matches any string of characters. A question mark matches any single character. A left bracket ("[") introduces a character class. The end of the character class is indicated by a ("]"); if the "]" is missing then the "[" matches a "[" rather than introducing a character class. A character class matches any of the characters between the square brackets. A range of characters may be specified using a minus sign. The character class may be complemented by making an exclamation point the first character of the character class.

To include a "]" in a character class, make it the first character listed (after the "!", if any). To include a minus sign, make it the first or last character listed.

Built-ins

This section lists the built-in commands which are built-in because they need to perform some operation that can't be performed by a separate process. In addition to these, there are several other commands that may be built in for efficiency (e.g. `printf`(1), `echo`(1), `test`(1), etc).

: A null command that returns a 0 (true) exit value.

. file The commands in the specified file are read and executed by the shell.

alias [*name*[=*string* ...]]

If *name*=*string* is specified, the shell defines the alias *name* with value *string*. If just *name* is specified, the value of the alias *name* is printed. With no arguments, the **alias** built-in prints the names and values of all defined aliases (see **unalias**).

bg [*job*] ...

Continue the specified jobs (or the current job if no jobs are given) in the background.

command [**-p**] [**-v**] [**-V**] *command* [*arg ...*]
Execute the specified command but ignore shell functions when searching for it. (This is useful when you have a shell function with the same name as a built-in command.)

-p search for command using a PATH that guarantees to find all the standard utilities.

-V Do not execute the command but search for the command and print the resolution of the command search. This is the same as the **type** built-in.

-v Do not execute the command but search for the command and print the absolute pathname of utilities, the name for built-ins or the expansion of aliases.

cd [*directory* [*replace*]]
Switch to the specified directory (default $HOME). If *replace* is specified, then the new directory name is generated by replacing the first occurrence of *directory* in the current directory name with *replace*. Otherwise if an entry for CDPATH appears in the environment of the **cd** command or the shell variable CDPATH is set and the directory name does not begin with a slash, or its first (or only) component isn't dot or dot dot, then the directories listed in CDPATH will be searched for the specified directory. The format of CDPATH is the same as that of PATH.

In an interactive shell, the **cd** command will print out the name of the directory that it actually switched to if this is different from the name that the user gave. These may be different either because the CDPATH mechanism was used or because a symbolic link was crossed.

eval *string ...*
Concatenate all the arguments with spaces. Then re-parse and execute the command.

exec [*command arg ...*]
Unless command is omitted, the shell process is replaced with the specified program (which must be a real program, not a shell built-in or function). Any redirections on the **exec** command are marked as permanent, so that they are not undone when the **exec** command finishes.

exit [*exitstatus*]
Terminate the shell process. If *exitstatus* is given it is used as the exit status of the shell; otherwise the exit status of the preceding command is used.

export *name ...*

export **-p**
The specified names are exported so that they will appear in the environment of subsequent commands. The only way to un-export a variable is to unset it. The shell allows the value of a variable to be set at the same time it is exported by writing

```
export name=value
```

With no arguments the export command lists the names of all exported variables. With the **-p** option specified the output will be formatted suitably for non-interactive use.

fc [**-e** *editor*] [*first* [*last*]]

fc **-l** [**-nr**] [*first* [*last*]]

fc **-s** [*old=new*] [*first*]
The **fc** built-in lists, or edits and re-executes, commands previously entered to an interactive shell.

-e editor
Use the editor named by editor to edit the commands. The editor string is a command name, subject to search via the PATH variable. The value in the FCEDIT variable is used as a default when **-e** is not specified. If FCEDIT is null or unset, the value of the EDITOR variable is used. If EDITOR is null or unset, ed(1) is used as the editor.

-l (ell)
List the commands rather than invoking an editor on them. The commands are written in the sequence indicated by the first and last operands, as affected by **-r**, with each command preceded by the command number.

-n Suppress command numbers when listing with -l.

-r Reverse the order of the commands listed (with **-l**) or edited (with neither **-l** nor **-s**).

-s Re-execute the command without invoking an editor.

first

last Select the commands to list or edit. The number of previous commands that can be accessed are determined by the value of the HISTSIZE variable. The value of first or last or both are one of the following:

[+]number
A positive number representing a command number; command numbers can be displayed with the **-l** option.

-number
A negative decimal number representing the command that was executed number of commands previously. For example, −1 is the immediately previous command.

string A string indicating the most recently entered command that begins with that string. If the old=new operand is not also specified with **-s**, the string form of the first operand cannot contain an embedded equal sign.

The following environment variables affect the execution of fc:

FCEDIT Name of the editor to use.

HISTSIZE The number of previous commands that are accessible.

fg [*job*]
Move the specified job or the current job to the foreground.

getopts *optstring var*
The POSIX **getopts** command, not to be confused with the *Bell Labs* -derived getopt(1).

The first argument should be a series of letters, each of which may be optionally followed by a colon to indicate that the option requires an argument. The variable specified is set to the parsed option.

The **getopts** command deprecates the older getopt(1) utility due to its handling of arguments containing whitespace.

The **getopts** built-in may be used to obtain options and their arguments from a list of parameters. When invoked, **getopts** places the value of the next option from the option string in the list in the shell variable specified by *var* and its index in the shell variable OPTIND. When the shell is invoked, OPTIND is initialized to 1. For each option that requires an argument, the **getopts** built-in will place it in the shell variable OPTARG. If an option is not allowed for in the *optstring*, then OPTARG will be unset.

optstring is a string of recognized option letters (see getopt(3)). If a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by whitespace. If an option character is not found where expected, **getopts** will set the variable *var* to a "?"; **getopts** will then unset OPTARG and write output to standard error. By specifying a colon as the first character of *optstring* all errors will be ignored.

A nonzero value is returned when the last option is reached. If there are no remaining arguments, **getopts** will set *var* to the special option, "--", otherwise, it will set *var* to "?".

The following code fragment shows how one might process the arguments for a command that can take the options [a] and [b], and the option [c], which requires an argument.

```
while getopts abc: f
do
        case $f in
        a | b)  flag=$f;;
        c)      carg=$OPTARG;;
        \?)     echo $USAGE; exit 1;;
        esac
done
shift 'expr $OPTIND - 1'
```

This code will accept any of the following as equivalent:

```
cmd −acarg file file
cmd −a −c arg file file
cmd −carg -a file file
cmd −a −carg −− file file
```

hash **−rv** *command ...*

The shell maintains a hash table which remembers the locations of commands. With no arguments whatsoever, the **hash** command prints out the contents of this table. Entries which have not been looked at since the last **cd** command are marked with an asterisk; it is possible for these entries to be invalid.

With arguments, the **hash** command removes the specified commands from the hash table (unless they are functions) and then locates them. With the **−v** option, hash prints the locations of the commands as it finds them. The **−r** option causes the hash command to delete all the entries in the hash table except for functions.

inputrc *file*

Read the *file* to set keybindings as defined by editrc(5).

jobid [*job*]

Print the process id's of the processes in the job. If the *job* argument is omitted, the current job is used.

jobs This command lists out all the background processes which are children of the current shell process.

pwd [**−LP**]

Print the current directory. If **−L** is specified the cached value (initially set from PWD) is checked to see if it refers to the current directory, if it does the value is printed. Otherwise the current directory name is found using getcwd(3). The environment variable PWD is set to printed value.

The default is **pwd −L**, but note that the built-in **cd** command doesn't currently support **−L** or **−P** and will cache (almost) the absolute path. If **cd** is changed, **pwd** may be changed to default to **pwd −P**.

If the current directory is renamed and replaced by a symlink to the same directory, or the initial PWD value followed a symbolic link, then the cached value may not be the absolute path.

The built-in command may differ from the program of the same name because the program will use PWD and the built-in uses a separately cached value.

read [**-p** *prompt*] [**-r**] *variable* [*. . .*]
The prompt is printed if the **-p** option is specified and the standard input is a terminal. Then a line is read from the standard input. The trailing newline is deleted from the line and the line is split as described in the section on word splitting above, and the pieces are assigned to the variables in order. If there are more pieces than variables, the remaining pieces (along with the characters in `IFS` that separated them) are assigned to the last variable. If there are more variables than pieces, the remaining variables are assigned the null string. The **read** built-in will indicate success unless EOF is encountered on input, in which case failure is returned.

By default, unless the **-r** option is specified, the backslash "\" acts as an escape character, causing the following character to be treated literally. If a backslash is followed by a newline, the backslash and the newline will be deleted.

readonly *name . . .*

readonly **-p**
The specified names are marked as read only, so that they cannot be subsequently modified or unset. The shell allows the value of a variable to be set at the same time it is marked read only by writing

```
readonly name=value
```

With no arguments the readonly command lists the names of all read only variables. With the **-p** option specified the output will be formatted suitably for non-interactive use.

set [{ **-options** | **+options** | **-- }**] *arg . . .*
The **set** command performs three different functions.

With no arguments, it lists the values of all shell variables.

If options are given, it sets the specified option flags, or clears them as described in the section called **Argument List Processing**.

The third use of the set command is to set the values of the shell's positional parameters to the specified arguments. To change the positional parameters without changing any options, use "--" as the first argument to set. If no arguments are present, the set command will clear all the positional parameters (equivalent to executing "shift $#".)

setvar *variable value*
Assigns value to variable. (In general it is better to write variable=value rather than using **setvar**. **setvar** is intended to be used in functions that assign values to variables whose names are passed as parameters.)

shift [*n*]
Shift the positional parameters n times. A **shift** sets the value of *$1* to the value of *$2*, the value of *$2* to the value of *$3*, and so on, decreasing the value of *$#* by one. If there are zero positional parameters, **shift** does nothing.

trap [**-l**]

trap [*action*] *signal . . .*
Cause the shell to parse and execute action when any of the specified signals are received. The signals are specified by signal number or as the name of the signal. If *signal* is `0`, the action is executed when the shell exits. *action* may be null, which cause the specified signals to be ignored. With *action* omitted or set to '-' the specified signals are set to their default action. When the shell forks off a subshell, it resets trapped (but not ignored) signals to the default action. On non-interactive shells, the **trap** command has no effect on signals that were ignored on entry to the shell. On interactive shells, the **trap** command will catch or reset signals ignored on entry. Issuing **trap** with option *-l* will print a list of valid signal names. **trap** without any arguments cause it to write a

list of signals and their associated action to the standard output in a format that is suitable as an input to the shell that achieves the same trapping results.

Examples:

```
trap
```

List trapped signals and their corresponding action

```
trap -l
```

Print a list of valid signals

```
trap '' INT QUIT tstp 30
```

Ignore signals INT QUIT TSTP USR1

```
trap date INT
```

Print date upon receiving signal INT

type [*name ...*]
Interpret each name as a command and print the resolution of the command search. Possible resolutions are: shell keyword, alias, shell built-in, command, tracked alias and not found. For aliases the alias expansion is printed; for commands and tracked aliases the complete pathname of the command is printed.

ulimit [**-H** | **-S**] [**-a** | **-tfdscmlpn** [*value*]]
Inquire about or set the hard or soft limits on processes or set new limits. The choice between hard limit (which no process is allowed to violate, and which may not be raised once it has been lowered) and soft limit (which causes processes to be signaled but not necessarily killed, and which may be raised) is made with these flags:

-H set or inquire about hard limits

-S set or inquire about soft limits. If neither **-H** nor **-S** is specified, the soft limit is displayed or both limits are set. If both are specified, the last one wins.

The limit to be interrogated or set, then, is chosen by specifying any one of these flags:

-a show all the current limits

-b show or set the limit on the socket buffer size of a process (in bytes)

-t show or set the limit on CPU time (in seconds)

-f show or set the limit on the largest file that can be created (in 512-byte blocks)

-d show or set the limit on the data segment size of a process (in kilobytes)

-s show or set the limit on the stack size of a process (in kilobytes)

-c show or set the limit on the largest core dump size that can be produced (in 512-byte blocks)

-m show or set the limit on the total physical memory that can be in use by a process (in kilobytes)

-l show or set the limit on how much memory a process can lock with `mlock`(2) (in kilobytes)

-p show or set the limit on the number of processes this user can have at one time

-n show or set the limit on the number of files a process can have open at once

If none of these is specified, it is the limit on file size that is shown or set. If value is specified, the limit is set to that number; otherwise the current limit is displayed.

Limits of an arbitrary process can be displayed or set using the `sysctl`(8) utility.

umask [*mask*]
Set the value of umask (see `umask`(2)) to the specified octal value. If the argument is omitted, the umask value is printed.

unalias [**-a**] [*name*]
If *name* is specified, the shell removes that alias. If **-a** is specified, all aliases are removed.

unset *name ...*
The specified variables and functions are unset and unexported. If a given name corresponds to both a variable and a function, both the variable and the function are unset.

wait [*job*]
Wait for the specified job to complete and return the exit status of the last process in the job. If the argument is omitted, wait for all jobs to complete and then return an exit status of zero.

Command Line Editing

When **sh** is being used interactively from a terminal, the current command and the command history (see **fc** in **Built-ins**) can be edited using emacs-mode or vi-mode command-line editing. The command `set -o emacs` enables emacs-mode editing. The command `set -o vi` enables vi-mode editing and places the current shell process into *vi* insert mode. (See the **Argument List Processing** section above.)

The *vi* mode uses commands similar to a subset of those described in the `vi`(1) man page. With vi-mode enabled, **sh** can be switched between insert mode and command mode. It's similar to `vi`(1): pressing the ⟨ESC⟩ key will throw you into command VI command mode. Pressing the ⟨return⟩ key while in command mode will pass the line to the shell.

The *emacs* mode uses commands similar to a subset available in the `emacs`(1) editor. With emacs-mode enabled, special keys can be used to modify the text in the buffer using the control key.

sh uses the `editline`(3) library.

EXIT STATUS

Errors that are detected by the shell, such as a syntax error, will cause the shell to exit with a non-zero exit status. If the shell is not an interactive shell, the execution of the shell file will be aborted. Otherwise the shell will return the exit status of the last command executed, or if the exit built-in is used with a numeric argument, it will return the argument.

ENVIRONMENT

`HOME` Set automatically by `login`(1) from the user's login directory in the password file (`passwd`(5)). This environment variable also functions as the default argument for the **cd** built-in.

`PATH` The default search path for executables. See the above section **Path Search**.

`CDPATH` The search path used with the **cd** built-in.

`LANG` The string used to specify localization information that allows users to work with different culture-specific and language conventions. See `nls`(7).

`MAIL` The name of a mail file, that will be checked for the arrival of new mail. Overridden by `MAILPATH`.

`MAILCHECK` The frequency in seconds that the shell checks for the arrival of mail in the files specified by the `MAILPATH` or the `MAIL` file. If set to 0, the check will occur at each prompt.

`MAILPATH` A colon ":" separated list of file names, for the shell to check for incoming mail. This environment setting overrides the `MAIL` setting. There is a maximum of 10 mailboxes that can be monitored at once.

`PS1` The primary prompt string, which defaults to "$ ", unless you are the superuser, in which case it defaults to "# ".

`PS2` The secondary prompt string, which defaults to "> ".

`PS4` Output before each line when execution trace (set -x) is enabled, defaults to "+ ".

`IFS` Input Field Separators. This is normally set to ⟨space⟩, ⟨tab⟩, and ⟨newline⟩. See the **White Space Splitting** section for more details.

`TERM` The default terminal setting for the shell. This is inherited by children of the shell, and is used in the history editing modes.

`HISTSIZE` The number of lines in the history buffer for the shell.

FILES

`$HOME/.profile`

`/etc/profile`

SEE ALSO

`csh`(1), `echo`(1), `getopt`(1), `ksh`(1), `login`(1), `printf`(1), `test`(1), `editline`(3), `getopt`(3), `editrc`(5), `passwd`(5), `environ`(7), `nls`(7), `sysctl`(8)

HISTORY

A **sh** command appeared in Version 1 AT&T UNIX. It was, however, unmaintainable so we wrote this one.

BUGS

Setuid shell scripts should be avoided at all costs, as they are a significant security risk.

PS1, PS2, and PS4 should be subject to parameter expansion before being displayed.

The characters generated by filename completion should probably be quoted to ensure that the filename is still valid after the input line has been processed.

NAME

shar — create a shell archive of files

SYNOPSIS

shar *file ...*

DESCRIPTION

shar writes an sh(1) shell script to the standard output which will recreate the file hierarchy specified by the command line operands. Directories will be recreated and must be specified before the files they contain (the find(1) utility does this correctly).

shar is normally used for distributing files by ftp(1) or mail(1).

EXAMPLES

To create a shell archive of the program ls(1) and mail it to Rick:

```
cd ls
shar ‘find . -print‘ |  mail -s "ls source" rick
```

To recreate the program directory:

```
mkdir ls
cd ls
...
<delete header lines and examine mailed archive>
...
sh archive
```

SEE ALSO

compress(1), mail(1), tar(1), uuencode(1)

HISTORY

The **shar** command appeared in 4.4BSD.

BUGS

shar makes no provisions for special types of files or files containing magic characters.

SECURITY CONSIDERATIONS

It is easy to insert trojan horses into **shar** files. It is strongly recommended that all shell archive files be examined before running them through sh(1). Archives produced using this implementation of **shar** may be easily examined with the command:

```
egrep -v ’^[X#]’ shar.file
```

NAME

shlock — create or verify a lock file for shell scripts

SYNOPSIS

shlock [**-du**] [**-p** *PID*] **-f** *lockfile*

DESCRIPTION

The **shlock** command can create or verify a lock file on behalf of a shell or other script program. When it attempts to create a lock file, if one already exists, **shlock** verifies that it is or is not valid. If valid, **shlock** will exit with a non-zero exit code. If invalid, **shlock** will remove the lock file, and create a new one.

shlock uses the `link`(2) system call to make the final target lock file, which is an atomic operation (i.e. "dot locking", so named for this mechanism's original use for locking system mailboxes). It puts the process ID ("PID") from the command line into the requested lock file.

shlock verifies that an extant lock file is still valid by using `kill`(2) with a zero signal to check for the existence of the process that holds the lock.

The **-d** option causes **shlock** to be verbose about what it is doing.

The **-f** argument with *lockfile* is always required.

The **-p** option with *PID* is given when the program is to create a lock file; when absent, **shlock** will simply check for the validity of the lock file.

The **-u** option causes **shlock** to read and write the PID as a binary pid_t, instead of as ASCII, to be compatible with the locks created by UUCP.

EXIT STATUS

A zero exit code indicates a valid lock file.

EXAMPLES

BOURNE SHELL

```
#!/bin/sh
lckfile=/tmp/foo.lock
if shlock -f ${lckfile} -p $$
then
#       do what required the lock
        rm ${lckfile}
else
        echo Lock ${lckfile} already held by ‘cat ${lckfile}‘
fi
```

C SHELL

```
#!/bin/csh -f
set lckfile=/tmp/foo.lock
shlock -f ${lckfile} -p $$
if ($status == 0) then
#       do what required the lock
        rm ${lckfile}
else
        echo Lock ${lckfile} already held by ‘cat ${lckfile}‘
endif
```

The examples assume that the file system where the lock file is to be created is writable by the user, and has space available.

HISTORY

shlock was written for the first Network News Transfer Protocol (NNTP) software distribution, released in March 1986. The algorithm was suggested by Peter Honeyman, from work he did on HoneyDanBer UUCP.

AUTHORS

Erik E. Fair ⟨fair@clock.org⟩

BUGS

Does not work on NFS or other network file system on different systems because the disparate systems have disjoint PID spaces.

Cannot handle the case where a lock file was not deleted, the process that created it has exited, and the system has created a new process with the same PID as in the dead lock file. The lock file will appear to be valid even though the process is unrelated to the one that created the lock in the first place. Always remove your lock files after you're done.

NAME

shuffle — print a random permutation of the command line arguments

SYNOPSIS

shuffle [**-0**] [**-f** *filename ...*] [**-n** *number*] [**-p** *number*] [*arg*] [*...*]

DESCRIPTION

The **shuffle** program prints a random permutation (or "shuffle") of its command line arguments. This can be useful in shell scripts for selecting a random order in which to do a set of tasks, view a set of files, etc.

If the **-f** option is given, the data is taken from that files' contents or if the filename is - "stdin".

If the **-n** option is given, its argument is treated as a number, and the program prints a random permutation of the numbers greater than or equal to 0 and less than the argument.

If the **-p** option is given, its argument is treated as a number, and the program prints that number of randomly selected lines or arguments in a random order.

The **-0** option changes the field separator character from \n to \0, so that the output is suitable to be sent to xargs(1) (to handle filenames with whitespace in them).

EXAMPLES

```
$ shuffle a b c d
c
b
d
a
$ shuffle -p 1 a b c d
d
$ shuffle -n 4 -p 2
0
3
```

SEE ALSO

jot(1), random(6)

HISTORY

The **shuffle** program first appeared in NetBSD 1.4.

AUTHORS

Written by Perry E. Metzger (perry@piermont.com).

BUGS

The random number generator isn't that great, and thus the permutations often aren't that great.

NAME

size − list section sizes and total size.

SYNOPSIS

size [**−A**|**−B**|**−−format=***compatibility*]
[**−−help**]
[**−d**|**−o**|**−x**|**−−radix=***number*]
[**−t**|**−−totals**]
[**−−target=***bfdname*] [**−V**|**−−version**]
[*objfile*...]

DESCRIPTION

The GNU **size** utility lists the section sizes−−−and the total size−−−for each of the object or archive files *objfile* in its argument list. By default, one line of output is generated for each object file or each module in an archive.

objfile... are the object files to be examined. If none are specified, the file `a.out` will be used.

OPTIONS

The command line options have the following meanings:

−A
−B
−−format=*compatibility*

Using one of these options, you can choose whether the output from GNU **size** resembles output from System V **size** (using **−A**, or **−−format=sysv**), or Berkeley **size** (using **−B**, or **−−format=berkeley**). The default is the one-line format similar to Berkeley's.

Here is an example of the Berkeley (default) format of output from **size**:

```
$ size --format=Berkeley ranlib size
text    data    bss     dec     hex     filename
294880  81920   11592   388392  5ed28   ranlib
294880  81920   11888   388688  5ee50   size
```

This is the same data, but displayed closer to System V conventions:

```
$ size --format=SysV ranlib size
ranlib  :
section         size         addr
.text         294880         8192
.data          81920       303104
.bss           11592       385024
Total         388392

size  :
section         size         addr
.text         294880         8192
.data          81920       303104
.bss           11888       385024
Total         388688
```

−−help

Show a summary of acceptable arguments and options.

−d
−o
−x
−−radix=*number*

Using one of these options, you can control whether the size of each section is given in decimal (**−d**, or **−−radix=10**); octal (**−o**, or **−−radix=8**); or hexadecimal (**−x**, or **−−radix=16**). In

−−radix=*number*, only the three values (8, 10, 16) are supported. The total size is always given in two radices; decimal and hexadecimal for **−d** or **−x** output, or octal and hexadecimal if you're using **−o**.

−t
−−totals
Show totals of all objects listed (Berkeley format listing mode only).

−−target=*bfdname*
Specify that the object-code format for *objfile* is *bfdname*. This option may not be necessary; **size** can automatically recognize many formats.

−V
−−version
Display the version number of **size**.

SEE ALSO

ar (1), *objdump* (1), *readelf* (1), and the Info entries for *binutils*.

COPYRIGHT

NAME

skey — respond to an OTP challenge

SYNOPSIS

skey [**-n** *count*] [**-p** *password*] [**-t** *hash*] [**-x**] *sequence#* [/] *key*

DESCRIPTION

S/Key is a One Time Password (OTP) authentication system. It is intended to be used when the communication channel between a user and host is not secure (e.g. not encrypted or hardwired). Since each password is used only once, even if it is "seen" by a hostile third party, it cannot be used again to gain access to the host.

S/Key uses 64 bits of information, transformed by the MD4 algorithm into 6 English words. The user supplies the words to authenticate himself to programs like `login`(1) or `ftpd`(8).

Example use of the *S/Key* program **skey**:

```
% skey  99  th91334
Enter password: <your secret password is entered here>
OMEN US HORN OMIT BACK AHOY
%
```

The string that is given back by **skey** can then be used to log into a system.

The programs that are part of the *S/Key* system are:

`skeyinit`(1) used to set up your *S/Key*.

skey used to get the one time password(s).

`skeyinfo`(1) used to initialize the *S/Key* database for the specified user. It also tells the user what the next challenge will be.

`skeyaudit`(1) used to inform users that they will soon have to rerun `skeyinit`(1).

When you run `skeyinit`(1) you inform the system of your secret password. Running **skey** then generates the one-time password(s), after requiring your secret password. If however, you misspell your secret password that you have given to `skeyinit`(1) while running **skey** you will get a list of passwords that will not work, and no indication about the problem.

Password sequence numbers count backward from 99. You can enter the passwords using small letters, even though **skey** prints them capitalized.

The **-n** *count* argument asks for *count* password sequences to be printed out ending with the requested sequence number.

The hash algorithm is selected using the **-t** *hash* option, possible choices here are md4, md5 or sha1.

The **-p** *password* allows the user to specify the *S/Key* password on the command line.

To output the S/Key list in hexadecimal instead of words, use the **-x** option.

EXAMPLES

Initialize generation of one time passwords:

```
host% skeyinit
Password: <normal login password>
[Adding username]
Enter secret password: <new secret password>
Again secret password: <new secret password again>
ID username s/key is 99 host12345
```

```
Next login password: SOME SIX WORDS THAT WERE COMPUTED
```

Produce a list of one time passwords to take with to a conference:

```
host% skey -n 3 99 host12345
Enter secret password: <secret password as used with skeyinit>
97: NOSE FOOT RUSH FEAR GREY JUST
98: YAWN LEO DEED BIND WACK BRAE
99: SOME SIX WORDS THAT WERE COMPUTED
```

Logging in to a host where **skey** is installed:

```
host% telnet host

login: <username>
Password [s/key 97 host12345]:
```

Note that the user can use either his/her *S/Key* password at the prompt but also the normal one unless the **-s** flag is given to login(1).

SEE ALSO

login(1), skeyaudit(1), skeyinfo(1), skeyinit(1), ftpd(8)

RFC 2289

TRADEMARKS AND PATENTS

S/Key is a trademark of Bellcore.

AUTHORS

Phil Karn
Neil M. Haller
John S. Walden
Scott Chasin

NAME

skeyaudit — warn users if their S/Key will soon expire

SYNOPSIS

skeyaudit [*limit*]

DESCRIPTION

skeyaudit searches through the file "/etc/skeykeys" for users whose S/Key sequence number is less than *limit*, and sends them a reminder to run skeyinit(1) soon. If no limit is specified a default of 12 is used.

FILES

/etc/skeykeys The S/Key key information database

SEE ALSO

skey(1), skeyinfo(1), skeyinit(1)

NAME

skeyinfo — obtain the next S/Key challenge for a user

SYNOPSIS

skeyinfo [*user*]

DESCRIPTION

skeyinfo prints out the next S/Key challenge for the specified user or for the current user if no user is specified.

SEE ALSO

skey(1), skeyaudit(1), skeyinit(1)

NAME

skeyinit — change password or add user to S/Key authentication system

SYNOPSIS

skeyinit [**-sxz**] [**-k** *passphrase*] [**-n** *count*] [**-p** *password*] [**-t** *hash*] [*user*]

DESCRIPTION

skeyinit initializes the system so you can use S/Key one-time passwords to login. The program will ask you to enter a secret pass phrase; enter a phrase of several words in response. After the S/Key database has been updated you can login using either your regular password or using S/Key one-time passwords.

skeyinit requires you to type a secret password, so it should be used only on a secure terminal.

OPTIONS

-k *passphrase*
: Use pass phrase *passphrase* instead of asking for one to be entered.

-n *count*
: Start the skey(1) sequence at *count* (default is 100).

-p *password*
: Use password *password* instead of asking for one to be entered.

-s
: allows the user to set the seed and count for complete control of the parameters. To do this run **skeyinit** in one window and put in your count and seed; then run skey(1) in another window to generate the correct 6 english words for that count and seed. You can then “cut-and-paste” or type the words into the **skeyinit** window.

-t *hash*
: Selects the hash algorithm to use. Available choices are md4 (the default), md5, or sha1.

-x
: Displays one-time password in hexadecimal instead of ASCII.

-z
: Allows the user to zero their S/Key entry.

user
: The username to be changed/added. By default the current user is operated on, only root may change other user’s entries.

FILES

/etc/skeykeys database of information for the S/Key system.

SEE ALSO

skey(1), skeyaudit(1), skeyinfo(1)

AUTHORS

Phil Karn
Neil M. Haller
John S. Walden
Scott Chasin

NAME

sleep — suspend execution for an interval of time

SYNOPSIS

sleep *seconds*

DESCRIPTION

The **sleep** utility suspends execution for a minimum of *seconds*. It is usually used to schedule the execution of other commands (see **EXAMPLES** below).

Note: The NetBSD **sleep** command will accept and honor a non-integer number of specified seconds. This is a non-portable extension, and its use will nearly guarantee that a shell script will not execute properly on another system.

EXIT STATUS

The **sleep** utility exits with one of the following values:

`0` On successful completion, or if the signal `SIGALRM` was received.

`>0` An error occurred.

EXAMPLES

To schedule the execution of a command for 1800 seconds later:

```
(sleep 1800; sh command_file >& errors)&
```

This incantation would wait a half hour before running the script command_file. (See the `at`(1) utility.)

To reiteratively run a command (with the `csh`(1)):

```
while (1)
        if (! -r zzz.rawdata) then
                sleep 300
        else
                foreach i (*.rawdata)
                        sleep 70
                        awk -f collapse_data $i >> results
                end
                break
        endif
end
```

The scenario for a script such as this might be: a program currently running is taking longer than expected to process a series of files, and it would be nice to have another program start processing the files created by the first program as soon as it is finished (when zzz.rawdata is created). The script checks every five minutes for the file zzz.rawdata, when the file is found, then another portion processing is done courteously by sleeping for 70 seconds in between each awk job.

SEE ALSO

`at`(1), `nanosleep`(2), `sleep`(3)

STANDARDS

The **sleep** command is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

smbutil — interface to the SMB requester

SYNOPSIS

smbutil [**-hv**] *command* [**-***options*] [*args*]

DESCRIPTION

The **smbutil** command is used to control SMB requester and issue various commands.

There are two types of options — global and local to the specified *command*.

Global options are as follows:

-h Print a short help message.

-v Verbose output.

The commands and local options are:

crypt [*password*]
Encrypt clear text password for use in the ~/.nsmbrc file. The encrypted password starts with the $$1 symbols. Warning: the encryption function is very weak and intended only to hide clear text password. If *password* is omitted from the command line, **smbutil** will prompt for one.

help *command*
Print usage information about *command*.

lc List active connections and their parameters.

login [**-***connection_options*] //*user*@*server*[/*share*]
Login/attach to the specified *server* and/or *share* as *user*. This command will create and authenticate connection to an SMB server, and will leave it active after exit. Thus, it is possible to login only once and then use other SMB commands without authentication procedure and additional connections. For the description of **-***connection_options* refer to the mount_smbfs(8) manpage (all uppercase options are connection options).

logout //*user*@*server*[/*share*]
Logout/detach from the specified *server* and/or *share* as *user*. This command will destroy a connection created by the **login** command. A connection may not be closed immediately if it is used by other programs.

lookup [**-w** *host*] *name*
Resolve the given *name* to an IP address. The NetBIOS name server can be directly specified via the **-w** option.

print [**-***connection_options*] //*user*@*server*/*share file*
Send the given *file* to the specified queue on the remote server. If *file* is -, then standard input will be used.

view [**-***connection_options*] //*user*@*server*
List resources available on the specified *server* for the user *user*.

FILES

~/.nsmbrc Keeps description for each connection. See ./examples/dot.nsmbrc for details.

AUTHORS

Boris Popov ⟨bp@butya.kz⟩, ⟨bp@FreeBSD.org⟩

BUGS

Please report any bugs to the author.

NAME

smtp-sink − multi-threaded SMTP/LMTP test server

SYNOPSIS

smtp-sink [*options*] [**inet:**][*host*]:*port backlog*

smtp-sink [*options*] **unix:***pathname backlog*

DESCRIPTION

smtp-sink listens on the named host (or address) and port. It takes SMTP messages from the network and throws them away. The purpose is to measure client performance, not protocol compliance.

smtp-sink may also be configured to capture each mail delivery transaction to file. Since disk latencies are large compared to network delays, this mode of operation can reduce the maximal performance by several orders of magnitude.

Connections can be accepted on IPv4 or IPv6 endpoints, or on UNIX-domain sockets. IPv4 and IPv6 are the default. This program is the complement of the **smtp-source**(1) program.

Note: this is an unsupported test program. No attempt is made to maintain compatibility between successive versions.

Arguments:

-4 Support IPv4 only. This option has no effect when Postfix is built without IPv6 support.

-6 Support IPv6 only. This option is not available when Postfix is built without IPv6 support.

-8 Do not announce 8BITMIME support.

-a Do not announce SASL authentication support.

-A *delay*
Wait *delay* seconds after responding to DATA, then abort prematurely with a 550 reply status. Do not read further input from the client; this is an attempt to block the client before it sends ".". Specify a zero delay value to abort immediately.

-c Display running counters that are updated whenever an SMTP session ends, a QUIT command is executed, or when "." is received.

-C Disable XCLIENT support.

-d *dump-template*
Dump each mail transaction to a single-message file whose name is created by expanding the *dump-template* via strftime(3) and appending a pseudo-random hexadecimal number (example: "%Y%m%d%H/%M." expands into "2006081203/05.809a62e3"). If the template contains "/" characters, missing directories are created automatically. The message dump format is described below.

Note: this option keeps one capture file open for every mail transaction in progress.

-D *dump-template*
Append mail transactions to a multi-message dump file whose name is created by expanding the *dump-template* via strftime(3). If the template contains "/" characters, missing directories are created automatically. The message dump format is described below.

Note: this option keeps one capture file open for every mail transaction in progress.

-e Do not announce ESMTP support.

-E Do not announce ENHANCEDSTATUSCODES support.

-f *command,command,...*
Reject the specified commands with a hard (5xx) error code. This option implies **-p**.

Examples of commands are CONNECT, HELO, EHLO, LHLO, MAIL, RCPT, VRFY, DATA, ., RSET, NOOP, and QUIT. Separate command names by white space or commas, and use quotes to protect white space from the shell. Command names are case-insensitive.

-F Disable XFORWARD support.

-h *hostname*
Use *hostname* in the SMTP greeting, in the HELO response, and in the EHLO response. The default hostname is "smtp-sink".

-L Enable LMTP instead of SMTP.

-m *count* (default: 256)
An upper bound on the maximal number of simultaneous connections that **smtp-sink** will handle. This prevents the process from running out of file descriptors. Excess connections will stay queued in the TCP/IP stack.

-n *count*
Terminate after *count* sessions. This is for testing purposes.

-p Do not announce support for ESMTP command pipelining.

-P Change the server greeting so that it appears to come through a CISCO PIX system. Implies **-e**.

-q *command,command,...*
Disconnect (without replying) after receiving one of the specified commands.

Examples of commands are CONNECT, HELO, EHLO, LHLO, MAIL, RCPT, VRFY, DATA, ., RSET, NOOP, and QUIT. Separate command names by white space or commas, and use quotes to protect white space from the shell. Command names are case-insensitive.

-r *command,command,...*
Reject the specified commands with a soft (4xx) error code. This option implies **-p**.

Examples of commands are CONNECT, HELO, EHLO, LHLO, MAIL, RCPT, VRFY, DATA, ., RSET, NOOP, and QUIT. Separate command names by white space or commas, and use quotes to protect white space from the shell. Command names are case-insensitive.

-R *root-directory*
Change the process root directory to the specified location. This option requires super-user privileges. See also the **-u** option.

-s *command,command,...*
Log the named commands to syslogd.

Examples of commands are CONNECT, HELO, EHLO, LHLO, MAIL, RCPT, VRFY, DATA, ., RSET, NOOP, and QUIT. Separate command names by white space or commas, and use quotes to protect white space from the shell. Command names are case-insensitive.

-S start-string
An optional string that is prepended to each message that is written to a dump file (see the dump file format description below). The following C escape sequences are supported: \a (bell), \b (backslace), \f (formfeed), \n (newline), \r (carriage return), \t (horizontal tab), \v (vertical tab), *ddd* (up to three octal digits) and \\ (the backslash character).

-t *timeout* (default: 100)
Limit the time for receiving a command or sending a response. The time limit is specified in seconds.

-u *username*
Switch to the specified user privileges after opening the network socket and optionally changing the process root directory. This option is required when the process runs with super-user privileges. See also the **-R** option.

-v Show the SMTP conversations.

-w *delay*
Wait *delay* seconds before responding to a DATA command.

[**inet:**][*host*]:*port*
Listen on network interface *host* (default: any interface) TCP port *port*. Both *host* and *port* may be specified in numeric or symbolic form.

unix:*pathname*
Listen on the UNIX-domain socket at *pathname*.

backlog
The maximum length the queue of pending connections, as defined by the **listen**(2) system call.

DUMP FILE FORMAT

Each dumped message contains a sequence of text lines, terminated with the newline character. The sequence of information is as follows:

- The optional string specified with the **-S** option.
- The **smtp-sink** generated headers as documented below.
- The message header and body as received from the SMTP client.
- An empty line.

The format of the **smtp-sink** generated headers is as follows:

X-Client-Addr: *text*
The client IP address without enclosing []. An IPv6 address is prefixed with "ipv6:". This record is always present.

X-Client-Proto: *text*
The client protocol: SMTP, ESMTP or LMTP. This record is always present.

X-Helo-Args: *text*
The arguments of the last HELO or EHLO command before this mail delivery transaction. This record is present only if the client sent a recognizable HELO or EHLO command before the DATA command.

X-Mail-Args: *text*
The arguments of the MAIL command that started this mail delivery transaction. This record is present exactly once.

X-Rcpt-Args: *text*
The arguments of an RCPT command within this mail delivery transaction. There is one record for each RCPT command, and they are in the order as sent by the client.

Received: *text*
A message header for compatibility with mail processing software. This three-line header marks the end of the headers provided by **smtp-sink**, and is formatted as follows:

from *helo* **([***addr***])**
The HELO or EHLO command argument and client IP address. If the client did not send HELO or EHLO, the client IP address is used instead.

by *host* **(smtp-sink) with** *proto* **id** *random***;**
The hostname specified with the **-h** option, the client protocol (see **X-Client-Proto** above), and the pseudo-random portion of the per-message capture file name.

time-stamp
A time stamp as defined in RFC 2822.

SEE ALSO

smtp-source(1), SMTP/LMTP message generator

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

smtp-source − multi-threaded SMTP/LMTP test generator

SYNOPSIS

smtp-source [*options*] [**inet:**]*host*[:*port*]

smtp-source [*options*] **unix:***pathname*

DESCRIPTION

smtp-source connects to the named *host* and TCP *port* (default: port 25) and sends one or more messages to it, either sequentially or in parallel. The program speaks either SMTP (default) or LMTP. Connections can be made to UNIX-domain and IPv4 or IPv6 servers. IPv4 and IPv6 are the default.

Note: this is an unsupported test program. No attempt is made to maintain compatibility between successive versions.

Arguments:

-4 Connect to the server with IPv4. This option has no effect when Postfix is built without IPv6 support.

-6 Connect to the server with IPv6. This option is not available when Postfix is built without IPv6 support.

-c Display a running counter that is incremented each time an SMTP DATA command completes.

-C *count*
When a host sends RESET instead of SYN|ACK, try *count* times before giving up. The default count is 1. Specify a larger count in order to work around a problem with TCP/IP stacks that send RESET when the listen queue is full.

-d Don't disconnect after sending a message; send the next message over the same connection.

-f *from* Use the specified sender address (default: <foo@myhostname>).

-l *length*
Send *length* bytes as message payload. The length does not include message headers.

-L Speak LMTP rather than SMTP.

-m *message_count*
Send the specified number of messages (default: 1).

-M *myhostname*
Use the specified hostname or [address] in the HELO command and in the default sender and recipient addresses, instead of the machine hostname.

-N Prepend a non-repeating sequence number to each recipient address. This avoids the artificial 100% hit rate in the resolve and rewrite client caches and exercises the trivial-rewrite daemon, better approximating Postfix performance under real-life work-loads.

-o Old mode: don't send HELO, and don't send message headers.

-r *recipient_count*
Send the specified number of recipients per transaction (default: 1). Recipient names are generated by prepending a number to the recipient address.

-s *session_count*
Run the specified number of SMTP sessions in parallel (default: 1).

-S *subject*
Send mail with the named subject line (default: none).

-t *to* Use the specified recipient address (default: <foo@myhostname>).

-R *interval*
Wait for a random period of time 0 <= n <= interval between messages. Suspending one thread does not affect other delivery threads.

-v Make the program more verbose, for debugging purposes.

-w *interval*
Wait a fixed time between messages. Suspending one thread does not affect other delivery threads.

[**inet:**]*host*[:*port*]
Connect via TCP to host *host*, port *port*. The default port is **smtp**.

unix:*pathname*
Connect to the UNIX-domain socket at *pathname*.

BUGS

No SMTP command pipelining support.

SEE ALSO

smtp-sink(1), SMTP/LMTP message dump

LICENSE

The Secure Mailer license must be distributed with this software.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

sntp − standard SNTP program

SYNOPSIS

sntp [−*flag* [*value*]]... [−**-***opt-name* [[=|]*value*]]...

All arguments must be options.

DESCRIPTION

This manual page documents, briefly, the **sntp** command. *sntp* can be used as a SNTP client to query a NTP or SNTP server and either display the time or set the local system's time (given suitable privilege). It can be run as an interactive command or in a *cron* job. NTP is the Network Time Protocol (RFC 1305) and SNTP is the Simple Network Time Protocol (RFC 2030, which supersedes RFC 1769).

Options

sntp recognizes the following options:

−v indicates that diagnostic messages for non-fatal errors and a limited amount of tracing should be written to standard error. Fatal ones always produce a diagnostic. This option should be set when there is a suspected problem with the server, network or the source.

−V requests more and less comprehensible output, mainly for investigating problems with apparently inconsistent timestamps. This option should be set when the program fails with a message indicating that is the trouble.

−W requests very verbose debugging output, and will interfere with the timing when writing to the terminal (because of line buffered output from C). Note that the times produced by this are the corrections needed, and not the error in the local clock. This option should be set only when debugging the source.

−q indicates that it should query a daemon save file being maintained by it. This needs no privilege and will change neither the save file nor the clock.

The default is that it should behave as a client, and the following options are then relevant:

−r indicates that the system clock should be reset by *settimeofday*. Naturally, this will work only if the user has enough privilege.

−a indicates that the system clock should be reset by *adjtime*. Naturally, this will work only if the user has enough privilege.

The default is to write the estimated correct local date and time (i.e. not UTC) to the standard output in a format like **'1996 Oct 15 20:17:25.123 + 4.567 +/- 0.089 secs'**, where the **'+ 4.567 +/- 0.089 secs'** indicates the estimated error in the time on the local system.

−l *lockfile*

sets the name of the lock file to ensure that there is only one copy of *sntp* running at once. The default is installation-dependent, but will usually be */etc/sntp.pid*.

−e *minerr*

sets the maximum ignorable variation between the clocks to *minerr*. Acceptable values are from 0.001 to 1, and the default is 0.1 if a NTP host is is specified and 0.5 otherwise.

−E *maxerr*

sets the maximum value of various delays that are deemed acceptable to *maxerr*. Acceptable values are from 1 to 60, and the default is 5. It should sometimes be increased if there are problems with the network, NTP server or system clock, but take care.

−P *prompt*

sets the maximum clock change that will be made automatically to *maxerr*. Acceptable values are from 1 to 3600 or *no*, and the default is 30. If the program is being run interactively in ordinary client mode, and the system clock is to be changed, larger corrections will prompt the user for confirmation. Specifying *no* will disable this and the correction will be made regardless.

−c *count*

sets the maximum number of NTP packets required to *count*. Acceptable values are from 1 to 25 if a NTP host is specified and from 5 to 25 otherwise, and the default is 5. If the maximum isn't enough, the system needs a better consistency algorithm than this program uses.

−d *delay*

sets a rough limit on the total running time to *delay* seconds. Acceptable values are from 1 to 3600, and the default is 15 if a NTP host is specified and 300 otherwise.

−4 force IPv4 DNS resolution.

−6 force IPv6 DNS resolution.

address(es) are the DNS names or IP numbers of hosts to use for the challenge and response protocol; if no names are given, the program waits for broadcasts. Polling a server is vastly more reliable than listening to broadcasts. Note that a single component numeric address is not allowed, to avoid ambiguities. If more than one name is give, they will be used in a round-robin fashion.

Constraints:

minerr must be less than **maxerr** which must be less than **delay** (or, if a NTP host is not specified **delay**/**count**), and **count** must be less than half of **delay**.

In update mode, **maxerr** must be less than **prompt.**

Note that none of the above values are closely linked to the limits described in the NTP protocol (RFC 1305).

USAGE

The simplest use of this program is as an unprivileged command to check the current time and error in the local clock. For example:

sntp ntpserver.somewhere

With suitable privilege, it can be run as a command or in a *cron* job to reset the local clock from a reliable server, like the *ntpdate* and *rdate* commands. For example:

sntp −a ntpserver.somewhere

More information on how to use this utility is given in the *README* file in the distribution. In particular, this *man* page does not describe how to set it up as a server, which needs special care to avoid propagating misinformation.

RETURN VALUE

When used as a client in non-daemon mode, the program returns a zero exit status for success, and a non-zero one otherwise. When used as a daemon (either client or server), it does not return except after a serious error.

BUGS

The program implements the SNTP protocol, and does not provide all NTP facilities. In particular, it contains no checks against any form of spoofing. If this is a serious concern, some network security mechanism (like a firewall or even just *tcpwrappers*) should be installed.

There are some errors, ambiguities and inconsistencies in the RFCs, and this code may not interwork with all other NTP implementations. Any unreasonable restrictions should be reported as bugs to whoever is responsible. It may be difficult to find out who that is.

The program will stop as soon as it feels that things have got out of control. In client daemon mode, it will usually fail during an extended period of network or server inaccessibility or excessively slow performance, or when the local clock is reset by another process. It will then need restarting manually. Experienced system administrators can write a shell script, a *cron* job or put it in *inittab*, to do this automatically.

The error cannot be estimated reliably with broadcast packets or for the drift in daemon mode (even with client-server packets), and the guess made by the program may be wrong (possibly even very wrong). If this is a problem, then setting the **−c** option to a larger value may help. Or it may not.

AUTHOR

sntp was developed by N.M. Maclaren of the University of Cambridge Computing Service.

OPTIONS

−4, **−-ipv4**

Force IPv4 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, **−-ipv6**

Force IPv6 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−u, **−-unprivport**

Use an unprivileged port.

Use an unprivilegded UDP port for our queries.

−v, **−-normalverbose**

Slightly verbose. This option must not appear in combination with any of the following options: extraverbose, megaverbose.

Diagnostic messages for non-fatal errors and a limited amount of tracing should be written to standard error. Fatal ones always produce a diagnostic. This option should be set when there is a suspected problem with the server, network or the source.

−V, **−-extraverbose**

Extra verbose. This option must not appear in combination with any of the following options: normalverbose, megaverbose.

Produce more and less comprehensible output, mainly for investigating problems with apparently inconsistent timestamps. This option should be set when the program fails with a message indicating that is the trouble.

−W, **−-megaverbose**

Mega verbose. This option must not appear in combination with any of the following options: normalverbose, extraverbose.

Very verbose debugging output that will interfere with the timing when writing to the terminal (because of line buffered output from C). Note that the times produced by this are the corrections needed, and not the error in the local clock. This option should be set only when debugging the source.

−r, **−-settimeofday**

Set (step) the time with settimeofday(). This option must not appear in combination with any of the following options: adjtime.

−a, **−-adjtime**

Set (slew) the time with adjtime(). This option must not appear in combination with any of the following options: settimeofday.

−?, **−-help**

Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−> [*rcfile*], **−-save-opts**[=*rcfile*]
Save the option state to *rcfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below.

−< *rcfile*, **−-load-opts**=*rcfile*, **−-no-load-opts**
Load options from *rcfile*. The *no-load-opts* form will disable the loading of earlier RC/INI files. *--no-load-opts* is handled early, out of order.

−v [{*v|c|n*}], **−-version**[*={v|c|n}*]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:
SNTP_<option-name> or **SNTP**
The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those directories.

AUTHOR

ntp.org
Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the MSNTP software is with you. Should MSNTP prove defective, you assume the cost of all necessary servicing or repair.

In no event, unless required by law, will the Author or the University of Cambridge, or any other party who may modify and redistribute this software as permitted in accordance with the provisions below, be liable for damages for any losses whatsoever, including but not limited to lost profits, lost monies, lost or corrupted data, or other special, incidental or consequential losses that may arise out of the use or inability to use the MSNTP software.

COPYING POLICY

Permission is hereby granted for copying and distribution of copies of the MSNTP source and binary files, and of any part thereof, subject to the following licence conditions:

1. You may distribute MSNTP or components of MSNTP, with or without additions developed by you or by others. No charge, other than an "at-cost" distribution fee, may be charged for copies, derivations, or distributions of this material without the express written consent of the copyright holders.

2. You may also distribute MSNTP along with any other product for sale, provided that the cost of the bundled package is the same regardless of whether MSNTP is included or not, and provided that those interested only in MSNTP must be notified that it is a product freely available from the University of Cambridge.

3. If you distribute MSNTP software or parts of MSNTP, with or without additions developed by you or others, then you must either make available the source to all portions of the MSNTP system (exclusive of any additions made by you or by others) upon request, or instead you may notify anyone requesting source that it is freely available from the University of Cambridge.

4. You may not omit any of the copyright notices on either the source files, the executable files, or the documentation.

5. You may not omit transmission of this License agreement with whatever portions of MSNTP that are distributed.

6. Any users of this software must be notified that it is without warranty or guarantee of any nature, express or implied, nor is there any fitness for use represented.

October 1996
April 1997
October 2000

This manual page was *AutoGen*-erated from the **sntp** option definitions.

NAME

sntp − standard SNTP program

SYNOPSIS

sntp [−*flag* [*value*]]... [−**-***opt-name* [[=|]*value*]]...

All arguments must be options.

DESCRIPTION

This manual page documents, briefly, the **sntp** command. *sntp* can be used as a SNTP client to query a NTP or SNTP server and either display the time or set the local system's time (given suitable privilege). It can be run as an interactive command or in a *cron* job. NTP is the Network Time Protocol (RFC 1305) and SNTP is the Simple Network Time Protocol (RFC 2030, which supersedes RFC 1769).

Options

sntp recognizes the following options:

−v indicates that diagnostic messages for non-fatal errors and a limited amount of tracing should be written to standard error. Fatal ones always produce a diagnostic. This option should be set when there is a suspected problem with the server, network or the source.

−V requests more and less comprehensible output, mainly for investigating problems with apparently inconsistent timestamps. This option should be set when the program fails with a message indicating that is the trouble.

−W requests very verbose debugging output, and will interfere with the timing when writing to the terminal (because of line buffered output from C). Note that the times produced by this are the corrections needed, and not the error in the local clock. This option should be set only when debugging the source.

−q indicates that it should query a daemon save file being maintained by it. This needs no privilege and will change neither the save file nor the clock.

The default is that it should behave as a client, and the following options are then relevant:

−r indicates that the system clock should be reset by *settimeofday*. Naturally, this will work only if the user has enough privilege.

−a indicates that the system clock should be reset by *adjtime*. Naturally, this will work only if the user has enough privilege.

The default is to write the estimated correct local date and time (i.e. not UTC) to the standard output in a format like **'1996 Oct 15 20:17:25.123 + 4.567 +/- 0.089 secs'**, where the **'+ 4.567 +/- 0.089 secs'** indicates the estimated error in the time on the local system.

−l *lockfile*

sets the name of the lock file to ensure that there is only one copy of *sntp* running at once. The default is installation-dependent, but will usually be */etc/sntp.pid*.

−e *minerr*

sets the maximum ignorable variation between the clocks to *minerr*. Acceptable values are from 0.001 to 1, and the default is 0.1 if a NTP host is is specified and 0.5 otherwise.

−E *maxerr*

sets the maximum value of various delays that are deemed acceptable to *maxerr*. Acceptable values are from 1 to 60, and the default is 5. It should sometimes be increased if there are problems with the network, NTP server or system clock, but take care.

−P *prompt*

sets the maximum clock change that will be made automatically to *maxerr*. Acceptable values are from 1 to 3600 or *no*, and the default is 30. If the program is being run interactively in ordinary client mode, and the system clock is to be changed, larger corrections will prompt the user for confirmation. Specifying *no* will disable this and the correction will be made regardless.

−c *count*

sets the maximum number of NTP packets required to *count*. Acceptable values are from 1 to 25 if a NTP host is specified and from 5 to 25 otherwise, and the default is 5. If the maximum isn't enough, the system needs a better consistency algorithm than this program uses.

−d *delay*

sets a rough limit on the total running time to *delay* seconds. Acceptable values are from 1 to 3600, and the default is 15 if a NTP host is specified and 300 otherwise.

−4 force IPv4 DNS resolution.

−6 force IPv6 DNS resolution.

address(es) are the DNS names or IP numbers of hosts to use for the challenge and response protocol; if no names are given, the program waits for broadcasts. Polling a server is vastly more reliable than listening to broadcasts. Note that a single component numeric address is not allowed, to avoid ambiguities. If more than one name is give, they will be used in a round-robin fashion.

Constraints:

minerr must be less than **maxerr** which must be less than **delay** (or, if a NTP host is not specified **delay**/**count**), and **count** must be less than half of **delay**.

In update mode, **maxerr** must be less than **prompt.**

Note that none of the above values are closely linked to the limits described in the NTP protocol (RFC 1305).

USAGE

The simplest use of this program is as an unprivileged command to check the current time and error in the local clock. For example:

sntp ntpserver.somewhere

With suitable privilege, it can be run as a command or in a *cron* job to reset the local clock from a reliable server, like the *ntpdate* and *rdate* commands. For example:

sntp −a ntpserver.somewhere

More information on how to use this utility is given in the *README* file in the distribution. In particular, this *man* page does not describe how to set it up as a server, which needs special care to avoid propagating misinformation.

RETURN VALUE

When used as a client in non-daemon mode, the program returns a zero exit status for success, and a non-zero one otherwise. When used as a daemon (either client or server), it does not return except after a serious error.

BUGS

The program implements the SNTP protocol, and does not provide all NTP facilities. In particular, it contains no checks against any form of spoofing. If this is a serious concern, some network security mechanism (like a firewall or even just *tcpwrappers*) should be installed.

There are some errors, ambiguities and inconsistencies in the RFCs, and this code may not interwork with all other NTP implementations. Any unreasonable restrictions should be reported as bugs to whoever is responsible. It may be difficult to find out who that is.

The program will stop as soon as it feels that things have got out of control. In client daemon mode, it will usually fail during an extended period of network or server inaccessibility or excessively slow performance, or when the local clock is reset by another process. It will then need restarting manually. Experienced system administrators can write a shell script, a *cron* job or put it in *inittab*, to do this automatically.

The error cannot be estimated reliably with broadcast packets or for the drift in daemon mode (even with client-server packets), and the guess made by the program may be wrong (possibly even very wrong). If this is a problem, then setting the **−c** option to a larger value may help. Or it may not.

AUTHOR

sntp was developed by N.M. Maclaren of the University of Cambridge Computing Service.

OPTIONS

−4, **−-ipv4**

Force IPv4 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv4 namespace.

−6, **−-ipv6**

Force IPv6 DNS name resolution. This option is a member of the ipv4 class of options.

Force DNS resolution of following host names on the command line to the IPv6 namespace.

−u, **−-unprivport**

Use an unprivileged port.

Use an unprivilegded UDP port for our queries.

−v, **−-normalverbose**

Slightly verbose. This option must not appear in combination with any of the following options: extraverbose, megaverbose.

Diagnostic messages for non-fatal errors and a limited amount of tracing should be written to standard error. Fatal ones always produce a diagnostic. This option should be set when there is a suspected problem with the server, network or the source.

−V, **−-extraverbose**

Extra verbose. This option must not appear in combination with any of the following options: normalverbose, megaverbose.

Produce more and less comprehensible output, mainly for investigating problems with apparently inconsistent timestamps. This option should be set when the program fails with a message indicating that is the trouble.

−W, **−-megaverbose**

Mega verbose. This option must not appear in combination with any of the following options: normalverbose, extraverbose.

Very verbose debugging output that will interfere with the timing when writing to the terminal (because of line buffered output from C). Note that the times produced by this are the corrections needed, and not the error in the local clock. This option should be set only when debugging the source.

−r, **−-settimeofday**

Set (step) the time with settimeofday(). This option must not appear in combination with any of the following options: adjtime.

−a, **−-adjtime**

Set (slew) the time with adjtime(). This option must not appear in combination with any of the following options: settimeofday.

−?, **−-help**

Display usage information and exit.

−!, **−-more-help**
Extended usage information passed thru pager.

−> [*rcfile*], **−-save-opts**[=*rcfile*]
Save the option state to *rcfile*. The default is the *last* configuration file listed in the **OPTION PRESETS** section, below.

−< *rcfile*, **−-load-opts**=*rcfile*, **−-no-load-opts**
Load options from *rcfile*. The *no-load-opts* form will disable the loading of earlier RC/INI files. *--no-load-opts* is handled early, out of order.

−v [{*v|c|n*}], **−-version**[*={v|c|n}*]
Output version of program and exit. The default mode is 'v', a simple version. The 'c' mode will print copyright information and 'n' will print the full copyright notice.

OPTION PRESETS

Any option that is not marked as *not presettable* may be preset by loading values from configuration ("RC" or ".INI") file(s) and values from environment variables named:
SNTP_<option-name> or **SNTP**
The environmental presets take precedence (are processed later than) the configuration files. The *homerc* files are "*$HOME*", and ".". If any of these are directories, then the file *.ntprc* is searched for within those directories.

AUTHOR

ntp.org
Please send bug reports to: http://bugs.ntp.isc.org, bugs@ntp.org

merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the MSNTP software is with you. Should MSNTP prove defective, you assume the cost of all necessary servicing or repair.

In no event, unless required by law, will the Author or the University of Cambridge, or any other party who may modify and redistribute this software as permitted in accordance with the provisions below, be liable for damages for any losses whatsoever, including but not limited to lost profits, lost monies, lost or corrupted data, or other special, incidental or consequential losses that may arise out of the use or inability to use the MSNTP software.

COPYING POLICY

Permission is hereby granted for copying and distribution of copies of the MSNTP source and binary files, and of any part thereof, subject to the following licence conditions:

1. You may distribute MSNTP or components of MSNTP, with or without additions developed by you or by others. No charge, other than an "at-cost" distribution fee, may be charged for copies, derivations, or distributions of this material without the express written consent of the copyright holders.

2. You may also distribute MSNTP along with any other product for sale, provided that the cost of the bundled package is the same regardless of whether MSNTP is included or not, and provided that those interested only in MSNTP must be notified that it is a product freely available from the University of Cambridge.

3. If you distribute MSNTP software or parts of MSNTP, with or without additions developed by you or others, then you must either make available the source to all portions of the MSNTP system (exclusive of any additions made by you or by others) upon request, or instead you may notify anyone requesting source that it is freely available from the University of Cambridge.

4. You may not omit any of the copyright notices on either the source files, the executable files, or the documentation.

5. You may not omit transmission of this License agreement with whatever portions of MSNTP that are distributed.

6. Any users of this software must be notified that it is without warranty or guarantee of any nature, express or implied, nor is there any fitness for use represented.

October 1996
April 1997
October 2000

This manual page was *AutoGen*-erated from the **sntp** option definitions.

NAME

sockstat — list open sockets

SYNOPSIS

sockstat [**-46clnu**] [**-f** *address_family*] [**-p** *ports*]

DESCRIPTION

The **sockstat** command lists open Internet or UNIX domain sockets.

The following options are available:

-4 Show AF_INET (IPv4) sockets.

-6 Show AF_INET6 (IPv6) sockets.

-c Show connected sockets.

-f *address_family*
Limit listed sockets to those of the specified *address_family*. The following address families are recognized: *inet*, for AF_INET; *inet6*, for AF_INET6; and *local* or *unix*, for AF_LOCAL.

-l Show listening sockets.

-n Numeric output only. No attempt will be made to look up symbolic names for addresses and ports.

-p *ports* Only show Internet sockets if either the local or foreign port number is on the specified list. The *ports* argument is a comma-separated list of port numbers and ranges specified as first and last port separated by a dash.

-u Show AF_LOCAL (UNIX) sockets.

If neither **-4**, **-6**, nor **-u** are specified, **sockstat** will list sockets in all three domains.

If neither **-c** nor **-l** are specified, **sockstat** will list both listening and connected sockets, as well as those sockets that are in neither state.

The information listed for each socket is:

USER The user who owns the socket.

COMMAND The command which holds the socket.

PID The process ID of the command which holds the socket.

FD The file descriptor number of the socket.

PROTO The transport protocol associated with the socket for Internet sockets, or the type of socket (stream or datagram) for UNIX sockets.

LOCAL ADDRESS For Internet sockets, this is the address to which the local end of the socket is bound (see getsockname(2)). For bound UNIX sockets, it is the socket's filename or "-".

FOREIGN ADDRESS The address to which the foreign end of the socket is bound (see getpeername(2)) or "-" for unconnected UNIX sockets.

SEE ALSO

fstat(1), netstat(1), inet(4), inet6(4), unix(4)

HISTORY

The **sockstat** command appeared in FreeBSD 3.1. It was then rewritten for NetBSD 3.0.

AUTHORS

This version of the **sockstat** command was written by Andrew Brown ⟨atatat@NetBSD.org⟩. This manual page was written by Dag-Erling Smørgrav ⟨des@FreeBSD.org⟩ and was adapted to match the NetBSD implementation by Andrew Brown ⟨atatat@NetBSD.org⟩.

NAME

soelim — eliminate .so's from nroff input

SYNOPSIS

soelim [**-I** *directory*] [*file ...*]

DESCRIPTION

soelim reads the specified files or the standard input and performs the textual inclusion implied by the nroff(1) directives of the form:

```
.so somefile
```

The directives need to appear at the beginning of input lines. This is useful since programs such as tbl(1) do not normally do this; it allows the placement of individual tables in separate files to be run as a part of a large document.

An argument consisting of a single minus '**-**' is taken to be a file name corresponding to the standard input.

Note that inclusion can be suppressed by using '\\'' instead of '\.', i.e.

```
´so /usr/lib/tmac.s
```

A sample usage of **soelim** would be

```
soelim exum?.n | tbl | nroff -ms | col | lpr
```

The options are as follows:

-I Add the specified directory to the search path for input files.

SEE ALSO

colcrt(1), more(1)

HISTORY

The **soelim** command appeared in 3.0 BSD.

BUGS

The format of the source commands must involve no strangeness − exactly one blank must precede and no blanks follow the file name.

NAME

sort — sort or merge text files

SYNOPSIS

sort [**-bcdfHimnrSsu**] [**-k** *field1*[,*field2*]] [**-o** *output*] [**-R** *char*] [**-T** *dir*] [**-t** *char*] [*file ...*]

DESCRIPTION

The **sort** utility sorts text files by lines. Comparisons are based on one or more sort keys extracted from each line of input, and are performed lexicographically. By default, if keys are not given, **sort** regards each input line as a single field.

The following options are available:

-c Check that the single input file is sorted. If the file is not sorted, **sort** produces the appropriate error messages and exits with code 1; otherwise, **sort** returns 0. **sort -c** produces no output.

-m Merge only; the input files are assumed to be pre-sorted.

-o *output*
The argument given is the name of an *output* file to be used instead of the standard output. This file can be the same as one of the input files.

-T *dir* Use *dir* as the directory for temporary files. The default is the value specified in the environment variable TMPDIR or /tmp if TMPDIR is not defined.

-u Unique: suppress all but one in each set of lines having equal keys. If used with the **-c** option, check that there are no lines with duplicate keys.

The following options override the default ordering rules. When ordering options appear independent of key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key (see **-k**), the ordering options override all global ordering options for that key.

-d Only blank space and alphanumeric characters are used in making comparisons.

-f Considers all lowercase characters that have uppercase equivalents to be the same for purposes of comparison.

-i Ignore all non-printable characters.

-n An initial numeric string, consisting of optional blank space, optional minus sign, and zero or more digits (including decimal point) is sorted by arithmetic value. (The **-n** option no longer implies the **-b** option.)

-r Reverse the sense of comparisons.

-S Don't use stable sort. Default is to use stable sort.

-s Use stable sort. This is the default. Provided for compatibility with other **sort** implementations only.

-H Use a merge sort instead of a radix sort. This option should be used for files larger than 60Mb.

The treatment of field separators can be altered using these options:

-b Ignores leading blank space when determining the start and end of a restricted sort key. A **-b** option specified before the first **-k** option applies globally to all **-k** options. Otherwise, the **-b** option can be attached independently to each *field* argument of the **-k** option (see below). Note that the **-b** option has no effect unless key fields are specified.

-t *char* *char* is used as the field separator character. The initial *char* is not considered to be part of a field when determining key offsets (see below). Each occurrence of *char* is significant (for example, "*charchar*" delimits an empty field). If **-t** is not specified, the default field separator is a sequence of blank-space characters, and consecutive blank spaces do *not* delimit an empty field; further, the initial blank space *is* considered part of a field when determining key offsets.

-R *char* *char* is used as the record separator character. This should be used with discretion; **-R** *<alphanumeric>* usually produces undesirable results. The default record separator is newline.

-k *field1*[,*field2*]
Designates the starting position, *field1*, and optional ending position, *field2*, of a key field. The **-k** option replaces the obsolescent options +*pos1* and **-***pos2*.

The following operands are available:

file The pathname of a file to be sorted, merged, or checked. If no *file* operands are specified, or if a *file* operand is **-**, the standard input is used.

A field is defined as a minimal sequence of characters followed by a field separator or a newline character. By default, the first blank space of a sequence of blank spaces acts as the field separator. All blank spaces in a sequence of blank spaces are considered as part of the next field; for example, all blank spaces at the beginning of a line are considered to be part of the first field.

Fields are specified by the **-k** *field1*[,*field2*] argument. A missing *field2* argument defaults to the end of a line.

The arguments *field1* and *field2* have the form *m.n* and can be followed by one or more of the letters **b**, **d**, **f**, **i**, **n**, and **r**, which correspond to the options discussed above. A *field1* position specified by *m.n* (*m*, *n* > 0) is interpreted as the *n*th character in the *m*th field. A missing *.n* in *field1* means '.1', indicating the first character of the *m*th field; if the **-b** option is in effect, *n* is counted from the first non-blank character in the *m*th field; *m*.1b refers to the first non-blank character in the *m*th field.

A *field2* position specified by *m.n* is interpreted as the *n*th character (including separators) of the *m*th field. A missing *.n* indicates the last character of the *m*th field; *m* = 0 designates the end of a line. Thus the option **-k** *v.x,w.y* is synonymous with the obsolescent option +*v-1.x-1* **-***w-1.y*; when *y* is omitted, **-k** *v.x,w* is synonymous with +*v-1.x-1* **-***w*+*1*.0. The obsolescent +*pos1* **-***pos2* option is still supported, except for **-***w*.0b, which has no **-k** equivalent.

RETURN VALUES

Sort exits with one of the following values:

0 Normal behavior.
1 On disorder (or non-uniqueness) with the **-c** option
2 An error occurred.

ENVIRONMENT

If the following environment variable exists, it is used by **sort**.

TMPDIR **sort** uses the contents of the TMPDIR environment variable as the path in which to store temporary files.

FILES

/tmp/sort.* Default temporary files.

*output*NUMBER Temporary file which is used for output if *output* already exists. Once sorting is finished, this file replaces *output* (via link(2) and unlink(2)).

SEE ALSO

comm(1), join(1), uniq(1), qsort(3), radixsort(3)

HISTORY

A **sort** command appeared in Version 5 AT&T UNIX. This **sort** implementation appeared in 4.4BSD and is used since NetBSD 1.6.

BUGS

To sort files larger than 60Mb, use **sort -H**; files larger than 704Mb must be sorted in smaller pieces, then merged.

NOTES

This **sort** has no limits on input line length (other than imposed by available memory) or any restrictions on bytes allowed within lines.

To protect data **sort -o** calls link(2) and unlink(2), and thus fails on protected directories.

Input files should be text files. If file doesn't end with record separator (which is typically newline), the **sort** utility silently supplies one.

The current **sort** uses lexicographic radix sorting, which requires that sort keys be kept in memory (as opposed to previous versions which used quick and merge sorts and did not.) Thus performance depends highly on efficient choice of sort keys, and the **-b** option and the *field2* argument of the **-k** option should be used whenever possible. Similarly, **sort -k1f** is equivalent to **sort -f** and may take twice as long.

NAME

spell — find spelling errors

SYNOPSIS

spell [**-biltvx**] [**-d** *list*] [**-h** *spellhist*] [**-m** *a* | *e* | *l* | *m* | *s*] [**-s** *stop*] [+*extra_list*] [*file ...*]

DESCRIPTION

spell collects words from the named documents and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes or suffixes) from words in the spelling list are printed on the standard output.

If no files are named, words are collected from the standard input. **spell** ignores most troff(1), tbl(1), eqn(1), and pic(1) constructions. Copies of all output may be accumulated in the history file, if one is specified.

By default, **spell** (like deroff(1)) follows chains of included files (".so" and ".nx" commands).

The default spelling list is based on Webster's Second International dictionary and should be fairly complete. Words that appear in the "stop list" are immediately flagged as misspellings, regardless of whether or not they exist in one of the word lists. This helps filter out misspellings (e.g. thier=thy−y+ier) that would otherwise pass. Additionally, the british file is also used as a stop list unless the **-b** option is specified.

Site administrators may add words to the local word list, /usr/local/share/dict/words or the local stop list, /usr/local/share/dict/stop.

All word (and stop) lists must be sorted in lexicographical order with case folded. The simplest way to achieve this is to use "sort -df". If the word files are incorrectly sorted, **spell** will not be able to operate correctly.

The options are as follows:

-b Check British spelling. Besides preferring *centre*, *colour*, *speciality*, *travelled*, etc., this option insists upon **-ise** in words like *standardise*, Fowler and the OED to the contrary notwithstanding. In this mode, American variants of words are added to the stop list.

-d *word_list*
Use the specified word list instead of the default system word list. The word list must be sorted as specified above.

-h *spellhist*
Store misspelled words in the specified history file. The output of who -m is appended to the history file after the list of misspelled words.

-i Instruct deroff(1) to ignore ".so" and ".nx" commands.

-l Use **delatex** instead of deroff(1) if it is present on the system.

-m Enable support for common troff(1) macro packages; this option is passed verbatim to deroff(1). Refer to the **-m** description in deroff(1) for details.

-s *stop_list*
Use the specified stop list instead of the default system stop list. The stop list must be sorted as specified above.

-t Use **detex** instead of deroff(1) if it is present on the system.

-v Print all words not literally in the spelling list in addition to plausible derivations from spelling list words.

-x Print every plausible stem, prefixed with '='.

+*extra_list*
Use *extra_list* in addition to the default word list. The extra word list must be sorted as specified above.

FILES

/usr/share/dict/words	Default spelling list
/usr/share/dict/american	American spelling of certain words
/usr/share/dict/british	British spelling of certain words
/usr/share/dict/stop	Default stop list.
/usr/local/share/dict/words	Local spelling list (optional)
/usr/local/share/dict/stop	Local stop list (optional)
/usr/libexec/spellprog	Binary executed by the shell script /usr/bin/spell.

SEE ALSO

deroff(1), look(1), sed(1), sort(1), tee(1), troff(1)

HISTORY

The **spell** command appeared in Version 6 AT&T UNIX.

Unlike historic versions, the NetBSD **spell** command does not use hashed word files. Instead, it uses lexicographically sorted files and the same technique as look(1).

BUGS

The spelling list lacks many technical terms; new installations will probably wish to monitor the output for several months to gather local additions.

British spelling was done by an American.

In **-x** mode it would be nicer if the stems were grouped with the appropriate word.

NAME

split — split a file into pieces

SYNOPSIS

split [**-a** *suffix_length*][**-b** *byte_count*[k|m]| **-l** *line_count* **-n** *chunk_count*] [*file* [*name*]]

DESCRIPTION

The **split** utility reads the given *file* and breaks it up into files of 1000 lines each. If *file* is a single dash or absent, **split** reads from the standard input. *file* itself is not altered.

The options are as follows:

-a Use *suffix_length* letters to form the suffix of the file name.

-b Create smaller files *byte_count* bytes in length. If 'k' is appended to the number, the file is split into *byte_count* kilobyte pieces. If 'm' is appended to the number, the file is split into *byte_count* megabyte pieces.

-l Create smaller files *line_count* lines in length.

-n Split file into *chunk_count* smaller files.

If additional arguments are specified, the first is used as the name of the input file which is to be split. If a second additional argument is specified, it is used as a prefix for the names of the files into which the file is split. In this case, each file into which the file is split is named by the prefix followed by a lexically ordered suffix using *suffix_length* characters in the range "a-z". If **-a** is not specified, two letters are used as the suffix.

If the *name* argument is not specified, 'x' is used.

STANDARDS

The **split** utility conforms to IEEE Std 1003.1-2001 ("POSIX.1").

HISTORY

A **split** command appeared in Version 6 AT&T UNIX.

The **-a** option was introduced in NetBSD 2.0. Before that, if *name* was not specified, **split** would vary the first letter of the filename to increase the number of possible output files. The **-a** option makes this unnecessary.

NAME

srtconfig — configure srt interfaces

SYNOPSIS

srtconfig *srtX*
srtconfig *srtX N*
srtconfig *srtX* del *N*
srtconfig *srtX* add *srcaddr mask dstif dstaddr*
srtconfig *srtX* set *N srcaddr mask dstif dstaddr*

DESCRIPTION

srtconfig configures, or queries the configuration of, srt(4) interfaces. An srt(4) interface parcels packets out to other interfaces based on their source addresses (the normal routing mechanisms handle routing decisions based on destination addresses). An interface may have any number of routing choices; they are examined in order until one matching the packet is found. The packet is sent to the corresponding interface. (Any interface, even another **srt** interface, may be specified; if the configurations collaborate to cause a packet to loop forever, the system will lock up or crash.)

When run with only one argument, **srtconfig** prints the settings for the specified interface.

When run with two arguments, **srtconfig** prints the settings for the routing choice whose number is given as the second argument.

The form with 'del' deletes a routing choice, identified by its number. Other choices with higher numbers, if any, will be renumbered accordingly.

The 'add' form adds a choice; the other arguments describe it, and are documented below. The new choice is added at the end of the list.

The 'set' form replaces an existing choice, given its number. The other arguments describe the new choice which is to replace whatever currently exists at the given number *N*.

A choice is described by four pieces of information: a source address and mask, which are used to determine which choice an outgoing packet uses, a destination interface, and a destination address for the new interface. The source address and mask are specified like any Internet addresses (for convenience, the mask may instead be specified as a '/' followed by a small integer, CIDR-style; note that in this case the mask must still be a separate argument; it cannot be appended to the end of the source address argument).

Each srt interface also has ordinary source and destination addresses which are set with ifconfig(8) like any other interface; these should not be confused with any of the above.

AUTHORS

der Mouse ⟨mouse@rodents.montreal.qc.ca⟩

NAME

ssh-add — adds RSA or DSA identities to the authentication agent

SYNOPSIS

ssh-add [**-cDdLlXx**] [**-t** *life*] [*file ...*]
ssh-add -s *reader*
ssh-add -e *reader*

DESCRIPTION

ssh-add adds RSA or DSA identities to the authentication agent, ssh-agent(1). When run without arguments, it adds the files ~/.ssh/id_rsa, ~/.ssh/id_dsa and ~/.ssh/identity. Alternative file names can be given on the command line. If any file requires a passphrase, **ssh-add** asks for the passphrase from the user. The passphrase is read from the user's tty. **ssh-add** retries the last passphrase if multiple identity files are given.

The authentication agent must be running and the SSH_AUTH_SOCK environment variable must contain the name of its socket for **ssh-add** to work.

The options are as follows:

-c Indicates that added identities should be subject to confirmation before being used for authentication. Confirmation is performed by the SSH_ASKPASS program mentioned below. Successful confirmation is signaled by a zero exit status from the SSH_ASKPASS program, rather than text entered into the requester.

-D Deletes all identities from the agent.

-d Instead of adding identities, removes identities from the agent. If **ssh-add** has been run without arguments, the keys for the default identities will be removed. Otherwise, the argument list will be interpreted as a list of paths to public key files and matching keys will be removed from the agent. If no public key is found at a given path, **ssh-add** will append .pub and retry.

-e *reader*
Remove key in smartcard *reader*.

-L Lists public key parameters of all identities currently represented by the agent.

-l Lists fingerprints of all identities currently represented by the agent.

-s *reader*
Add key in smartcard *reader*.

-t *life*
Set a maximum lifetime when adding identities to an agent. The lifetime may be specified in seconds or in a time format specified in sshd_config(5).

-X Unlock the agent.

-x Lock the agent with a password.

ENVIRONMENT

DISPLAY and SSH_ASKPASS
If **ssh-add** needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If **ssh-add** does not have a terminal associated with it but DISPLAY and SSH_ASKPASS are set, it will execute the program specified by SSH_ASKPASS and open an X11 window to read the passphrase. This is particularly useful when calling **ssh-add** from a .xsession or related script. (Note that on some machines it may be necessary to redirect the input from /dev/null to make this work.)

`SSH_AUTH_SOCK`
Identifies the path of a unix-domain socket used to communicate with the agent.

FILES

`~/.ssh/identity`
Contains the protocol version 1 RSA authentication identity of the user.

`~/.ssh/id_dsa`
Contains the protocol version 2 DSA authentication identity of the user.

`~/.ssh/id_rsa`
Contains the protocol version 2 RSA authentication identity of the user.

Identity files should not be readable by anyone but the user. Note that **ssh-add** ignores identity files if they are accessible by others.

DIAGNOSTICS

Exit status is 0 on success, 1 if the specified command fails, and 2 if **ssh-add** is unable to contact the authentication agent.

SEE ALSO

ssh(1), ssh-agent(1), ssh-keygen(1), sshd(8)

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

NAME

ssh-agent — authentication agent

SYNOPSIS

ssh-agent [**-c** | **-s**] [**-d**] [**-a** *bind_address*] [**-t** *life*] [*command* [*arg ...*]]
ssh-agent [**-c** | **-s**] **-k**

DESCRIPTION

ssh-agent is a program to hold private keys used for public key authentication (RSA, DSA). The idea is that **ssh-agent** is started in the beginning of an X-session or a login session, and all other windows or programs are started as clients to the ssh-agent program. Through use of environment variables the agent can be located and automatically used for authentication when logging in to other machines using ssh(1).

The options are as follows:

-a *bind_address*
: Bind the agent to the unix-domain socket *bind_address*. The default is /tmp/ssh-XXXXXXXXXX/agent.<ppid>.

-c
: Generate C-shell commands on stdout. This is the default if SHELL looks like it's a csh style of shell.

-d
: Debug mode. When this option is specified **ssh-agent** will not fork.

-k
: Kill the current agent (given by the SSH_AGENT_PID environment variable).

-s
: Generate Bourne shell commands on stdout. This is the default if SHELL does not look like it's a csh style of shell.

-t *life*
: Set a default value for the maximum lifetime of identities added to the agent. The lifetime may be specified in seconds or in a time format specified in sshd_config(5). A lifetime specified for an identity with ssh-add(1) overrides this value. Without this option the default maximum lifetime is forever.

If a commandline is given, this is executed as a subprocess of the agent. When the command dies, so does the agent.

The agent initially does not have any private keys. Keys are added using ssh-add(1). When executed without arguments, ssh-add(1) adds the files ~/.ssh/id_rsa, ~/.ssh/id_dsa and ~/.ssh/identity. If the identity has a passphrase, ssh-add(1) asks for the passphrase (using a small X11 application if running under X11, or from the terminal if running without X). It then sends the identity to the agent. Several identities can be stored in the agent; the agent can automatically use any of these identities. **ssh-add -l** displays the identities currently held by the agent.

The idea is that the agent is run in the user's local PC, laptop, or terminal. Authentication data need not be stored on any other machine, and authentication passphrases never go over the network. However, the connection to the agent is forwarded over SSH remote logins, and the user can thus use the privileges given by the identities anywhere in the network in a secure way.

There are two main ways to get an agent set up: The first is that the agent starts a new subcommand into which some environment variables are exported, eg **ssh-agent xterm &**. The second is that the agent prints the needed shell commands (either sh(1) or csh(1) syntax can be generated) which can be evalled in the calling shell, eg **eval 'ssh-agent -s'** for Bourne-type shells such as sh(1) or ksh(1) and **eval 'ssh-agent -c'** for csh(1) and derivatives.

Later ssh(1) looks at these variables and uses them to establish a connection to the agent.

The agent will never send a private key over its request channel. Instead, operations that require a private key will be performed by the agent, and the result will be returned to the requester. This way, private keys are not exposed to clients using the agent.

A unix-domain socket is created and the name of this socket is stored in the SSH_AUTH_SOCK environment variable. The socket is made accessible only to the current user. This method is easily abused by root or another instance of the same user.

The SSH_AGENT_PID environment variable holds the agent's process ID.

The agent exits automatically when the command given on the command line terminates.

FILES

~/.ssh/identity
: Contains the protocol version 1 RSA authentication identity of the user.

~/.ssh/id_dsa
: Contains the protocol version 2 DSA authentication identity of the user.

~/.ssh/id_rsa
: Contains the protocol version 2 RSA authentication identity of the user.

/tmp/ssh-XXXXXXXXXX/agent.<ppid>
: Unix-domain sockets used to contain the connection to the authentication agent. These sockets should only be readable by the owner. The sockets should get automatically removed when the agent exits.

SEE ALSO

ssh(1), ssh-add(1), ssh-keygen(1), sshd(8)

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

NAME

ssh-keygen — authentication key generation, management and conversion

SYNOPSIS

ssh-keygen [**-q**] [**-b** *bits*] **-t** *type* [**-N** *new_passphrase*] [**-C** *comment*]
[**-f** *output_keyfile*]
ssh-keygen -p [**-P** *old_passphrase*] [**-N** *new_passphrase*] [**-f** *keyfile*]
ssh-keygen -i [**-f** *input_keyfile*]
ssh-keygen -e [**-f** *input_keyfile*]
ssh-keygen -y [**-f** *input_keyfile*]
ssh-keygen -c [**-P** *passphrase*] [**-C** *comment*] [**-f** *keyfile*]
ssh-keygen -l [**-f** *input_keyfile*]
ssh-keygen -B [**-f** *input_keyfile*]
ssh-keygen -D *reader*
ssh-keygen -F *hostname* [**-f** *known_hosts_file*]
ssh-keygen -H [**-f** *known_hosts_file*]
ssh-keygen -R *hostname* [**-f** *known_hosts_file*]
ssh-keygen -U *reader* [**-f** *input_keyfile*]
ssh-keygen -r *hostname* [**-f** *input_keyfile*] [**-g**]
ssh-keygen -G *output_file* [**-v**] [**-b** *bits*] [**-M** *memory*] [**-S** *start_point*]
ssh-keygen -T *output_file* **-f** *input_file* [**-v**] [**-a** *num_trials*] [**-W** *generator*]

DESCRIPTION

ssh-keygen generates, manages and converts authentication keys for ssh(1). **ssh-keygen** can create RSA keys for use by SSH protocol version 1 and RSA or DSA keys for use by SSH protocol version 2. The type of key to be generated is specified with the **-t** option. If invoked without any arguments, **ssh-keygen** will generate an RSA key for use in SSH protocol 2 connections.

ssh-keygen is also used to generate groups for use in Diffie-Hellman group exchange (DH-GEX). See the **MODULI GENERATION** section for details.

Normally each user wishing to use SSH with RSA or DSA authentication runs this once to create the authentication key in `~/.ssh/identity`, `~/.ssh/id_dsa` or `~/.ssh/id_rsa`. Additionally, the system administrator may use this to generate host keys, as seen in `/etc/rc.d/sshd`.

Normally this program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but ".pub" appended. The program also asks for a passphrase. The passphrase may be empty to indicate no passphrase (host keys must have an empty passphrase), or it may be a string of arbitrary length. A passphrase is similar to a password, except it can be a phrase with a series of words, punctuation, numbers, whitespace, or any string of characters you want. Good passphrases are 10-30 characters long, are not simple sentences or otherwise easily guessable (English prose has only 1-2 bits of entropy per character, and provides very bad passphrases), and contain a mix of upper and lowercase letters, numbers, and non-alphanumeric characters. The passphrase can be changed later by using the **-p** option.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, a new key must be generated and copied to the corresponding public key to other machines.

For RSA1 keys, there is also a comment field in the key file that is only for convenience to the user to help identify the key. The comment can tell what the key is for, or whatever is useful. The comment is initialized to "user@host" when the key is created, but can be changed using the **-c** option.

After a key is generated, instructions below detail where the keys should be placed to be activated.

The options are as follows:

-a *trials*
Specifies the number of primality tests to perform when screening DH-GEX candidates using the **-T** command.

-B Show the bubblebabble digest of specified private or public key file.

-b *bits*
Specifies the number of bits in the key to create. For RSA keys, the minimum size is 768 bits and the default is 2048 bits. Generally, 2048 bits is considered sufficient. DSA keys must be exactly 1024 bits as specified by FIPS 186-2.

-C *comment*
Provides a new comment.

-c Requests changing the comment in the private and public key files. This operation is only supported for RSA1 keys. The program will prompt for the file containing the private keys, for the passphrase if the key has one, and for the new comment.

-D *reader*
Download the RSA public key stored in the smartcard in *reader*.

-e This option will read a private or public OpenSSH key file and print the key in RFC 4716 SSH Public Key File Format to stdout. This option allows exporting keys for use by several commercial SSH implementations.

-F *hostname*
Search for the specified *hostname* in a known_hosts file, listing any occurrences found. This option is useful to find hashed host names or addresses and may also be used in conjunction with the **-H** option to print found keys in a hashed format.

-f *filename*
Specifies the filename of the key file.

-G *output_file*
Generate candidate primes for DH-GEX. These primes must be screened for safety (using the **-T** option) before use.

-g Use generic DNS format when printing fingerprint resource records using the **-r** command.

-H Hash a known_hosts file. This replaces all hostnames and addresses with hashed representations within the specified file; the original content is moved to a file with a .old suffix. These hashes may be used normally by **ssh** and **sshd**, but they do not reveal identifying information should the file's contents be disclosed. This option will not modify existing hashed hostnames and is therefore safe to use on files that mix hashed and non-hashed names.

-i This option will read an unencrypted private (or public) key file in SSH2-compatible format and print an OpenSSH compatible private (or public) key to stdout. **ssh-keygen** also reads the RFC 4716 SSH Public Key File Format. This option allows importing keys from several commercial SSH implementations.

-l Show fingerprint of specified public key file. Private RSA1 keys are also supported. For RSA and DSA keys **ssh-keygen** tries to find the matching public key file and prints its fingerprint.

-M *memory*
Specify the amount of memory to use (in megabytes) when generating candidate moduli for DH-GEX.

-N *new_passphrase*
Provides the new passphrase.

-P *passphrase*
Provides the (old) passphrase.

-p Requests changing the passphrase of a private key file instead of creating a new private key. The program will prompt for the file containing the private key, for the old passphrase, and twice for the new passphrase.

-q Silence **ssh-keygen**. Used by `/etc/rc.d/sshd` when creating a new key.

-R *hostname*
Removes all keys belonging to *hostname* from a `known_hosts` file. This option is useful to delete hashed hosts (see the **-H** option above).

-r *hostname*
Print the SSHFP fingerprint resource record named *hostname* for the specified public key file.

-S *start*
Specify start point (in hex) when generating candidate moduli for DH-GEX.

-T *output_file*
Test DH group exchange candidate primes (generated using the **-G** option) for safety.

-t *type*
Specifies the type of key to create. The possible values are "rsa1" for protocol version 1 and "rsa" or "dsa" for protocol version 2.

-U *reader*
Upload an existing RSA private key into the smartcard in *reader*.

-v Verbose mode. Causes **ssh-keygen** to print debugging messages about its progress. This is helpful for debugging moduli generation. Multiple **-v** options increase the verbosity. The maximum is 3.

-W *generator*
Specify desired generator when testing candidate moduli for DH-GEX.

-y This option will read a private OpenSSH format file and print an OpenSSH public key to stdout.

MODULI GENERATION

ssh-keygen may be used to generate groups for the Diffie-Hellman Group Exchange (DH-GEX) protocol. Generating these groups is a two-step process: first, candidate primes are generated using a fast, but memory intensive process. These candidate primes are then tested for suitability (a CPU-intensive process).

Generation of primes is performed using the **-G** option. The desired length of the primes may be specified by the **-b** option. For example:

```
# ssh-keygen -G moduli-2048.candidates -b 2048
```

By default, the search for primes begins at a random point in the desired length range. This may be overridden using the **-S** option, which specifies a different start point (in hex).

Once a set of candidates have been generated, they must be tested for suitability. This may be performed using the **-T** option. In this mode **ssh-keygen** will read candidates from standard input (or a file specified using the **-f** option). For example:

```
# ssh-keygen -T moduli-2048 -f moduli-2048.candidates
```

By default, each candidate will be subjected to 100 primality tests. This may be overridden using the **-a** option. The DH generator value will be chosen automatically for the prime under consideration. If a specific generator is desired, it may be requested using the **-W** option. Valid generator values are 2, 3, and 5.

Screened DH groups may be installed in `/etc/moduli`. It is important that this file contains moduli of a range of bit lengths and that both ends of a connection share common moduli.

FILES

`~/.ssh/identity`
: Contains the protocol version 1 RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **ssh-keygen** but it is offered as the default file for the private key. `ssh`(1) will read this file when a login attempt is made.

`~/.ssh/identity.pub`
: Contains the protocol version 1 RSA public key for authentication. The contents of this file should be added to `~/.ssh/authorized_keys` on all machines where the user wishes to log in using RSA authentication. There is no need to keep the contents of this file secret.

`~/.ssh/id_dsa`
: Contains the protocol version 2 DSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **ssh-keygen** but it is offered as the default file for the private key. `ssh`(1) will read this file when a login attempt is made.

`~/.ssh/id_dsa.pub`
: Contains the protocol version 2 DSA public key for authentication. The contents of this file should be added to `~/.ssh/authorized_keys` on all machines where the user wishes to log in using public key authentication. There is no need to keep the contents of this file secret.

`~/.ssh/id_rsa`
: Contains the protocol version 2 RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file using 3DES. This file is not automatically accessed by **ssh-keygen** but it is offered as the default file for the private key. `ssh`(1) will read this file when a login attempt is made.

`~/.ssh/id_rsa.pub`
: Contains the protocol version 2 RSA public key for authentication. The contents of this file should be added to `~/.ssh/authorized_keys` on all machines where the user wishes to log in using public key authentication. There is no need to keep the contents of this file secret.

`/etc/moduli`
: Contains Diffie-Hellman groups used for DH-GEX. The file format is described in `moduli`(5).

SEE ALSO

`ssh`(1), `ssh-add`(1), `ssh-agent`(1), `moduli`(5), `sshd`(8)

The Secure Shell (SSH) Public Key File Format, RFC 4716, 2006.

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

NAME

ssh-keyscan — gather ssh public keys

SYNOPSIS

ssh-keyscan [**-46Hv**] [**-f** *file*] [**-p** *port*] [**-T** *timeout*] [**-t** *type*]
[*host* | *addrlist namelist*] [*. . .*]

DESCRIPTION

ssh-keyscan is a utility for gathering the public ssh host keys of a number of hosts. It was designed to aid in building and verifying `ssh_known_hosts` files. **ssh-keyscan** provides a minimal interface suitable for use by shell and perl scripts.

ssh-keyscan uses non-blocking socket I/O to contact as many hosts as possible in parallel, so it is very efficient. The keys from a domain of 1,000 hosts can be collected in tens of seconds, even when some of those hosts are down or do not run ssh. For scanning, one does not need login access to the machines that are being scanned, nor does the scanning process involve any encryption.

The options are as follows:

-4 Forces **ssh-keyscan** to use IPv4 addresses only.

-6 Forces **ssh-keyscan** to use IPv6 addresses only.

-f *file*
Read hosts or `addrlist namelist` pairs from this file, one per line. If `-` is supplied instead of a filename, **ssh-keyscan** will read hosts or `addrlist namelist` pairs from the standard input.

-H Hash all hostnames and addresses in the output. Hashed names may be used normally by **ssh** and **sshd**, but they do not reveal identifying information should the file's contents be disclosed.

-p *port*
Port to connect to on the remote host.

-T *timeout*
Set the timeout for connection attempts. If `timeout` seconds have elapsed since a connection was initiated to a host or since the last time anything was read from that host, then the connection is closed and the host in question considered unavailable. Default is 5 seconds.

-t *type*
Specifies the type of the key to fetch from the scanned hosts. The possible values are "rsa1" for protocol version 1 and "rsa" or "dsa" for protocol version 2. Multiple values may be specified by separating them with commas. The default is "rsa1".

-v Verbose mode. Causes **ssh-keyscan** to print debugging messages about its progress.

SECURITY

If an ssh_known_hosts file is constructed using **ssh-keyscan** without verifying the keys, users will be vulnerable to *man in the middle* attacks. On the other hand, if the security model allows such a risk, **ssh-keyscan** can help in the detection of tampered keyfiles or man in the middle attacks which have begun after the ssh_known_hosts file was created.

FILES

`Input format:`

```
1.2.3.4,1.2.4.4 name.my.domain,name,n.my.domain,n,1.2.3.4,1.2.4.4
```

```
Output format for rsa1 keys:

host-or-namelist bits exponent modulus

Output format for rsa and dsa keys:

host-or-namelist keytype base64-encoded-key
```

Where `keytype` is either “ssh-rsa” or “ssh-dss”.

`/etc/ssh/ssh_known_hosts`

EXAMPLES

Print the `rsa1` host key for machine `hostname`:

```
$ ssh-keyscan hostname
```

Find all hosts from the file `ssh_hosts` which have new or different keys from those in the sorted file `ssh_known_hosts`:

```
$ ssh-keyscan -t rsa,dsa -f ssh_hosts | \
        sort -u - ssh_known_hosts | diff ssh_known_hosts -
```

SEE ALSO

`ssh`(1), `sshd`(8)

AUTHORS

David Mazieres ⟨dm@lcs.mit.edu⟩ wrote the initial version, and Wayne Davison ⟨wayned@users.sourceforge.net⟩ added support for protocol version 2.

BUGS

It generates "Connection closed by remote host" messages on the consoles of all the machines it scans if the server is older than version 2.9. This is because it opens a connection to the ssh port, reads the public key, and drops the connection as soon as it gets the key.

NAME

ssh — OpenSSH SSH client (remote login program)

SYNOPSIS

ssh [**-1246AaCfgKkMNnqsTtVvXxY**] [**-b** *bind_address*] [**-c** *cipher_spec*] [**-D** [*bind_address*:]*port*] [**-e** *escape_char*] [**-F** *configfile*] [**-i** *identity_file*] [**-L** [*bind_address*:]*port*:*host*:*hostport*] [**-l** *login_name*] [**-m** *mac_spec*] [**-O** *ctl_cmd*] [**-o** *option*] [**-p** *port*] [**-R** [*bind_address*:]*port*:*host*:*hostport*] [**-S** *ctl_path*] [**-w** *local_tun*[:*remote_tun*]] [*user*@]*hostname* [*command*]

DESCRIPTION

ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified *hostname* (with optional *user* name). The user must prove his/her identity to the remote machine using one of several methods depending on the protocol version used (see below).

If *command* is specified, it is executed on the remote host instead of a login shell.

The options are as follows:

-1 Forces **ssh** to try protocol version 1 only.

-2 Forces **ssh** to try protocol version 2 only.

-4 Forces **ssh** to use IPv4 addresses only.

-6 Forces **ssh** to use IPv6 addresses only.

-A Enables forwarding of the authentication agent connection. This can also be specified on a per-host basis in a configuration file.

Agent forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the agent's Unix-domain socket) can access the local agent through the forwarded connection. An attacker cannot obtain key material from the agent, however they can perform operations on the keys that enable them to authenticate using the identities loaded into the agent.

-a Disables forwarding of the authentication agent connection.

-b *bind_address*
Use *bind_address* on the local machine as the source address of the connection. Only useful on systems with more than one address.

-C Requests compression of all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP connections). The compression algorithm is the same used by gzip(1), and the "level" can be controlled by the **CompressionLevel** option for protocol version 1. Compression is desirable on modem lines and other slow connections, but will only slow down things on fast networks. The default value can be set on a host-by-host basis in the configuration files; see the **Compression** option.

-c *cipher_spec*
Selects the cipher specification for encrypting the session.

Protocol version 1 allows specification of a single cipher. The supported values are “3des”, “blowfish”, and “des”. *3des* (triple-des) is an encrypt-decrypt-encrypt triple with three different keys. It is believed to be secure. *blowfish* is a fast block cipher; it appears very secure and is much faster than *3des*. *des* is only supported in the **ssh** client for interoperability with legacy protocol 1 implementations that do not support the *3des* cipher. Its use is strongly discouraged due to cryptographic weaknesses. The default is “3des”.

For protocol version 2, *cipher_spec* is a comma-separated list of ciphers listed in order of preference. The supported ciphers are: 3des-cbc, aes128-cbc, aes192-cbc, aes256-cbc, aes128-ctr, aes192-ctr, aes256-ctr, arcfour128, arcfour256, arcfour, blowfish-cbc, and cast128-cbc. The default is:

```
aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour128,
arcfour256,arcfour,aes192-cbc,aes256-cbc,aes128-ctr,
aes192-ctr,aes256-ctr
```

-D [*bind_address*:]*port*

Specifies a local “dynamic” application-level port forwarding. This works by allocating a socket to listen to *port* on the local side, optionally bound to the specified *bind_address*. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine. Currently the SOCKS4 and SOCKS5 protocols are supported, and **ssh** will act as a SOCKS server. Only root can forward privileged ports. Dynamic port forwardings can also be specified in the configuration file.

IPv6 addresses can be specified with an alternative syntax: [*bind_address*/]*port* or by enclosing the address in square brackets. Only the superuser can forward privileged ports. By default, the local port is bound in accordance with the **GatewayPorts** setting. However, an explicit *bind_address* may be used to bind the connection to a specific address. The *bind_address* of “localhost” indicates that the listening port be bound for local use only, while an empty address or ‘*’ indicates that the port should be available from all interfaces.

-e *escape_char*

Sets the escape character for sessions with a pty (default: ‘~’). The escape character is only recognized at the beginning of a line. The escape character followed by a dot (‘.’) closes the connection; followed by control-Z suspends the connection; and followed by itself sends the escape character once. Setting the character to “none” disables any escapes and makes the session fully transparent.

-F *configfile*

Specifies an alternative per-user configuration file. If a configuration file is given on the command line, the system-wide configuration file (/etc/ssh/ssh_config) will be ignored. The default for the per-user configuration file is ~/.ssh/config.

-f Requests **ssh** to go to background just before command execution. This is useful if **ssh** is going to ask for passwords or passphrases, but the user wants it in the background. This implies **-n**. The recommended way to start X11 programs at a remote site is with something like **ssh -f host xterm**.

-g Allows remote hosts to connect to local forwarded ports.

-I *smartcard_device*

Specify the device **ssh** should use to communicate with a smartcard used for storing the user’s private RSA key. This option is only available if support for smartcard devices is compiled in (default is no support).

-i *identity_file*
Selects a file from which the identity (private key) for RSA or DSA authentication is read. The default is `~/.ssh/identity` for protocol version 1, and `~/.ssh/id_rsa` and `~/.ssh/id_dsa` for protocol version 2. Identity files may also be specified on a per-host basis in the configuration file. It is possible to have multiple **-i** options (and multiple identities specified in configuration files).

-K Enables GSSAPI-based authentication and forwarding (delegation) of GSSAPI credentials to the server.

-k Disables forwarding (delegation) of GSSAPI credentials to the server.

-L [*bind_address*:]*port*:*host*:*hostport*
Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to *port* on the local side, optionally bound to the specified *bind_address*. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to *host* port *hostport* from the remote machine. Port forwardings can also be specified in the configuration file. IPv6 addresses can be specified with an alternative syntax: [*bind_address*/]*port*/*host*/*hostport* or by enclosing the address in square brackets. Only the superuser can forward privileged ports. By default, the local port is bound in accordance with the **GatewayPorts** setting. However, an explicit *bind_address* may be used to bind the connection to a specific address. The *bind_address* of "localhost" indicates that the listening port be bound for local use only, while an empty address or '*' indicates that the port should be available from all interfaces.

-l *login_name*
Specifies the user to log in as on the remote machine. This also may be specified on a per-host basis in the configuration file.

-M Places the **ssh** client into "master" mode for connection sharing. Multiple **-M** options places **ssh** into "master" mode with confirmation required before slave connections are accepted. Refer to the description of **ControlMaster** in `ssh_config`(5) for details.

-m *mac_spec*
Additionally, for protocol version 2 a comma-separated list of MAC (message authentication code) algorithms can be specified in order of preference. See the **MACs** keyword for more information.

-N Do not execute a remote command. This is useful for just forwarding ports (protocol version 2 only).

-n Redirects stdin from `/dev/null` (actually, prevents reading from stdin). This must be used when **ssh** is run in the background. A common trick is to use this to run X11 programs on a remote machine. For example, **ssh -n shadows.cs.hut.fi emacs &** will start an emacs on shadows.cs.hut.fi, and the X11 connection will be automatically forwarded over an encrypted channel. The **ssh** program will be put in the background. (This does not work if **ssh** needs to ask for a password or passphrase; see also the **-f** option.)

-O *ctl_cmd*
Control an active connection multiplexing master process. When the **-O** option is specified, the *ctl_cmd* argument is interpreted and passed to the master process. Valid commands are: "check" (check that the master process is running) and "exit" (request the master to exit).

-o *option*
Can be used to give options in the format used in the configuration file. This is useful for specifying options for which there is no separate command-line flag. For full details of the options listed below, and their possible values, see `ssh_config`(5).

AddressFamily
BatchMode
BindAddress
ChallengeResponseAuthentication
CheckHostIP
Cipher
Ciphers
ClearAllForwardings
Compression
CompressionLevel
ConnectionAttempts
ConnectTimeout
ControlMaster
ControlPath
DynamicForward
EscapeChar
ExitOnForwardFailure
ForwardAgent
ForwardX11
ForwardX11Trusted
GatewayPorts
GlobalKnownHostsFile
GSSAPIAuthentication
GSSAPIDelegateCredentials
HashKnownHosts
Host
HostbasedAuthentication
HostKeyAlgorithms
HostKeyAlias
HostName
IdentityFile
IdentitiesOnly
KbdInteractiveDevices
LocalCommand
LocalForward
LogLevel
MACs
NoHostAuthenticationForLocalhost
NumberOfPasswordPrompts
PasswordAuthentication
PermitLocalCommand
Port
PreferredAuthentications
Protocol
ProxyCommand
PubkeyAuthentication
RekeyLimit
RemoteForward
RhostsRSAAuthentication

RSAAuthentication
SendEnv
ServerAliveInterval
ServerAliveCountMax
SmartcardDevice
StrictHostKeyChecking
TCPKeepAlive
Tunnel
TunnelDevice
UsePrivilegedPort
User
UserKnownHostsFile
VerifyHostKeyDNS
XAuthLocation

-p *port*
Port to connect to on the remote host. This can be specified on a per-host basis in the configuration file.

-q Quiet mode. Causes most warning and diagnostic messages to be suppressed.

-R [*bind_address*:]*port*:*host*:*hostport*
Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to *port* on the remote side, and whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to *host* port *hostport* from the local machine.

Port forwardings can also be specified in the configuration file. Privileged ports can be forwarded only when logging in as root on the remote machine. IPv6 addresses can be specified by enclosing the address in square braces or using an alternative syntax: [*bind_address*/]*host*/*port*/*hostport*.

By default, the listening socket on the server will be bound to the loopback interface only. This may be overriden by specifying a *bind_address*. An empty *bind_address*, or the address '*', indicates that the remote socket should listen on all interfaces. Specifying a remote *bind_address* will only succeed if the server's **GatewayPorts** option is enabled (see sshd_config(5)).

-S *ctl_path*
Specifies the location of a control socket for connection sharing. Refer to the description of **ControlPath** and **ControlMaster** in ssh_config(5) for details.

-s May be used to request invocation of a subsystem on the remote system. Subsystems are a feature of the SSH2 protocol which facilitate the use of SSH as a secure transport for other applications (eg. sftp(1)). The subsystem is specified as the remote command.

-T Disable pseudo-tty allocation.

-t Force pseudo-tty allocation. This can be used to execute arbitrary screen-based programs on a remote machine, which can be very useful, e.g. when implementing menu services. Multiple **-t** options force tty allocation, even if **ssh** has no local tty.

-V Display the version number and exit.

-v Verbose mode. Causes **ssh** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Multiple **-v** options increase the verbosity. The maximum is 3.

-w *local_tun*[:*remote_tun*]
Requests tunnel device forwarding with the specified tun(4) devices between the client (*local_tun*) and the server (*remote_tun*).

The devices may be specified by numerical ID or the keyword "any", which uses the next available tunnel device. If *remote_tun* is not specified, it defaults to "any". See also the **Tunnel** and **TunnelDevice** directives in ssh_config(5). If the **Tunnel** directive is unset, it is set to the default tunnel mode, which is "point-to-point".

-X Enables X11 forwarding. This can also be specified on a per-host basis in a configuration file.

X11 forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the user's X authorization database) can access the local X11 display through the forwarded connection. An attacker may then be able to perform activities such as keystroke monitoring.

For this reason, X11 forwarding is subjected to X11 SECURITY extension restrictions by default. Please refer to the **ssh -Y** option and the **ForwardX11Trusted** directive in ssh_config(5) for more information.

-x Disables X11 forwarding.

-Y Enables trusted X11 forwarding. Trusted X11 forwardings are not subjected to the X11 SECURITY extension controls.

ssh may additionally obtain configuration data from a per-user configuration file and a system-wide configuration file. The file format and configuration options are described in ssh_config(5).

ssh exits with the exit status of the remote command or with 255 if an error occurred.

AUTHENTICATION

The OpenSSH SSH client supports SSH protocols 1 and 2. Protocol 2 is the default, with **ssh** falling back to protocol 1 if it detects protocol 2 is unsupported. These settings may be altered using the **Protocol** option in ssh_config(5), or enforced using the **-1** and **-2** options (see above). Both protocols support similar authentication methods, but protocol 2 is preferred since it provides additional mechanisms for confidentiality (the traffic is encrypted using AES, 3DES, Blowfish, CAST128, or Arcfour) and integrity (hmac-md5, hmac-sha1, umac-64, hmac-ripemd160). Protocol 1 lacks a strong mechanism for ensuring the integrity of the connection.

The methods available for authentication are: GSSAPI-based authentication, host-based authentication, public key authentication, challenge-response authentication, and password authentication. Authentication methods are tried in the order specified above, though protocol 2 has a configuration option to change the default order: **PreferredAuthentications**.

Host-based authentication works as follows: If the machine the user logs in from is listed in /etc/hosts.equiv or /etc/shosts.equiv on the remote machine, and the user names are the same on both sides, or if the files ~/.rhosts or ~/.shosts exist in the user's home directory on the remote machine and contain a line containing the name of the client machine and the name of the user on that machine, the user is considered for login. Additionally, the server *must* be able to verify the client's host key (see the description of /etc/ssh/ssh_known_hosts and ~/.ssh/known_hosts, below) for login to be permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing. [Note to the administrator: /etc/hosts.equiv, ~/.rhosts, and the rlogin/rsh protocol in general, are inherently insecure and should be disabled if security is desired.]

Public key authentication works as follows: The scheme is based on public-key cryptography, using cryptosystems where encryption and decryption are done using separate keys, and it is unfeasible to derive the decryption key from the encryption key. The idea is that each user creates a public/private key pair for

authentication purposes. The server knows the public key, and only the user knows the private key. **ssh** implements public key authentication protocol automatically, using either the RSA or DSA algorithms. Protocol 1 is restricted to using only RSA keys, but protocol 2 may use either. The **HISTORY** section of ssl(8) contains a brief discussion of the two algorithms.

The file ~/.ssh/authorized_keys lists the public keys that are permitted for logging in. When the user logs in, the **ssh** program tells the server which key pair it would like to use for authentication. The client proves that it has access to the private key and the server checks that the corresponding public key is authorized to accept the account.

The user creates his/her key pair by running ssh-keygen(1). This stores the private key in ~/.ssh/identity (protocol 1), ~/.ssh/id_dsa (protocol 2 DSA), or ~/.ssh/id_rsa (protocol 2 RSA) and stores the public key in ~/.ssh/identity.pub (protocol 1), ~/.ssh/id_dsa.pub (protocol 2 DSA), or ~/.ssh/id_rsa.pub (protocol 2 RSA) in the user's home directory. The user should then copy the public key to ~/.ssh/authorized_keys in his/her home directory on the remote machine. The authorized_keys file corresponds to the conventional ~/.rhosts file, and has one key per line, though the lines can be very long. After this, the user can log in without giving the password.

The most convenient way to use public key authentication may be with an authentication agent. See ssh-agent(1) for more information.

Challenge-response authentication works as follows: The server sends an arbitrary "challenge" text, and prompts for a response. Protocol 2 allows multiple challenges and responses; protocol 1 is restricted to just one challenge/response. Examples of challenge-response authentication include BSD Authentication (see login.conf(5)) and PAM (some non-OpenBSD systems).

Finally, if other authentication methods fail, **ssh** prompts the user for a password. The password is sent to the remote host for checking; however, since all communications are encrypted, the password cannot be seen by someone listening on the network.

ssh automatically maintains and checks a database containing identification for all hosts it has ever been used with. Host keys are stored in ~/.ssh/known_hosts in the user's home directory. Additionally, the file /etc/ssh/ssh_known_hosts is automatically checked for known hosts. Any new hosts are automatically added to the user's file. If a host's identification ever changes, **ssh** warns about this and disables password authentication to prevent server spoofing or man-in-the-middle attacks, which could otherwise be used to circumvent the encryption. The **StrictHostKeyChecking** option can be used to control logins to machines whose host key is not known or has changed.

When the user's identity has been accepted by the server, the server either executes the given command, or logs into the machine and gives the user a normal shell on the remote machine. All communication with the remote command or shell will be automatically encrypted.

If a pseudo-terminal has been allocated (normal login session), the user may use the escape characters noted below.

If no pseudo-tty has been allocated, the session is transparent and can be used to reliably transfer binary data. On most systems, setting the escape character to "none" will also make the session transparent even if a tty is used.

The session terminates when the command or shell on the remote machine exits and all X11 and TCP connections have been closed.

ESCAPE CHARACTERS

When a pseudo-terminal has been requested, **ssh** supports a number of functions through the use of an escape character.

A single tilde character can be sent as **~~** or by following the tilde by a character other than those described below. The escape character must always follow a newline to be interpreted as special. The escape character can be changed in configuration files using the **EscapeChar** configuration directive or on the command line by the **-e** option.

The supported escapes (assuming the default '~') are:

~. Disconnect.

~^Z Background **ssh**.

~# List forwarded connections.

~& Background **ssh** at logout when waiting for forwarded connection / X11 sessions to terminate.

~? Display a list of escape characters.

~B Send a BREAK to the remote system (only useful for SSH protocol version 2 and if the peer supports it).

~C Open command line. Currently this allows the addition of port forwardings using the **-L** and **-R** options (see above). It also allows the cancellation of existing remote port-forwardings using **-KR**[*bind_address*:]*port*. **!***command* allows the user to execute a local command if the **PermitLocalCommand** option is enabled in ssh_config(5). Basic help is available, using the **-h** option.

~R Request rekeying of the connection (only useful for SSH protocol version 2 and if the peer supports it).

TCP FORWARDING

Forwarding of arbitrary TCP connections over the secure channel can be specified either on the command line or in a configuration file. One possible application of TCP forwarding is a secure connection to a mail server; another is going through firewalls.

In the example below, we look at encrypting communication between an IRC client and server, even though the IRC server does not directly support encrypted communications. This works as follows: the user connects to the remote host using **ssh**, specifying a port to be used to forward connections to the remote server. After that it is possible to start the service which is to be encrypted on the client machine, connecting to the same local port, and **ssh** will encrypt and forward the connection.

The following example tunnels an IRC session from client machine "127.0.0.1" (localhost) to remote server "server.example.com":

```
$ ssh -f -L 1234:localhost:6667 server.example.com sleep 10
$ irc -c '#users' -p 1234 pinky 127.0.0.1
```

This tunnels a connection to IRC server "server.example.com", joining channel "#users", nickname "pinky", using port 1234. It doesn't matter which port is used, as long as it's greater than 1023 (remember, only root can open sockets on privileged ports) and doesn't conflict with any ports already in use. The connection is forwarded to port 6667 on the remote server, since that's the standard port for IRC services.

The **-f** option backgrounds **ssh** and the remote command "sleep 10" is specified to allow an amount of time (10 seconds, in the example) to start the service which is to be tunnelled. If no connections are made within the time specified, **ssh** will exit.

X11 FORWARDING

If the **ForwardX11** variable is set to "yes" (or see the description of the **-X**, **-x**, and **-Y** options above) and the user is using X11 (the DISPLAY environment variable is set), the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or

command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set DISPLAY. Forwarding of X11 connections can be configured on the command line or in configuration files.

The DISPLAY value set by **ssh** will point to the server machine, but with a display number greater than zero. This is normal, and happens because **ssh** creates a "proxy" X server on the server machine for forwarding the connections over the encrypted channel.

ssh will also automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie, store it in Xauthority on the server, and verify that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

If the **ForwardAgent** variable is set to "yes" (or see the description of the **-A** and **-a** options above) and the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side.

VERIFYING HOST KEYS

When connecting to a server for the first time, a fingerprint of the server's public key is presented to the user (unless the option **StrictHostKeyChecking** has been disabled). Fingerprints can be determined using ssh-keygen(1):

```
$ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
```

If the fingerprint is already known, it can be matched and verified, and the key can be accepted. If the fingerprint is unknown, an alternative method of verification is available: SSH fingerprints verified by DNS. An additional resource record (RR), SSHFP, is added to a zonefile and the connecting client is able to match the fingerprint with that of the key presented.

In this example, we are connecting a client to a server, "host.example.com". The SSHFP resource records should first be added to the zonefile for host.example.com:

```
$ ssh-keygen -r host.example.com.
```

The output lines will have to be added to the zonefile. To check that the zone is answering fingerprint queries:

```
$ dig -t SSHFP host.example.com
```

Finally the client connects:

```
$ ssh -o "VerifyHostKeyDNS ask" host.example.com
[...]
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
```

See the **VerifyHostKeyDNS** option in ssh_config(5) for more information.

SSH-BASED VIRTUAL PRIVATE NETWORKS

ssh contains support for Virtual Private Network (VPN) tunnelling using the tun(4) network pseudo-device, allowing two networks to be joined securely. The sshd_config(5) configuration option **PermitTunnel** controls whether the server supports this, and at what level (layer 2 or 3 traffic).

The following example would connect client network 10.0.50.0/24 with remote network 10.0.99.0/24 using a point-to-point connection from 10.1.1.1 to 10.1.1.2, provided that the SSH server running on the gateway to the remote network, at 192.168.1.15, allows it.

On the client:

```
# ssh -f -w 0:1 192.168.1.15 true
# ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
# route add 10.0.99.0/24 10.1.1.2
```

On the server:

```
# ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
# route add 10.0.50.0/24 10.1.1.1
```

Client access may be more finely tuned via the `/root/.ssh/authorized_keys` file (see below) and the **`PermitRootLogin`** server option. The following entry would permit connections on `tun`(4) device 1 from user "jane" and on tun device 2 from user "john", if **`PermitRootLogin`** is set to "forced-commands-only":

```
tunnel="1",command="sh /etc/netstart tun1" ssh-rsa ... jane
tunnel="2",command="sh /etc/netstart tun2" ssh-rsa ... john
```

Since an SSH-based setup entails a fair amount of overhead, it may be more suited to temporary setups, such as for wireless VPNs. More permanent VPNs are better provided by tools such as `ipsecctl`(8) and `isakmpd`(8).

ENVIRONMENT

`ssh` will normally set the following environment variables:

`DISPLAY` — The `DISPLAY` variable indicates the location of the X11 server. It is automatically set by **`ssh`** to point to a value of the form "hostname:n", where "hostname" indicates the host where the shell runs, and 'n' is an integer ≥ 1. **`ssh`** uses this special value to forward X11 connections over the secure channel. The user should normally not set `DISPLAY` explicitly, as that will render the X11 connection insecure (and will require the user to manually copy any required authorization cookies).

`HOME` — Set to the path of the user's home directory.

`LOGNAME` — Synonym for `USER`; set for compatibility with systems that use this variable.

`MAIL` — Set to the path of the user's mailbox.

`PATH` — Set to the default `PATH`, as specified when compiling **`ssh`**.

`SSH_ASKPASS` — If **`ssh`** needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If **`ssh`** does not have a terminal associated with it but `DISPLAY` and `SSH_ASKPASS` are set, it will execute the program specified by `SSH_ASKPASS` and open an X11 window to read the passphrase. This is particularly useful when calling **`ssh`** from a `.xsession` or related script. (Note that on some machines it may be necessary to redirect the input from `/dev/null` to make this work.)

`SSH_AUTH_SOCK` — Identifies the path of a UNIX-domain socket used to communicate with the agent.

`SSH_CONNECTION` — Identifies the client and server ends of the connection. The variable contains four space-separated values: client IP address, client port number, server IP address, and server port number.

`SSH_ORIGINAL_COMMAND` This variable contains the original command line if a forced command is executed. It can be used to extract the original arguments.

`SSH_TTY` This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

`TZ` This variable is set to indicate the present time zone if it was set when the daemon was started (i.e. the daemon passes the value on to new connections).

`USER` Set to the name of the user logging in.

Additionally, **ssh** reads `~/.ssh/environment`, and adds lines of the format "VARNAME=value" to the environment if the file exists and users are allowed to change their environment. For more information, see the **PermitUserEnvironment** option in `sshd_config`(5).

FILES

~/.rhosts
This file is used for host-based authentication (see above). On some machines this file may need to be world-readable if the user's home directory is on an NFS partition, because `sshd`(8) reads it as root. Additionally, this file must be owned by the user, and must not have write permissions for anyone else. The recommended permission for most machines is read/write for the user, and not accessible by others.

~/.shosts
This file is used in exactly the same way as `.rhosts`, but allows host-based authentication without permitting login with rlogin/rsh.

~/.ssh/ This directory is the default location for all user-specific configuration and authentication information. There is no general requirement to keep the entire contents of this directory secret, but the recommended permissions are read/write/execute for the user, and not accessible by others.

~/.ssh/authorized_keys
Lists the public keys (RSA/DSA) that can be used for logging in as this user. The format of this file is described in the `sshd`(8) manual page. This file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

~/.ssh/config
This is the per-user configuration file. The file format and configuration options are described in `ssh_config`(5). Because of the potential for abuse, this file must have strict permissions: read/write for the user, and not accessible by others.

~/.ssh/environment
Contains additional definitions for environment variables; see **ENVIRONMENT**, above.

~/.ssh/identity
~/.ssh/id_dsa
~/.ssh/id_rsa
Contains the private key for authentication. These files contain sensitive data and should be readable by the user but not accessible by others (read/write/execute). **ssh** will simply ignore a private key file if it is accessible by others. It is possible to specify a passphrase when generating the key which will be used to encrypt the sensitive part of this file using 3DES.

~/.ssh/identity.pub
~/.ssh/id_dsa.pub

~/.ssh/id_rsa.pub
: Contains the public key for authentication. These files are not sensitive and can (but need not) be readable by anyone.

~/.ssh/known_hosts
: Contains a list of host keys for all hosts the user has logged into that are not already in the systemwide list of known host keys. See `sshd`(8) for further details of the format of this file.

~/.ssh/rc
: Commands in this file are executed by **ssh** when the user logs in, just before the user's shell (or command) is started. See the `sshd`(8) manual page for more information.

/etc/hosts.equiv
: This file is for host-based authentication (see above). It should only be writable by root.

/etc/shosts.equiv
: This file is used in exactly the same way as `hosts.equiv`, but allows host-based authentication without permitting login with rlogin/rsh.

`/etc/ssh/ssh_config`
: Systemwide configuration file. The file format and configuration options are described in `ssh_config`(5).

/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_rsa_key
: These three files contain the private parts of the host keys and are used for host-based authentication. If protocol version 1 is used, **ssh** must be setuid root, since the host key is readable only by root. For protocol version 2, **ssh** uses `ssh-keysign`(8) to access the host keys, eliminating the requirement that **ssh** be setuid root when host-based authentication is used. By default **ssh** is not setuid root.

/etc/ssh/ssh_known_hosts
: Systemwide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. It should be world-readable. See `sshd`(8) for further details of the format of this file.

/etc/ssh/sshrc
: Commands in this file are executed by **ssh** when the user logs in, just before the user's shell (or command) is started. See the `sshd`(8) manual page for more information.

SEE ALSO

`scp`(1), `sftp`(1), `ssh-add`(1), `ssh-agent`(1), `ssh-keygen`(1), `ssh-keyscan`(1), `tun`(4), `hosts.equiv`(5), `ssh_config`(5), `ssh-keysign`(8), `sshd`(8)

The Secure Shell (SSH) Protocol Assigned Numbers, RFC 4250, 2006.

The Secure Shell (SSH) Protocol Architecture, RFC 4251, 2006.

The Secure Shell (SSH) Authentication Protocol, RFC 4252, 2006.

The Secure Shell (SSH) Transport Layer Protocol, RFC 4253, 2006.

The Secure Shell (SSH) Connection Protocol, RFC 4254, 2006.

Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints, RFC 4255, 2006.

Generic Message Exchange Authentication for the Secure Shell Protocol (SSH), RFC 4256, 2006.

The Secure Shell (SSH) Session Channel Break Extension, RFC 4335, 2006.

The Secure Shell (SSH) Transport Layer Encryption Modes, RFC 4344, 2006.

Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol, RFC 4345, 2006.

Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol, RFC 4419, 2006.

The Secure Shell (SSH) Public Key File Format, RFC 4716, 2006.

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

NAME

stat, **readlink** — display file status

SYNOPSIS

stat [**-FLnq**][**-f** *format* | **-l** | **-r** | **-s** | **-x**][**-t** *timefmt*][*file ...*]
readlink [**-fn**][*file ...*]

DESCRIPTION

The **stat** utility displays information about the file pointed to by *file*. Read, write, or execute permissions of the named file are not required, but all directories listed in the pathname leading to the file must be searchable. If no argument is given, **stat** displays information about the file descriptor for standard input.

When invoked as **readlink**, only the target of the symbolic link is printed. If the given argument is not a symbolic link and the **-f** option is not specified, **readlink** will print nothing and exit with an error. If the **-f** option is specified, the output is canonicalized by following every symlink in every component of the given path recursively. **readlink** will resolve both absolute and relative paths, and return the absolute pathname corresponding to *file*. In this case, the argument does not need to be a symbolic link.

The information displayed is obtained by calling lstat(2) with the given argument and evaluating the returned structure. The default format displays the *st_dev*, *st_ino*, *st_mode*, *st_nlink*, *st_uid*, *st_gid*, *st_rdev*, *st_size*, *st_atime*, *st_mtime*, *st_ctime*, *st_blksize*, *st_blocks*, and *st_flags* fields, in that order.

The options are as follows:

-F As in ls(1), display a slash (‘/’) immediately after each pathname that is a directory, an asterisk (‘*’) after each that is executable, an at sign (‘@’) after each symbolic link, a percent sign (‘%’) after each whiteout, an equal sign (‘=’) after each socket, and a vertical bar (‘|’) after each that is a FIFO. The use of **-F** implies **-l**.

-f *format* Display information using the specified format. See the **FORMATS** section for a description of valid formats.

-L Use stat(2) instead of lstat(2). The information reported by **stat** will refer to the target of *file*, if file is a symbolic link, and not to *file* itself.

-l Display output in **ls -lT** format.

-n Do not force a newline to appear at the end of each piece of output.

-q Suppress failure messages if calls to stat(2) or lstat(2) fail. When run as **readlink**, error messages are automatically suppressed.

-r Display raw information. That is, for all the fields in the stat-structure, display the raw, numerical value (for example, times in seconds since the epoch, etc.)

-s Display information in “shell output”, suitable for initializing variables.

-t *timefmt* Display timestamps using the specified format. This format is passed directly to strftime(3).

-x Display information in a more verbose way as known from some Linux distributions.

FORMATS

Format strings are similar to printf(3) formats in that they start with **%**, are then followed by a sequence of formatting characters, and end in a character that selects the field of the struct stat which is to be formatted. If the **%** is immediately followed by one of **n**, **t**, **%**, or **@**, then a newline character, a tab character, a percent character, or the current file number is printed, otherwise the string is examined for the following:

Any of the following optional flags:

Selects an alternate output form for octal and hexadecimal output. Non-zero octal output will have a leading zero, and non-zero hexadecimal output will have "0x" prepended to it.

+ Asserts that a sign indicating whether a number is positive or negative should always be printed. Non-negative numbers are not usually printed with a sign.

- Aligns string output to the left of the field, instead of to the right.

0 Sets the fill character for left padding to the 0 character, instead of a space.

space Reserves a space at the front of non-negative signed output fields. A '**+**' overrides a space if both are used.

Then the following fields:

size An optional decimal digit string specifying the minimum field width.

prec An optional precision composed of a decimal point '**.**' and a decimal digit string that indicates the maximum string length, the number of digits to appear after the decimal point in floating point output, or the minimum number of digits to appear in numeric output.

fmt An optional output format specifier which is one of **D**, **O**, **U**, **X**, **F**, or **S**. These represent signed decimal output, octal output, unsigned decimal output, hexadecimal output, floating point output, and string output, respectively. Some output formats do not apply to all fields. Floating point output only applies to timespec fields (the **a**, **m**, and **c** fields).

The special output specifier **S** may be used to indicate that the output, if applicable, should be in string format. May be used in combination with

amc Display date in strftime(3) format.

dr Display actual device name.

gu Display group or user name.

p Display the mode of *file* as in **ls -lTd**.

N Displays the name of *file*.

T Displays the type of *file*.

Y Insert a " -> " into the output. Note that the default output format for **Y** is a string, but if specified explicitly, these four characters are prepended.

sub An optional sub field specifier (high, middle, or low). Only applies to the **p**, **d**, **r**, **T**, **N**, and **z** output formats. It can be one of the following:

H "High" -- depending on the **datum**:
- **d**, **r** Major number for devices
- **p** "User" bits from the string form of permissions or the file "type" bits from the numeric forms
- **T** The long output form of file type
- **N** Directory path of the file, similar to what dirname(1) would show
- **z** File size, rounded to the nearest gigabyte

M "Middle" -- depending on the **datum**:
- **p** The "group" bits from the string form of permissions or the "suid", "sgid", and "sticky" bits from the numeric forms

z File size, rounded to the nearest megabyte

L "Low" -- depending on the **datum**:
- **r**, **d** Minor number for devices
- **p** The "other" bits from the string form of permissions or the "user", "group", and "other" bits from the numeric forms
- **T** The **ls -F** style output character for file type (the use of **L** here is optional)
- **N** Base filename of the file, similar to what basename(1) would show
- **z** File size, rounded to the nearest kilobyte

datum A required field specifier, being one of the following:

d Device upon which *file* resides (*st_dev*).

i *file*'s inode number (*st_ino*).

p File type and permissions (*st_mode*).

l Number of hard links to *file* (*st_nlink*).

u, **g** User-id and group-id of *file*'s owner (*st_uid*, *st_gid*).

r Device number for character and block device special files (*st_rdev*).

a, **m**, **c**, **B** The time *file* was last accessed or modified, or when the inode was last changed, or the birth time of the inode (*st_atimespec*, *st_mtimespec*, *st_ctimespec*).

z The size of *file* in bytes (*st_size*).

b Number of blocks allocated for *file* (*st_blocks*).

k Optimal file system I/O operation block size (*st_blksize*).

f User defined flags for *file* (*st_flags*).

v Inode generation number (*st_gen*).

The following five field specifiers are not drawn directly from the data in struct stat, but are:

N The name of the file.

R The absolute pathname corresponding to the file.

T The file type, either as in **ls -F** or in a more descriptive form if the sub field specifier **H** is given.

Y The target of a symbolic link.

Z Expands to "major,minor" from the rdev field for character or block special devices and gives size output for all others.

Only the **%** and the field specifier are required. Most field specifiers default to **U** as an output form, with the exception of **p** which defaults to **O**; **a**, **m**, and **c** which default to **D**; and **Y**, **T**, and **N**, which default to **S**.

EXIT STATUS

stat exits 0 on success, and >0 if an error occurred.

EXAMPLES

If no options are specified, the default format is "%d %i %Sp %l %Su %Sg %r %z \"%Sa\" \"%Sm\" \"%Sc\" \"%SB\" %k %b %#Xf %N".

```
> stat /tmp/bar
0 78852 -rw-r--r-- 1 root wheel 0 0 "Jul  8 10:26:03 2004" "Jul  8 10:26:03 2004
```

Given a symbolic link “foo” that points from `/tmp/foo` to `/`, you would use **`stat`** as follows:

```
> stat -F /tmp/foo
lrwxrwxrwx 1 jschauma cs 1 Apr 24 16:37:28 2002 /tmp/foo@ -> /

> stat -LF /tmp/foo
drwxr-xr-x 16 root wheel 512 Apr 19 10:57:54 2002 /tmp/foo/
```

To initialize some shell-variables, you could use the **`-s`** flag as follows:

```
> csh
% eval set ‘stat -s .cshrc‘
% echo $st_size $st_mtime
1148 1015432481

> sh
$ eval $(stat -s .profile)
$ echo $st_size $st_mtime
1148 1015432481
```

In order to get a list of the kind of files including files pointed to if the file is a symbolic link, you could use the following format:

```
$ stat -f "%N: %HT%SY" /tmp/*
/tmp/bar: Symbolic Link -> /tmp/foo
/tmp/output25568: Regular File
/tmp/blah: Directory
/tmp/foo: Symbolic Link -> /
```

In order to get a list of the devices, their types and the major and minor device numbers, formatted with tabs and linebreaks, you could use the following format:

```
stat -f "Name: %N%n%tType: %HT%n%tMajor: %Hr%n%tMinor: %Lr%n%n" /dev/*
[...]
Name: /dev/wt8
        Type: Block Device
        Major: 3
        Minor: 8

Name: /dev/zero
        Type: Character Device
        Major: 2
        Minor: 12
```

In order to determine the permissions set on a file separately, you could use the following format:

```
> stat -f "%Sp -> owner=%SHp group=%SMp other=%SLp" .
drwxr-xr-x -> owner=rwx group=r-x other=r-x
```

In order to determine the three files that have been modified most recently, you could use the following format:

```
> stat -f "%m%t%Sm %N" /tmp/* | sort -rn | head -3 | cut -f2-
Apr 25 11:47:00 2002 /tmp/blah
Apr 25 10:36:34 2002 /tmp/bar
```

```
Apr 24 16:47:35 2002 /tmp/foo
```

SEE ALSO

basename(1), dirname(1), file(1), ls(1), lstat(2), readlink(2), stat(2), printf(3), strftime(3)

HISTORY

The **stat** utility appeared in NetBSD 1.6.

AUTHORS

The **stat** utility was written by Andrew Brown ⟨atatat@NetBSD.org⟩. This man page was written by Jan Schaumann ⟨jschauma@NetBSD.org⟩.

NAME

strings − print the strings of printable characters in files.

SYNOPSIS

strings [**−afov**] [*−min-len*]
[**−n** *min-len*] [**−−bytes=***min-len*]
[**−t** *radix*] [**−−radix=***radix*]
[**−e** *encoding*] [**−−encoding=***encoding*]
[**−**] [**−−all**] [**−−print−file−name**]
[**−−target=***bfdname*]
[**−−help**] [**−−version**] *file*...

DESCRIPTION

For each *file* given, GNU **strings** prints the printable character sequences that are at least 4 characters long (or the number given with the options below) and are followed by an unprintable character. By default, it only prints the strings from the initialized and loaded sections of object files; for other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non-text files.

OPTIONS

−a
−−all
− Do not scan only the initialized and loaded sections of object files; scan the whole files.

−f
−−print−file−name
Print the name of the file before each string.

−−help
Print a summary of the program usage on the standard output and exit.

−min-len
−n *min-len*
−−bytes=*min-len*
Print sequences of characters that are at least *min-len* characters long, instead of the default 4.

−o Like **−t o**. Some other versions of **strings** have **−o** act like **−t d** instead. Since we can not be compatible with both ways, we simply chose one.

−t *radix*
−−radix=*radix*
Print the offset within the file before each string. The single character argument specifies the radix of the offset−−−**o** for octal, **x** for hexadecimal, or **d** for decimal.

−e *encoding*
−−encoding=*encoding*
Select the character encoding of the strings that are to be found. Possible values for *encoding* are: **s** = single−7−bit−byte characters (ASCII, ISO 8859, etc., default), **S** = single−8−bit−byte characters, **b** = 16−bit bigendian, **l** = 16−bit littleendian, **B** = 32−bit bigendian, **L** = 32−bit littleendian. Useful for finding wide character strings.

−−target=*bfdname*
Specify an object code format other than your system's default format.

−v
−−version
Print the program version number on the standard output and exit.

SEE ALSO

ar (1), *nm* (1), *objdump* (1), *ranlib* (1), *readelf* (1) and the Info entries for *binutils*.

COPYRIGHT

NAME

strip − Discard symbols from object files.

SYNOPSIS

strip [**−F** *bfdname* |**−−target=***bfdname*]
[**−I** *bfdname* |**−−input−target=***bfdname*]
[**−O** *bfdname* |**−−output−target=***bfdname*]
[**−s**|**−−strip−all**]
[**−S**|**−g**|**−d**|**−−strip−debug**]
[**−K** *symbolname* |**−−keep−symbol=***symbolname*]
[**−N** *symbolname* |**−−strip−symbol=***symbolname*]
[**−w**|**−−wildcard**]
[**−x**|**−−discard−all**] [**−X** |**−−discard−locals**]
[**−R** *sectionname* |**−−remove−section=***sectionname*]
[**−o** *file*] [**−p**|**−−preserve−dates**]
[**−−only−keep−debug**]
[**−v** |**−−verbose**] [**−V**|**−−version**]
[**−−help**] [**−−info**]
objfile...

DESCRIPTION

GNU **strip** discards all symbols from object files *objfile*. The list of object files may include archives. At least one object file must be given.

strip modifies the files named in its argument, rather than writing modified copies under different names.

OPTIONS

−F *bfdname*
−−target=*bfdname*
Treat the original *objfile* as a file with the object code format *bfdname*, and rewrite it in the same format.

−−help
Show a summary of the options to **strip** and exit.

−−info
Display a list showing all architectures and object formats available.

−I *bfdname*
−−input−target=*bfdname*
Treat the original *objfile* as a file with the object code format *bfdname*.

−O *bfdname*
−−output−target=*bfdname*
Replace *objfile* with a file in the output format *bfdname*.

−R *sectionname*
−−remove−section=*sectionname*
Remove any section named *sectionname* from the output file. This option may be given more than once. Note that using this option inappropriately may make the output file unusable.

−s
−−strip−all
Remove all symbols.

−g
−S
−d
−−strip−debug
Remove debugging symbols only.

--strip-unneeded
Remove all symbols that are not needed for relocation processing.

-K *symbolname*
--keep-symbol=*symbolname*
Keep only symbol *symbolname* from the source file. This option may be given more than once.

-N *symbolname*
--strip-symbol=*symbolname*
Remove symbol *symbolname* from the source file. This option may be given more than once, and may be combined with strip options other than **-K**.

-o *file*
Put the stripped output in *file*, rather than replacing the existing file. When this argument is used, only one *objfile* argument may be specified.

-p
--preserve-dates
Preserve the access and modification dates of the file.

-w
--wildcard
Permit regular expressions in *symbolname*s used in other command line options. The question mark (?), asterisk (*), backslash (\) and square brackets ([]) operators can be used anywhere in the symbol name. If the first character of the symbol name is the exclamation point (!) then the sense of the switch is reversed for that symbol. For example:

```
-w -K !foo -K fo*
```

would cause strip to only keep symbols that start with the letters ‘‘fo’’, but to discard the symbol ‘‘foo’’.

-x
--discard-all
Remove non-global symbols.

-X
--discard-locals
Remove compiler-generated local symbols. (These usually start with **L** or **..**)

--only-keep-debug
Strip a file, removing any sections that would be stripped by **--strip-debug** and leaving the debugging sections.

The intention is that this option will be used in conjunction with **--add-gnu-debuglink** to create a two part executable. One a stripped binary which will occupy less space in RAM and in a distribution and the second a debugging information file which is only needed if debugging abilities are required. The suggested procedure to create these files is as follows:

1.<Link the executable as normal. Assuming that is is called>
`foo` then...

1.<Run `objcopy --only-keep-debug foo foo.dbg` to>
create a file containing the debugging info.

1.<Run `objcopy --strip-debug foo` to create a>
stripped executable.

1.<Run `objcopy --add-gnu-debuglink=foo.dbg foo`>
to add a link to the debugging info into the stripped executable.

Note - the choice of `.dbg` as an extension for the debug info file is arbitrary. Also the `--only-keep-debug` step is optional. You could instead do this:

1.<Link the executable as normal.>
1.<Copy `foo` to `foo.full`>
1.<Run `strip --strip-debug foo`>
1.<Run `objcopy --add-gnu-debuglink=foo.full foo`>

ie the file pointed to by the **--add-gnu-debuglink** can be the full executable. It does not have to be a file created by the **--only-keep-debug** switch.

-V
--version
Show the version number for **strip**.

-v
--verbose
Verbose output: list all object files modified. In the case of archives, **strip -v** lists all members of the archive.

SEE ALSO

the Info entries for *binutils*.

COPYRIGHT

NAME

stty — set the options for a terminal device interface

SYNOPSIS

stty [**-a** | **-e** | **-g**] [**-f** *file*] [operands]

DESCRIPTION

The **stty** utility sets or reports on terminal characteristics for the device that is its standard input. If no options or operands are specified, it reports the settings of a subset of characteristics as well as additional ones if they differ from their default values. Otherwise it modifies the terminal state according to the specified arguments. Some combinations of arguments are mutually exclusive on some terminal types.

The following options are available:

-a Display all the current settings for the terminal to standard output as per IEEE Std 1003.2 (“POSIX.2”).

-e Display all the current settings for the terminal to standard output in the traditional BSD ‘‘all’’ and ‘‘everything’’ formats.

-f Open and use the terminal named by *file* rather than using standard input. The file is opened using the O_NONBLOCK flag of **open**(), making it possible to set or display settings on a terminal that might otherwise block on the open.

-g Display all the current settings for the terminal to standard output in a form that may be used as an argument to a subsequent invocation of **stty** to restore the current terminal state as per IEEE Std 1003.2 (“POSIX.2”).

The following arguments are available to set the terminal characteristics:

Control Modes

Control mode flags affect hardware characteristics associated with the terminal. This corresponds to the c_cflag in the termios structure.

parenb (**-parenb**)
Enable (disable) parity generation and detection.

parodd (**-parodd**)
Select odd (even) parity.

cs5 cs6 cs7 cs8
Select character size, if possible.

number Set terminal baud rate to the number given, if possible. If the baud rate is set to zero, modem control is no longer asserted.

ispeed *number*
Set terminal input baud rate to the number given, if possible. If the input baud rate is set to zero, the input baud rate is set to the value of the output baud rate.

ospeed *number*
Set terminal output baud rate to the number given, if possible. If the output baud rate is set to zero, modem control is no longer asserted.

speed *number*
This sets both **ispeed** and **ospeed** to *number*.

hupcl (**-hupcl**)
Stop asserting modem control (do not stop asserting modem control) on last close.

hup (**-hup**)
Same as hupcl (**-hupcl**).

cstopb (**-cstopb**)
Use two (one) stop bits per character.

cread (**-cread**)
Enable (disable) the receiver.

clocal (**-clocal**)
Assume a line without (with) modem control.

crtscts (**-crtscts**)
Enable RTS/CTS flow control.

cdtrcts (**-cdtrcts**)
Enable DTR/CTS flow control (if supported).

Input Modes
This corresponds to the c_iflag in the termios structure.

ignbrk (**-ignbrk**)
Ignore (do not ignore) break on input.

brkint (**-brkint**)
Signal (do not signal) INTR on break.

ignpar (**-ignpar**)
Ignore (do not ignore) parity errors.

parmrk (**-parmrk**)
Mark (do not mark) parity errors.

inpck (**-inpck**)
Enable (disable) input parity checking.

istrip (**-istrip**)
Strip (do not strip) input characters to seven bits.

inlcr (**-inlcr**)
Map (do not map) NL to CR on input.

igncr (**-igncr**)
Ignore (do not ignore) CR on input.

icrnl (**-icrnl**)
Map (do not map) CR to NL on input.

ixon (**-ixon**)
Enable (disable) START/STOP output control. Output from the system is stopped when the system receives STOP and started when the system receives START, or if **ixany** is set, any character restarts output.

ixoff (**-ixoff**)
Request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

ixany (**-ixany**)
Allow any character (allow only START) to restart output.

imaxbel (**-imaxbel**)
The system imposes a limit of MAX_INPUT (currently 255) characters in the input queue. If **imaxbel** is set and the input queue limit has been reached, subsequent input causes the system to send an ASCII BEL character to the output queue (the terminal beeps at you). Otherwise, if **imaxbel** is unset and the input queue is full, the next input character causes the entire input and output queues to be discarded.

Output Modes
This corresponds to the c_oflag of the termios structure.

opost (**-opost**)
Post-process output (do not post-process output; ignore all other output modes).

onlcr (**-onlcr**)
Map (do not map) NL to CR-NL on output.

ocrnl (**-ocrnl**)
Map (do not map) CR to NL on output.

oxtabs (**-oxtabs**)
Expand (do not expand) tabs to spaces on output.

onocr (**-onocr**)
Do not (do) output CRs at column zero.

onlret (**-onlret**)
On the terminal NL performs (does not perform) the CR function.

Local Modes
Local mode flags (lflags) affect various and sundry characteristics of terminal processing. Historically the term "local" pertained to new job control features implemented by Jim Kulp on a PDP-11/70 at IIASA. Later the driver ran on the first VAX at Evans Hall, UC Berkeley, where the job control details were greatly modified but the structure definitions and names remained essentially unchanged. The second interpretation of the 'l' in lflag is ''line discipline flag'' which corresponds to the *c_lflag* of the *termios* structure.

isig (**-isig**)
Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SUSP.

icanon (**-icanon**)
Enable (disable) canonical input (ERASE and KILL processing).

iexten (**-iexten**)
Enable (disable) any implementation defined special control characters not currently controlled by icanon, isig, or ixon.

echo (**-echo**)
Echo back (do not echo back) every character typed.

echoe (**-echoe**)
The ERASE character shall (shall not) visually erase the last character in the current line from the display, if possible.

echok (**-echok**)
Echo (do not echo) NL after KILL character.

echoke (**-echoke**)
The KILL character shall (shall not) visually erase the current line from the display, if possible.

echonl (**-echonl**)
Echo (do not echo) NL, even if echo is disabled.

echoctl (**-echoctl**)
If **echoctl** is set, echo control characters as ^X. Otherwise control characters echo as themselves.

echoprt (**-echoprt**)
For printing terminals. If set, echo erased characters backwards within ‘\’ and ‘/’. Otherwise, disable this feature.

noflsh (**-noflsh**)
Disable (enable) flush after INTR, QUIT, SUSP.

tostop (**-tostop**)
Send (do not send) SIGTTOU for background output. This causes background jobs to stop if they attempt terminal output.

altwerase (**-altwerase**)
Use (do not use) an alternative word erase algorithm when processing WERASE characters. This alternative algorithm considers sequences of alphanumeric/underscores as words. It also skips the first preceding character in its classification (as a convenience since the one preceding character could have been erased with simply an ERASE character.)

mdmbuf (**-mdmbuf**)
If set, flow control output based on condition of Carrier Detect. Otherwise writes return an error if Carrier Detect is low (and Carrier is not being ignored with the CLOCAL flag.)

flusho (**-flusho**)
Indicates output is (is not) being discarded.

pendin (**-pendin**)
Indicates input is (is not) pending after a switch from non-canonical to canonical mode and will be re-input when a read becomes pending or more input arrives.

Control Characters

control-character string
Set *control-character* to *string*. If string is a single character, the control character is set to that character. If string is the two character sequence "^-" or the string "undef" the control character is disabled (i.e. set to {_POSIX_VDISABLE}.)

Recognized control-characters:

control-character	Subscript	Description
eof	VEOF	EOF character
eol	VEOL	EOL character
eol2	VEOL2	EOL2 character

erase	VERASE	ERASE character
werase	VWERASE	WERASE character
kill	VKILL	KILL character
reprint	VREPRINT	REPRINT character
intr	VINTR	INTR character
quit	VQUIT	QUIT character
susp	VSUSP	SUSP character
dsusp	VDSUSP	DSUSP character
start	VSTART	START character
stop	VSTOP	STOP character
lnext	VLNEXT	LNEXT character
status	VSTATUS	STATUS character
discard	VDISCARD	DISCARD character

min *number*

time *number*

Set the value of min or time to number. `MIN` and `TIME` are used in Non-Canonical mode input processing (-icanon).

Combination Modes

saved settings

Set the current terminal characteristics to the saved settings produced by the **-g** option.

evenp or **parity**

Enable parenb and cs7; disable parodd.

oddp Enable parenb, cs7, and parodd.

-parity, **-evenp**, **-oddp**

Disable parenb, and set cs8.

nl (**-nl**)

Enable (disable) icrnl. In addition -nl unsets inlcr and igncr.

ek Reset `ERASE` and `KILL` characters back to system defaults.

sane Resets all modes to reasonable values for interactive terminal use.

insane Resets all modes to unreasonable values for interactive terminal use.

tty Set the line discipline to the standard terminal line discipline `TTYDISC`.

crt (**-crt**)

Set (disable) all modes suitable for a CRT display device.

kerninfo (**-kerninfo**)

Enable (disable) the system generated status line associated with processing a `STATUS` character (usually set to ^T). The status line consists of the system load average, the current command name, its process ID, the event the process is waiting on (or the status of the process), the user and system times, percent CPU, and current memory usage.

columns *number*

The terminal size is recorded as having *number* columns.

cols *number*

is an alias for **columns**.

rows *number*
The terminal size is recorded as having *number* rows.

dec Set modes suitable for users of Digital Equipment Corporation systems (ERASE, KILL, and INTR characters are set to ^?, ^U, and ^C; ixany is disabled, and crt is enabled.)

extproc (**-extproc**)
If set, this flag indicates that some amount of terminal processing is being performed by either the terminal hardware or by the remote side connected to a pty.

raw (**-raw**)
If set, change the modes of the terminal so that no input or output processing is performed. If unset, change the modes of the terminal to some reasonable state that performs input and output processing. Note that since the terminal driver no longer has a single RAW bit, it is not possible to intuit what flags were set prior to setting **raw**. This means that unsetting **raw** may not put back all the setting that were previously in effect. To set the terminal into a raw state and then accurately restore it, the following shell code is recommended:

```
save_state=$(stty -g)
stty raw
...
stty "$save_state"
```

size The size of the terminal is printed as two numbers on a single line, first rows, then columns.

Compatibility Modes
These modes remain for compatibility with the previous version of the stty command.

all Reports all the terminal modes as with **stty -a** except that the control characters are printed in a columnar format.

everything
Same as **all**.

cooked Same as **sane**.

cbreak If set, enables **brkint**, **ixon**, **imaxbel**, **opost**, **isig**, **iexten**, and **-icanon**. If unset, same as **sane**.

new Same as **tty**.

old Same as **tty**.

newcrt (**-newcrt**)
Same as **crt**.

pass8 The converse of **parity**.

tandem (**-tandem**)
Same as **ixoff**.

decctlq (**-decctlq**)
The converse of **ixany**.

crterase (**-crterase**)
Same as **echoe**.

crtbs (**-crtbs**)
Same as **echoe**.

crtkill (**-crtkill**)
Same as **echoke**.

ctlecho (**-ctlecho**)
Same as **echoctl**.

prterase (**-prterase**)
Same as **echoprt**.

litout (**-litout**)
The converse of **opost**.

tabs (**-tabs**)
The converse of **oxtabs**.

brk *value*
Same as the control character **eol**.

flush *value*
Same as the control character **discard**.

rprnt *value*
Same as the control character **reprint**.

Control operations
These operations are not modes, but rather commands to be performed by the tty layer.

ostart Performs a "start output" operation, as normally done by an incoming START character when **ixon** is set.

ostop Performs a "stop output" operation, as normally done by an incoming STOP character when **ixon** is set.

EXIT STATUS
The **stty** utility exits with a value of 0 if successful, and >0 if an error occurs.

SEE ALSO
termios(4), tty(4)

STANDARDS
The **stty** utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible. The flags **-e** and **-f** are extensions to the standard, as are the operands mentioned in the control operations section.

NAME

su — substitute user identity

SYNOPSIS

su [**-K** | **--no-kerberos**] [**-f**] [**-l** | **--full**] [**-m**] [**-i** *instance* | **--instance=***instance*] [**-c** *command* | **--command=***command*] [*login* [*shell arguments*]]

DESCRIPTION

su will use Kerberos authentication provided that an instance for the user wanting to change effective UID is present in a file named `.k5login` in the target user id's home directory

A special case exists where `root`'s `~/.k5login` needs to contain an entry for: `user`/⟨instance⟩@REALM for **su** to succed (where ⟨instance⟩ is `root` unless changed with **-i**).

In the absence of either an entry for current user in said file or other problems like missing `host/hostname@REALM` keys in the system's keytab, or user typing the wrong password, **su** will fall back to traditional `/etc/passwd` authentication.

When using `/etc/passwd` authentication, **su** allows `root` access only to members of the group `wheel`, or to any user (with knowledge of the `root` password) if that group does not exist, or has no members.

The options are as follows:

-K, **--no-kerberos** don't use Kerberos.

-f don't read .cshrc.

-l, **--full** simulate full login.

-m leave environment unmodified.

-i *instance*, **--instance=***instance* root instance to use.

-c *command*, **--command=***command* command to execute.

NAME

su — substitute user identity

SYNOPSIS

su [**-Kdflm**] [**-c** *login-class*] [*login*[:*group*] [*shell arguments*]]
su [**-Kdflm**] [**-c** *login-class*] [:*group* [*shell arguments*]]

DESCRIPTION

su allows one user to become another user *login* without logging out and in as the new user. If a *group* is specified and *login* is a member of *group*, then the group is changed to *group* rather than to *login*'s primary group. If *login* is omitted and *group* is provided (form two above), then *login* is assumed to be the current username.

When executed by a user, the *login* user's password is requested. When using Kerberos, the password for *login* (or for "*login*.root", if no login is provided) is requested, and **su** switches to that user and group ID after obtaining a Kerberos ticket granting ticket. A shell is then executed, and any additional *shell arguments* after the login name are passed to the shell. **su** will resort to the local password file to find the password for *login* if there is a Kerberos error. If **su** is executed by root, no password is requested and a shell with the appropriate user ID is executed; no additional Kerberos tickets are obtained.

Alternatively, if the user enters the password "s/key", authentication will use the S/Key one-time password system as described in skey(1). S/Key is a Trademark of Bellcore.

By default, the environment is unmodified with the exception of LOGNAME, USER, HOME, SHELL, and SU_FROM. HOME and SHELL are set to the target login's default values. LOGNAME and USER are set to the target login, unless the target login has a user ID of 0, in which case they are unmodified. SU_FROM is set to the caller's login. The invoked shell is the target login's. With the exception of SU_FROM this is the traditional behavior of **su**.

The options are as follows:

-K Do not attempt to use Kerberos to authenticate the user.

-c Specify a login class. You may only override the default class if you're already root. See login.conf(5) for details.

-d Same as **-l**, but does not change the current directory.

-f If the invoked shell is csh(1), this option prevents it from reading the "**.cshrc**" file. If the invoked shell is sh(1), or ksh(1), this option unsets ENV, thus preventing the shell from executing the startup file pointed to by this variable.

-l Simulate a full login. The environment is discarded except for HOME, SHELL, PATH, TERM, LOGNAME, USER, and SU_FROM. HOME, SHELL, and SU_FROM are modified as above. LOGNAME and USER are set to the target login. PATH is set to the path specified in the /etc/login.conf file (or to the default of "/usr/bin:/bin:/usr/pkg/bin:/usr/local/bin"). TERM is imported from your current environment. The invoked shell is the target login's, and **su** will change directory to the target login's home directory.

- Same as **-l**.

-m Leave the environment unmodified. The invoked shell is your login shell, and no directory changes are made. As a security precaution, if the target user's shell is a non-standard shell (as defined by getusershell(3)) and the caller's real uid is non-zero, **su** will fail.

The **-l** and **-m** options are mutually exclusive; the last one specified overrides any previous ones.

Only users in group "wheel" (normally gid 0), as listed in `/etc/group`, can **su** to "root", unless group wheel does not exist or has no members. (If you do not want anybody to be able to **su** to "root", make "root" the only member of group "wheel", which is the default.)

For sites with very large user populations, group "wheel" can contain the names of other groups that will be considered authorized to **su** to "root".

By default (unless the prompt is reset by a startup file) the super-user prompt is set to "#" to remind one of its awesome power.

CUSTOMIZATION

Changing required group

For the pam(8) version of **su** the name of the required group can be changed by setting *gname* in pam.conf(5):

```
auth requisite pam_group.so no_warn group=gname root_only fail_safe
```

For the non pam(8) version of **su** the same can be achieved by compiling with SU_GROUP set to the desired group name.

Supplying own password

su can be configured so that users in a particular group can supply their own password to become "root". For the pam(8) version of **su** this can be done by adding a line to pam.conf(5) such as:

```
auth sufficient pam_group.so no_warn group=gname root_only authenticate
```

where *gname* is the name of the desired group. For the non pam(8) version of **su** the same can be achieved by compiling with SU_ROOTAUTH set to the desired group name.

Indirect groups

This option is not available with the pam(8) version of **su**. For the non pam(8) version of **su**, if SU_INDIRECT_GROUP is defined, the *SU_GROUP* and *SU_ROOTAUTH* groups are treated as indirect groups. The group members of those two groups are treated as groups themselves.

EXIT STATUS

su returns the exit status of the executed subshell, or 1 if any error occurred while switching privileges.

ENVIRONMENT

Environment variables used by **su**:

HOME Default home directory of real user ID unless modified as specified above.

LOGNAME
The user ID is always the effective ID (the target user ID) after an **su** unless the user ID is 0 (root).

PATH Default search path of real user ID unless modified as specified above.

TERM Provides terminal type which may be retained for the substituted user ID.

USER The user ID is always the effective ID (the target user ID) after an **su** unless the user ID is 0 (root).

EXAMPLES

To become user username and use the same environment as in original shell, execute:

```
su username
```

To become user username and use environment as if full login would be performed, execute:

```
su -l username
```

When a **-c** option is included *after* the *login* name it is not a **su** option, because any arguments after the *login* are passed to the shell. (See csh(1), ksh(1) or sh(1) for details.) To execute arbitrary command with privileges of user *username*, execute:

```
su username -c "command args"
```

SEE ALSO

csh(1), kinit(1), login(1), sh(1), skey(1), setusercontext(3), group(5), login.conf(5), passwd(5), environ(7), kerberos(8)

HISTORY

A **su** command existed in Version 5 AT&T UNIX (and probably earlier).

NAME

sup − software upgrade protocol

SYNOPSIS

sup [*flags*] [*supfile*] [*collection* ...]

DESCRIPTION

Sup is a program used for upgrading collections of files from other machines to your machine. You execute *sup*, the *client* program, which talks over the network using IP/TCP to a *file server* process. The file server process cooperates with *sup* to determine which files of the collection need to be upgraded on your machine.

Sup collections can have multiple releases. One use for such releases is to provide different versions of the same files. At CMU, for example, system binaries have alpha, beta and default release corresponding to different staging levels of the software. We also use release names default and minimal to provide complete releases or subset releases. In both of these cases, it only makes sense to sup one release of the collections. Releases have also been used in private or external sups to provide subsets of collections where it makes sense to pick up several of the releases. For example the Mach 3.0 kernel sources has a default release of machine independent sources and separate releases of machine dependent sources for each supported platform.

In performing an upgrade, the file server constructs a list of files included in the specified release of the collection. The list is sent to your machine, which determines which files are needed. Those files are then sent from the file server. It will be most useful to run *sup* as a daemon each night so you will continually have the latest version of the files in the needed collections.

The only required argument to *sup* is the name of a supfile. It must either be given explicitly on the command line, or the **-s** flag must be specified. If the **-s** flag is given, the system supfile will be used and a supfile command argument should not be specified. The list of collections is optional and if specified will be the only collections upgraded. The following flags affect all collections specified:

-s As described above.

-t When this flag is given, *sup* will print the time that each collection was last upgraded, rather than performing actual upgrades.

-u When this flag is given, *sup* will not try to restore the user access and modified times of files in the collections from the server.

-S Operate silently printing messages only on errors.

-N *Sup* will trace network messages sent and received that implement the *sup* network protocol.

-P Sup will use a set of non-privileged network ports reserved for debugging purposes.

The remaining flags affect all collections unless an explicit list of collections are given with the flags. Multiple flags may be specified together that affect the same collections. For the sake of convenience, any flags that always affect all collections can be specified with flags that affect only some collections. For example, **sup -sde=coll1,coll2** would perform a system upgrade, and the first two collections would allow both file deletions and command executions. Note that this is not the same command as **sup -sde=coll1 coll2,** which would perform a system upgrade of just the coll2 collection and would ignore the flags given for the coll1 collection.

-a All files in the collection will be copied from the repository, regardless of their status on the current machine. Because of this, it is a very expensive operation and should only be done for small collections if data corruption is suspected and been confirmed. In most cases, the **-o** flag should be sufficient.

-b If the **-b** flag if given, or the **backup** supfile option is specified, the contents of regular files on the local system will be saved before they are overwritten with new data. The file collection

maintainer can designate specific files to be worthy of backing up whenever they are upgraded. However, such backup will only take place if you specify this flag or the **backup** option to allow backups for a file collection on your machine. The backup mechanism will create a copy of the current version of a file immediately before a new copy is received from the file server; the copy is given the same name as the original file but is put into a directory called **BACKUP** within the directory containing the original file. For example, **/usr/sas/src/foo.c** would have a backup copy called **/usr/sas/src/BACKUP/foo.c.** There is no provision for automatically maintaining multiple old versions of files; you would have to do this yourself.

-B The **-B** flag overrides and disables the **-b** flag and the **backup** supfile option.

-d Files that are no longer in the collection on the repository will be deleted if present on the local machine and were put there by a previous sup. This may also be specified in a supfile with the **delete** option.

-D The **-D** flag overrides and disables the **-d** flag and the **delete** supfile option.

-e Sup will execute commands sent from the repository that should be run when a file is upgraded. If the **-e** flag is omitted, Sup will print a message that specifies the command to execute. This may also be specified in a supfile with the **execute** option.

-E The **-E** flag overrides and disables the **-e** flag and the **execute** supfile option.

-f A *list-only* upgrade will be performed. Messages will be printed that indicate what would happen if an actual upgrade were done.

-k *Sup* will check the modification times of files on the local disk before updating them. Only files which are newer on the repository than on the local disk will be updated; files that are newer on the local disk will be kept as they are. This may also be specified in a supfile with the **keep** option.

-K The **-K** flag overrides and disables the **-k** flag and the **keep** supfile option.

-l Normally, *sup* will not upgrade a collection if the repository is on the same machine. This allows users to run upgrades on all machines without having to make special checks for the repository machine. If the **-l** flag is specified, collections will be upgraded even if the repository is local.

-m Normally, *sup* used standard output for messages. If the **-m** flag if given, *sup* will send mail to the user running *sup*, or a user specified with the **notify** supfile option, that contains messages printed by *sup*.

-o *Sup* will normally only upgrade files that have changed on the repository since the last time an upgrade was performed. That is, if the file in the repository is newer than the date stored in the *when* file on the client. The **-o** flag, or the **old** supfile option, will cause *sup* to check all files in the collection for changes instead of just the new ones.

-O The **-O** flag overrides and disables the **-o** flag and the **old** supfile option.

-z Normally sup transfers files directly without any other processing, but with the **-z** flag, or the **compress** supfile option, sup will compress the file before sending it across the network and uncompress it and restore all the correct file attributes at the receiving end.

-Z The **-Z** flag overrides and disables the **-z** flag and the **compress** supfile option.

-v Normally, *sup* will only print messages if there are problems. This flag causes *sup* to also print messages during normal progress showing what *sup* is doing.

SETTING UP UPGRADES

Each file collection to be upgraded must have a *base directory* which contains a subdirectory called **sup** that will be used by the *sup* program; it will be created automatically if you do not create it. *Sup* will put subdirectories and files into this directory as needed.

Sup will look for a subdirectory with the same name as the collection within the **sup** subdirectory of the *base directory.* If it exists it may contain any of the following files:

when.<rel-suffix>
This file is automatically updated by *sup* when a collection is successfully upgraded and contains the time that the file server, or possibly *supscan*, created the list of files in the upgrade list. *Sup* will send this time to the file server for generating the list of files that have been changed on the repository machine.

refuse This file contains a list of files and directories, one per line, that the client is not interested in that should not be upgraded.

lock This file is used by *sup* to lock a collection while it is being upgraded. *Sup* will get exclusive access to the lock file using *flock*(2), preventing more than one *sup* from upgrading the same collection at the same time.

last.<rel-suffix>
This file contains a list of files and directories, one per line, that have been upgraded by *sup* in the past. This information is used when the **delete** option, or the **-d** flag is used to locate files previously upgraded that are no longer in the collection that should be deleted.

Each file collection must also be described in one or more supfiles. When *sup* is executed, it reads the specified supfile to determine what file collections and releases to upgrade. Each collection-release set is described by a single line of text in the supfile; this line must contain the name of the collection, and possibly one or more options separated by spaces. The options are:

release=*releasename*
If a collection contains multiple releases, you need to specify which release you want. You can only specify one release per line, so if you want multiple releases from the same collections, you will need to specify the collection more than once. In this case, you should use the *use-rel-suffix* option in the supfile to keep the last and when files for the two releases separate.

base=*directory*
The usual default name of the base directory for a collection is described below (see FILES); if you want to specify another directory name, use this option specifying the desired directory.

prefix=*directory*
Each collection may also have an associated *prefix directory* which is used instead of the base directory to specify in what directory files within the collection will be placed.

host=*hostname*
hostbase=*directory*
System collections are supported by the system maintainers, and *sup* will automatically find out the name of the host machine and base directory on that machine. However, you can also upgrade *private* collections; you simply specify with these options the *hostname* of the machine containing the files and the *directory* used as a base directory for the file server on that machine. Details of setting up a file collection are given in the section below.

login=*accountid*
password= *password*
crypt=*key*
Files on the file server may be protected, and network transmissions may be encrypted. This prevents unauthorized access to files via *sup*. When files are not accessible to the default account (e.g. the **anon** anonymous account), you can specify an alternative *accountid* and *password* for the file server to use on the repository host. Network transmission of the password will be always be encrypted. You can also have the actual file data encrypted by specifying a *key*; the file collection on the repository must specify the same key or else *sup* will not be able to upgrade files from that collection. In this case, the default account used by the file server on the repository machine will be the owner of the encryption key file (see FILES) rather than the **anon** anonymous account.

notify=*address*

If you use the **-m** option to receive log messages by mail, you can have the mail sent to different user, possibly on another host, than the user running the sup program. Messages will be sent to the specified *address*, which can be any legal netmail address. In particular, a project maintainer can be designated to receive mail for that project's file collection from all users running *sup* to upgrade that collection.

backup

As described above under the **-b** flag.

delete As described above under the **-d** flag.

execute

As described above under the **-e** flag.

keep As described above under the **-k** flag.

old As described above under the **-o** flag.

use-rel-suffix

Causes the release name to be used as a suffix to the *last* and *when* files. This is necessary whenever you are supping more than one release in the same collection.

PREPARING A FILE COLLECTION REPOSITORY

A set of files residing on a repository must be prepared before *sup* client processes can upgrade those files. The collection must be given a *name* and a *base directory.* If it is a private collection, client users must be told the name of the collection, repository host, and base directory; these will be specified in the supfile via the **host** and **hostbase** options. For a system-maintained file collection, entries must be placed into the host list file and directory list file as described in *supservers*(8).

Within the base directory, a subdirectory must be created called **sup .** Within this directory there must be a subdirectory for each collection using that base directory, whose name is the name of the collection; within each of these directories will be a list file and possibly a prefix file, a host file, an encryption key file, a log file and a scan file. The filenames are listed under FILES below.

prefix Normally, all files in the collection are relative to the base directory. This file contains a single line which is the name of a directory to be used in place of the base directory for file references.

host Normally, all remote host machines are allowed access to a file collection. If you wish to restrict access to specific remote hosts for this collection, put each allowed hostname on a separate line of text in this file. If a host has more than one name, only one of its names needs to be listed. The name **LOCAL** can be used to grant access to all hosts on the local network. The host name may be a numeric network address or a network name. If a crypt appears on the same line as the host name, that crypt will be used for that host. Otherwise, the crypt appearing in the *crypt* file, if any will be used.

crypt If you wish to use the *sup* data encryption mechanism, create an encryption file containing, on a single line of text, the desired encryption key. Client processes must then specify the same key with the **crypt** option in the supfile or they will be denied access to the files. In addition, actual network transmission of file contents and filenames will be encrypted.

list This file describes the actual list of files to be included in this file collection, in a format described below.

releases

This file describes any releases that the collection may have. Each line starts with the release name and then may specify any of the following files: *prefix=<dirname>* to use a different parent directory for the files in this release. *list=<listname>* to specify the list of files in the release. *scan=<scanfile>* must be used in multi-release collections that are scanned to keep the scan files for the different releases separate. *host=<hostfile>* to allow different host restrictions for this release. *next=<release>* used to chain releases together. This has the effect of making one release

be a combination of several other releases. If the same file appears in more than one chained release, the first one found will be used. If these files are not specified for a release the default names: prefix,list,scan and host will be used.

scan This file, created by *supscan*, is the list of filenames that correspond to the instructions in the list file. The scan file is only used for frequently updated file collections; it makes the file server run much faster. See *supservers*(8) for more information.

lock As previously mentioned, this file is used to indicate that the collection should be locked while upgrades are in progress. All file servers will try to get shared access to the lock file with *flock*(2).

logfile If a log file exists in the collection directory, the file server will append the last time an upgrade was successfully completed, the time the last upgrade started and finished, and the name of the host requesting the upgrade.

It should be noted that *sup* allows several different named collections to use the same base directory. Separate encryption, remote host access, and file lists are used for each collection, since these files reside in subdirectories <*basedir*>*/sup/*<*coll.name*>.

The list file is a text file with one command on each line. Each command contains a keyword and a number of operands separated by spaces. All filenames in the list file are evaluated on the repository machine relative to the host's base directory, or prefix directory if one is specified, and on your machine with respect to the base, or prefix, directory for the client. The *filenames* below (except *exec-command*) may all include wild-cards and meta-characters as used by *csh*(1) including *, ?, [...], and {...}. The commands are:

upgrade *filename ...*

The specified file(s) (or directories) will be included in the list of files to be upgraded. If a directory name is given, it recursively includes all subdirectories and files within that directory.

always *filename ...*

The always command is identical to upgrade, except that omit and omitany commands do not affect filenames specified with the always command.

omit *filename ...*

The specified file(s) (or directories) will be excluded from the list of files to be upgraded. For example, by specifying **upgrade /usr/vision** and **omit /usr/vision/exp,** the generated list of files would include all subdirectories and files of /usr/vision except /usr/vision/exp (and its subdirectories and files).

omitany *pattern ...*

The specified patterns are compared against the files in the upgrade list. If a pattern matches, the file is omitted. The omitany command currently supports all wild-card patterns except {...}. Also, the pattern must match the entire filename, so a leading */, or a trailing /*, may be necessary in the pattern.

backup *filename ...*

The specified file(s) are marked for backup; if they are upgraded and the client has specified the **backup** option in the corresponding line of the supfile, then backup copies will be created as described above. Directories may not be specified, and no recursive filename construction is performed; you must specify the names of the specific files to be backed up before upgrading.

noaccount *filename ...*

The accounting information of the specified file(s) will not be preserved by *sup*. Accounting information consists of the owner, group, mode and modified time of a file.

symlink *filename ...*

The specified file(s) are to be treated as symbolic links and will be transferred as such and not followed. By default, *sup* will follow symbolic links.

rsymlink *dirname ...*

All symbolic links in the specified directory and its subdirectories are to be treated as symbolic links. That is the links will be transferred and not the files to which they point.

execute *exec-command* (*filename* ...)

The *exec-command* you specified will be executed on the client process whenever any of the files listed in parentheses are upgraded. A special token, **%s,** may be specified in the *exec-command* and will be replaced by the name of the file that was upgraded. For example, if you say **execute ranlib %s (libc.a)**, then whenever libc.a is upgraded, the client machine will execute **ranlib libc.a.** As described above, the client must invoke *sup* with the **-e** flag to allow the automatic execution of command files.

include *listfile ...*

The specified *listfiles* will be read at this point. This is useful when one collection subsumes other collections; the larger collection can simply specify the listfiles for the smaller collections contained within it.

The order in which the command lines appear in the list file does not matter. Blank lines may appear freely in the list file.

FILES

Files on the client machine for *sup*:

/etc/supfiles/coll.list
supfile used for -s flag

/etc/supfiles/coll.what
supfile used for -s flag when -t flag is also specified

/etc/supfiles/coll.host
host name list for system collections

<*base-directory*>**/sup/**<*collection*>**/last**<*.release*>
recorded list of files in collection as of last upgrade

<*base-directory*>**/sup/**<*collection*>**/lock**
file used to lock collection

<*base-directory*>**/sup/**<*collection*>**/refuse**
list of files to refuse in collection

<*base-directory*>**/sup/**<*collection*>**/when**<*.release*>
recorded time of last upgrade

/usr/sup/<*collection*>
default base directory for file collection

Files needed on each repository machine for the file server:

/etc/supfiles/coll.dir
base directory list for system collections

<*base-directory*>**/sup/**<*collection*>**/crypt**
data encryption key for a collection. the owner of this file is the default account used when data encryption is specified

<*base-directory*>**/sup/**<*collection*>**/host**
list of remote hosts allowed to upgrade a collection

<*base-directory*>**/sup/**<*collection*>**/list**
list file for a collection

<*base-directory*>**/sup/**<*collection*>**/lock**
lock file for a collection

<*base-directory*>**/sup/**<*collection*>**/logfile**
log file for a collection

<*base-directory*>/**sup**/<*collection*>/**prefix**
file containing the name of the prefix directory for a collection

<*base-directory*>/**sup**/<*collection*>/**scan**
scan file for a collection

/usr/<*collection*>
default base directory for a file collection

SEE ALSO

supservers(8)
The SUP Software Upgrade Protocol, S. A. Shafer, CMU Computer Science Department, 1985.

EXAMPLE

<example>

BUGS

The encryption mechanism should be strengthened, although it’s not trivial.

NAME

systat — display system statistics on a CRT

SYNOPSIS

systat [**-n**] [**-M** *core*] [**-N** *system*] [**-t** *turns*] [**-w** *wait*] [*display*]
[*refresh-interval*]

DESCRIPTION

systat displays various system statistics in a screen oriented fashion using the curses screen display library, curses(3).

While **systat** is running the screen is usually divided into two windows (an exception is the vmstat display which uses the entire screen). The upper window depicts the current system load average. The information displayed in the lower window may vary, depending on user commands. The last line on the screen is reserved for user input and error messages.

By default **systat** displays the processes getting the largest percentage of the processor in the lower window. Other displays show more detailed process information, swap space usage, disk usage statistics (a la df(1)), disk I/O statistics (a la iostat(8)), virtual memory statistics (a la vmstat(1)), network ''mbuf'' utilization, and network connections (a la netstat(1)).

Input is interpreted at two different levels. A ''global'' command interpreter processes all keyboard input. If this command interpreter fails to recognize a command, the input line is passed to a per-display command interpreter. This allows each display to have certain display-specific commands.

Command line options:

-M *core* — Extract values associated with the name list from *core* instead of the default /dev/mem.

-N *system* — Extract the name list from *system* instead of the default /netbsd.

-n — Do not resolve IP addresses into string hostnames (FQDNs) on **netstat**. It has the same effect as **numbers** subcommand in **netstat**.

-w *wait* — See *refresh-interval*.

-t *turns* — How many refreshes to show each screen in 'all' display mode.

display — The *display* argument expects to be one of: **all**, **bufcache**, **df**, **inet.icmp**, **inet.ip**, **inet.tcp**, **inet.tcpsyn**, **inet6.ip6**, **ipsec**, **iostat**, **mbufs**, **netstat**, **pigs**, **ps**, **swap**, **syscall** or **vmstat**. These displays can also be requested interactively and are described in full detail below.

refresh-interval — The *refresh-interval* specifies the screen refresh time interval in seconds. This is provided for backwards compatibility, and overrides the *refresh-interval* specified with the **-w** flag.

Certain characters cause immediate action by **systat**. These are

^L — Refresh the screen.

^G — Print the name of the current ''display'' being shown in the lower window and the refresh interval.

^Z — Stop **systat**.

?, **h** — Print the names of the available displays on the command line.

: Move the cursor to the command line and interpret the input line typed as a command. While entering a command the current character erase, word erase, and line kill characters may be used.

The following commands are interpreted by the ‘‘global’’ command interpreter.

help *key* Print the names of the available displays on the command line. It will print long names as “**inet.**∗”. To print items under “**inet**”, give **inet** as *key*.

load Print the load average over the past 1, 5, and 15 minutes on the command line.

stop Stop refreshing the screen.

[**start**] [*number*]
Start (continue) refreshing the screen. If a second, numeric, argument is provided it is interpreted as a refresh interval in seconds. Supplying only a number will set the refresh interval to this value.

quit Exit **systat**. (This may be abbreviated to **q**.)

The available displays are:

all Cycle through all displays automatically. At each display, wait some refresh-turns, then switch to the next display. Duration of one refresh-turn is adjustable with the **-w** option, number of refresh-turns can be changed with the **-t** option.

bufcache Display, in the lower window, statistics about the file system buffers. Statistics for each file system that has active buffers include the number of buffers for that file system, the number of active kilobytes in those buffers and the total size of the buffers for that file system.

df Lists disk usage statistics for all filesystems, including the available free space as well as a bar graph indicating the used capacity.

The following commands are specific to the **df** display:

all Displays information for all filesystems, including kernfs, procfs and null-mounts.

some Suppress information about procfs, kernfs and null-mounts (default).

inet.icmp
Display ICMP statistics.

inet.ip Display IPv4 and UDP statistics.

inet.tcp Display TCP statistics.

inet.tcpsyn
Display statistics about the TCP ‘‘syncache’’.

inet6.ip6
Display IPv6 statistics.

ipsec Display IPsec statistics for both IPv4 and v6.

iostat Display, in the lower window, statistics about processor use and disk throughput. Statistics on processor use appear as bar graphs of the amount of time executing in user mode (‘‘user’’), in user mode running low priority processes (‘‘nice’’), in system mode (‘‘system’’), and idle (‘‘idle’’). Statistics on disk throughput show, for each drive, kilobytes of data transferred, number of disk transactions performed, and time spent in disk accesses in milliseconds. This information may be displayed as bar graphs or as rows of numbers which scroll downward. Bar graphs are shown by default;

The following commands are specific to the **iostat** display; the minimum unambiguous prefix may be supplied.

numbers Show the disk I/O statistics in numeric form. Values are displayed in numeric columns which scroll downward.
bars Show the disk I/O statistics in bar graph form (default).
secs Toggle the display of time in disk activity (the default is to not display time).
all Show the read and write statistics combined (default).
rw Show the read and write statistics separately.

mbufs Display, in the lower window, the number of mbufs allocated for particular uses, i.e. data, socket structures, etc.

netstat Display, in the lower window, network connections. By default, network servers awaiting requests are not displayed. Each address is displayed in the format ‘‘host.port’’, with each shown symbolically, when possible. It is possible to have addresses displayed numerically, limit the display to a set of ports, hosts, and/or protocols (the minimum unambiguous prefix may be supplied):

all Toggle the displaying of server processes awaiting requests (this is the equivalent of the **-a** flag to *netstat 1*).
numbers Display network addresses numerically.
names Display network addresses symbolically.
protocol Display only network connections using the indicated protocol (currently either ‘‘tcp’’ or ‘‘udp’’).
ignore [*items*]
Do not display information about connections associated with the specified hosts or ports. Hosts and ports may be specified by name (‘‘vangogh’’, ‘‘ftp’’), or numerically. Host addresses use the Internet dot notation (‘‘128.32.0.9’’). Multiple items may be specified with a single command by separating them with spaces.
display [*items*]
Display information about the connections associated with the specified hosts or ports. As for *ignore*, [*items*] may be names or numbers.
show [*ports*|*hosts*]
Show, on the command line, the currently selected protocols, hosts, and ports. Hosts and ports which are being ignored are prefixed with a ‘!’. If *ports* or *hosts* is supplied as an argument to **show**, then only the requested information will be displayed.
reset Reset the port, host, and protocol matching mechanisms to the default (any protocol, port, or host).

pigs Display, in the lower window, those processes resident in main memory and getting the largest portion of the processor (the default display). When less than 100% of the processor is scheduled to user processes, the remaining time is accounted to the ‘‘idle’’ process.

ps Display, in the lower window, the same information provided by the command ps(1) with the flags **-aux**.

The following command is specific to the **ps** display; the minimum unambiguous prefix may be supplied.

user *name*
Limit the list of processes displayed to those owned by user *name*. If *name* is specified as ‘+’, processes owned by any user are displayed (default).

swap Show information about swap space usage on all the swap areas configured with `swapctl`(8). The first column is the device name of the partition. The next column is the total space available in the partition. The *Used* column indicates the total blocks used so far; the graph shows the percentage of space in use on each partition. If there are more than one swap partition in use, a total line is also shown. Areas known to the kernel, but not in use are shown as not available.

syscall Show per system call statistics. The display consists of several columns of system call name and counts.

In order to stop entries moving around the screen too much, an infinite response filter is applied to the values before they are sorted.

The following commands are specific to the **syscall** display:

sort name Sort display by the syscall name (default).
sort count Sort display by the count of calls or time spent in the calls.
sort syscall
Sort display be syscall number.
show count Show the number of times the system call has be called (default).
show time Show the average amount of time (in arbitrary units) spent in a call of the syscall.

vmstat Take over the entire display and show a (rather crowded) compendium of statistics related to virtual memory usage, process scheduling, device interrupts, system name translation caching, disk I/O etc.

The upper left quadrant of the screen shows the number of users logged in and the load average over the last one, five, and fifteen minute intervals. Below this is a list of the average number of processes (over the last refresh interval) that are runnable ('r'), in page wait ('p'), in disk wait other than paging ('d'), sleeping ('s'), and swapped out but desiring to run ('w'). Below the queue length listing is a numerical listing and a bar graph showing the amount of system (shown as '='), user (shown as '>'), nice (shown as '-'), and idle time (shown as ' ').

To the right of the process statistics is a column that lists the average number of context switches ('Csw'), traps ('Trp'; includes page faults), system calls ('Sys'), interrupts ('Int'), network software interrupts ('Sof'), page faults ('Flt').

Below this are statistics on memory utilization. The first row of the table reports memory usage only among active processes, that is processes that have run in the previous twenty seconds. The second row reports on memory usage of all processes. The first column reports on the number of physical pages claimed by processes. The second column reports the number of pages of memory and swap. The third column gives the number of pages of free memory and swap.

Below the memory display are statistics on name translations. It lists the number of names translated in the previous interval, the number and percentage of the translations that were handled by the system wide name translation cache, and the number and percentage of the translations that were handled by the per process name translation cache.

At the bottom left is the disk usage display. It reports the number of seeks, transfers, number of kilobyte blocks transferred per second averaged over the refresh period of the display (by default, five seconds), and the time spent in disk accesses. If there are more than five disks, and the terminal window has more than 24 lines, the disks display will be flipped so that more of the disk statistics are visible.

Under the date in the upper right hand quadrant are statistics on paging and swapping activity. The first two columns report the average number of pages brought in and out per second over the last refresh interval due to page faults and the paging daemon. The third and fourth columns report the average number of pages brought in and out per second over the last refresh interval due to swap requests initiated by the scheduler. The first row of the display shows the average number of disk transfers per second over the last refresh interval; the second row of the display shows the average number of pages transferred per second over the last refresh interval.

Below the paging statistics is another columns of paging data. From top to bottom, these represent average numbers of copy on write faults ('cow'), object cache lookups ('objlk'), object cache hits ('objht'), pages zero filled on demand ('zfodw'), number zfod's created ('nzfod'), percentage of zfod's used ('%zfod'), number of kernel pages ('kern'), number of wired pages ('wire'), number of active pages ('act'), number of inactive pages ('inact'), number of free pages ('free'), pages freed by daemon ('daefr'), pages freed by exiting processes ('prcfr'), number of pages reactivated from freelist ('react'), scans in page out daemon ('scan'), revolutions of the hand ('hdrev'), and in-transit blocking page faults ('intrn'), per second over the refresh period. Note that the '%zfod' percentage is usually less than 100%, however it may exceed 100% if a large number of requests are actually used long after they were set up during a period when no new pages are being set up. Thus this figure is most interesting when observed over a long time period, such as from boot time (see below on getting such a display).

To the left of the column of paging statistics is a breakdown of the interrupts being handled by the system. At the top of the list is the total interrupts per second over the time interval. The rest of the column breaks down the total on a device by device basis. Only devices that have interrupted at least once since boot time are shown.

Commands to switch between displays may be abbreviated to the minimum unambiguous prefix; for example, ''io'' for ''iostat''. Certain information may be discarded when the screen size is insufficient for display. For example, on a machine with 10 drives the **iostat** bar graph displays only 3 drives on a 24 line terminal. When a bar graph would overflow the allotted screen space it is truncated and the actual value is printed ''over top'' of the bar.

The following commands are common to each display which shows information about disk drives. These commands are used to select a set of drives to report on, should your system have more drives configured than can normally be displayed on the screen.

display [*drives*]
: Display information about the drives indicated. Multiple drives may be specified, separated by spaces.

ignore [*drives*]
: Do not display information about the drives indicated. Multiple drives may be specified, separated by spaces.

drives [*drives*]
: With no arguments, display a list of available drives. With arguments, replace the list of currently displayed drives with the ones specified.

The following commands are specific to the **inet.***, **inet6.***, **ipsec**, **syscall** and **vmstat** displays; the minimum unambiguous prefix may be supplied.

boot Display cumulative statistics since the system was booted.
run Display statistics as a running total from the point this command is given.
time Display statistics averaged over the refresh interval (the default).
zero Reset running statistics to zero.

FILES

`/netbsd`	For the namelist.
`/dev/kmem`	For information in main memory.
`/dev/drum`	For information about swapped out processes.
`/etc/hosts`	For host names.
`/etc/networks`	For network names.
`/etc/services`	For port names.

NOTES

Much of the information that **systat vmstat** uses is obtained from **struct vmmeter cnt**.

SEE ALSO

`df`(1), `netstat`(1), `ps`(1), `top`(1), `vmstat`(1), `iostat`(8), `pstat`(8)

HISTORY

The **systat** program appeared in 4.3BSD.

BUGS

Consumes CPU resources and thus may skew statistics.

Certain displays presume a minimum of 80 characters per line.

The **vmstat** display looks out of place because it is (it was added in as a separate display from what used to be a different program).

NAME

tail — display the last part of a file

SYNOPSIS

tail [**-f** | **-F** | **-r**] [**-b** *number* | **-c** *number* | **-n** *number*] [*file ...*]

DESCRIPTION

The **tail** utility displays the contents of *file* or, by default, its standard input, to the standard output.

The display begins at a byte, line or 512-byte block location in the input. Numbers having a leading plus (‘‘+’’) sign are relative to the beginning of the input, for example, “-c +2” starts the display at the second byte of the input. Numbers having a leading minus (‘‘-’’) sign or no explicit sign are relative to the end of the input, for example, “-n 2” displays the last two lines of the input. The default starting location is “-n 10”, or the last 10 lines of the input.

The options are as follows:

-b *number*
: The location is *number* 512-byte blocks.

-c *number*
: The location is *number* bytes.

-f
: The **-f** option causes **tail** to not stop when end of file is reached, but rather to wait for additional data to be appended to the input. The **-f** option is ignored if the standard input is a pipe, but not if it is a FIFO.

-F
: The **-F** option is the same as the **-f** option, except that every five seconds **tail** will check to see if the file named on the command line has been shortened or moved (it is considered moved if the inode or device number changes) and, if so, it will close the current file, open the filename given, print out the entire contents, and continue to wait for more data to be appended. This option is used to follow log files though rotation by `newsyslog`(8) or similar programs.

-n *number*
: The location is *number* lines.

-r
: The **-r** option causes the input to be displayed in reverse order, by line. Additionally, this option changes the meaning of the **-b**, **-c** and **-n** options. When the **-r** option is specified, these options specify the number of bytes, lines or 512-byte blocks to display, instead of the bytes, lines or blocks from the beginning or end of the input from which to begin the display. The default for the **-r** option is to display all of the input.

If more than a single file is specified, each file is preceded by a header consisting of the string “==> XXX ≤=” where “XXX” is the name of the file.

The **tail** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

`cat`(1), `head`(1), `sed`(1)

STANDARDS

The **tail** utility is expected to be a superset of the IEEE Std 1003.2-1992 (“POSIX.2”) specification. In particular, the **-b**, **-r** and **-F** options are extensions to that standard.

The historic command line syntax of **tail** is supported by this implementation. The only difference between this implementation and historic versions of **tail**, once the command line syntax translation has been done, is that the **-b**, **-c** and **-n** options modify the **-r** option, i.e. ‘‘-r -c 4’’ displays the last 4 charac-

ters of the last line of the input, while the historic tail (using the historic syntax ‘‘-4cr’’) would ignore the **-c** option and display the last 4 lines of the input.

HISTORY

A **tail** command appeared in Version 7 AT&T UNIX.

BUGS

When using the **-F** option, **tail** will not detect a file truncation if, between the truncation and the next check of the file size, data written to the file make it larger than the last known file size.

NAME

talk — talk to another user

SYNOPSIS

talk *person* [*ttyname*]

DESCRIPTION

talk is a visual communication program which copies lines from your terminal to that of another user.

Options available:

person If you wish to talk to someone on your own machine, then *person* is just the person's login name. If you wish to talk to a user on another host, then *person* is of the form `user@host`.

ttyname If you wish to talk to a user who is logged in more than once, the *ttyname* argument may be used to indicate the appropriate terminal name, where *ttyname* is of the form `ttyXX`.

When first called, **talk** sends the message

```
Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine
```

to the user you wish to talk to. At this point, the recipient of the message should reply by typing

```
talk  your_name@your_machine
```

It doesn't matter from which machine the recipient replies, as long as his login-name is the same. Once communication is established, the two parties may type simultaneously, with their output appearing in separate windows. Typing control-L '^L' will cause the screen to be reprinted, while your erase, kill, and word kill characters will behave normally. To exit, just type your interrupt character; **talk** then moves the cursor to the bottom of the screen and restores the terminal to its previous state.

Permission to talk may be denied or granted by use of the `mesg`(1) command. At the outset talking is allowed. Certain commands, in particular `nroff`(1) and `pr`(1), disallow messages in order to prevent messy output.

ENVIRONMENT

If the `TALKHOST` environment variable is set, its value is used as the *hostname* the **talk** packets appear to be originating from. This is useful if you wish to talk to someone on another machine and your internal hostname does not resolve to the address of your external interface as seen from the other machine.

FILES

`/etc/hosts` to find the recipient's machine
`/var/run/utmp` to find the recipient's tty

SEE ALSO

`mail`(1), `mesg`(1), `who`(1), `write`(1)

HISTORY

The **talk** command appeared in 4.2BSD.

BUGS

The version of **talk** released with 4.3BSD uses a protocol that is incompatible with the protocol used in the version released with 4.2BSD.

NAME

tar — tape archiver

SYNOPSIS

tar [-]{crtux}[**-014578befHhjklmOoPpqSvwXZz**] [*archive*] [*blocksize*]
[**-C** *directory*] [**-s** *replstr*] [**-T** *file*] [*file ...*]

DESCRIPTION

The **tar** command creates, adds files to, or extracts files from an archive file in "tar" format. A tar archive is often stored on a magnetic tape, but can be stored equally well on a floppy, CD-ROM, or in a regular disk file.

One of the following flags must be present:

-c, **--create**
: Create new archive, or overwrite an existing archive, adding the specified files to it.

-r, **--append**
: Append the named new files to existing archive. Note that this will only work on media on which an end-of-file mark can be overwritten.

-t, **--list** List contents of archive. If any files are named on the command line, only those files will be listed.

-u, **--update**
: Alias for **-r**.

-x, **--extract**, **--get**
: Extract files from archive. If any files are named on the command line, only those files will be extracted from the archive. If more than one copy of a file exists in the archive, later copies will overwrite earlier copies during extraction. The file mode and modification time are preserved if possible. The file mode is subject to modification by the `umask`(2).

In addition to the flags mentioned above, any of the following flags may be used:

-b *blocking factor*, **--block-size** *blocking factor*
: Set blocking factor to use for the archive. **tar** uses 512 byte blocks. The default is 20, the maximum is 126. Archives with a blocking factor larger 63 violate the POSIX standard and will not be portable to all systems.

-e Stop after first error.

-f *archive*, **--file** *archive*
: Filename where the archive is stored. Defaults to `/dev/rst0`. If the archive is of the form: *[[user@]host:]file* then the archive will be processed using `rmt`(8).

-h, **--dereference**
: Follow symbolic links as if they were normal files or directories.

-j, --bzip2, --bunzip2
: Use `bzip2`(1) for compression of the archive. This option is a GNU extension.

-k, **--keep-old-files**
: Keep existing files; don't overwrite them from archive.

-l, **--one-file-system**
: Do not descend across mount points.

-m, **--modification-time**
Do not preserve modification time.

-O When creating and appending to an archive, write old-style (non-POSIX) archives. When extracting from an archive, extract to standard output.

-o, **--portability**, **--old-archive**
Don't write directory information that the older (V7) style **tar** is unable to decode. This implies the **-O** flag.

-p, **--preserve-permissions**, **--preserve**
Preserve user and group ID as well as file mode regardless of the current umask(2). The setuid and setgid bits are only preserved if the user is the superuser. Only meaningful in conjunction with the **-x** flag.

-q, **--fast-read**
Select the first archive member that matches each *pattern* operand. No more than one archive member is matched for each *pattern*. When members of type directory are matched, the file hierarchy rooted at that directory is also matched.

-S, **--sparse**
This flag has no effect as **tar** always generates sparse files.

-s *replstr*
Modify the file or archive member names specified by the *pattern* or *file* operands according to the substitution expression *replstr*, using the syntax of the ed(1) utility regular expressions. The format of these regular expressions are:

```
/old/new/[gps]
```

As in ed(1), **old** is a basic regular expression and **new** can contain an ampersand (&), \n (where n is a digit) back-references, or subexpression matching. The **old** string may also contain ⟨newline⟩ characters. Any non-null character can be used as a delimiter (/ is shown here). Multiple **-s** expressions can be specified. The expressions are applied in the order they are specified on the command line, terminating with the first successful substitution. The optional trailing **g** continues to apply the substitution expression to the pathname substring which starts with the first character following the end of the last successful substitution. The first unsuccessful substitution stops the operation of the **g** option. The optional trailing **p** will cause the final result of a successful substitution to be written to standard error in the following format:

```
<original pathname> >> <new pathname>
```

File or archive member names that substitute to the empty string are not selected and will be skipped. The substitutions are applied by default to the destination hard and symbolic links. The optional trailing **s** prevents the substitutions from being performed on symbolic link destinations.

-v Verbose operation mode.

-w, **--interactive**, **--confirmation**
Interactively rename files. This option causes **tar** to prompt the user for the filename to use when storing or extracting files in an archive.

-z, **--gzip**, **--gunzip**
Compress archive using gzip.

-B, **--read-full-blocks**
Reassemble small reads into full blocks (For reading from 4.2BSD pipes).

-C *directory*, **--directory** *directory*
This is a positional argument which sets the working directory for the following files. When extracting, files will be extracted into the specified directory; when creating, the specified files will be matched from the directory. This argument and its parameter may also appear in a file list specified by **-T**.

-H Only follow symlinks given on command line.

Note SysVr3/i386 picked up ISC/SCO UNIX compatabilty which implemented "**-F** *file*" which was defined as obtaining a list of command line switches and files on which to operate from the specified file, but SunOS-5 uses "**-I** *file*" because they use '**-F**' to mean something else. We might someday provide SunOS-5 compatability but it makes little sense to confuse things with ISC/SCO compatability.

-P, **--absolute-paths**
Do not strip leading slashes ('/') from pathnames. The default is to strip leading slashes.

-T *file*, **--files-from** *file*
Read the names of files to archive or extract from the given file, one per line. A line may also specify the positional argument "**-C** *directory*".

-X *file*, **--exclude-from** *file*
Exclude files listed in the given file.

Note that it would be more standard to use this option to mean ''do not cross filesystem mount points.''

-Z, **--compress**, **--uncompress**
Compress archive using compress.

--strict Do not enable GNU tar extensions such as long filenames and long link names.

--atime-preserve
Preserve file access times.

--chroot **chroot**() to the current directory before extracting files. Use with **-x** and **-h** to make absolute symlinks relative to the current directory.

--unlink Ignored, only accepted for compatibility with other **tar** implementations. **tar** always unlinks files before creating them.

--use-compress-program *program*
Use the named program as the program to decompress the input.

--force-local
Do not interpret filenames that contain a ':' as remote files.

--insecure
Normally **tar** ignores filenames that contain ".." as a path component. With this option, files that contain ".." can be processed.

--no-recursion
Cause files of type directory being copied or archived, or archive members of type directory being extracted, to match only the directory file or archive member and not the file hierarchy rooted at the directory.

The options [**-014578**] can be used to select one of the compiled-in backup devices, `/dev/rstN`.

FILES

`/dev/rst0` default archive name

DIAGNOSTICS

`tar` will exit with one of the following values:

0 All files were processed successfully.

1 An error occurred.

Whenever **`tar`** cannot create a file or a link when extracting an archive or cannot find a file while writing an archive, or cannot preserve the user ID, group ID, file mode, or access and modification times when the **`-p`** option is specified, a diagnostic message is written to standard error and a non-zero exit value will be returned, but processing will continue. In the case where **`tar`** cannot create a link to a file, **`tar`** will not create a second copy of the file.

If the extraction of a file from an archive is prematurely terminated by a signal or error, **`tar`** may have only partially extracted the file the user wanted. Additionally, the file modes of extracted files and directories may have incorrect file bits, and the modification and access times may be wrong.

If the creation of an archive is prematurely terminated by a signal or error, **`tar`** may have only partially created the archive which may violate the specific archive format specification.

SEE ALSO

`cpio`(1), `pax`(1)

HISTORY

A **`tar`** command first appeared in Version 7 AT&T UNIX.

AUTHORS

Keith Muller at the University of California, San Diego.

NAME

tcopy — copy and/or verify mag tapes

SYNOPSIS

tcopy [**-cvx**] [**-s** *maxblk*] [*src* [*dest*]]

DESCRIPTION

tcopy is designed to copy magnetic tapes. The only assumption made about the tape is that there are two tape marks at the end. **tcopy** with only a source tape (*/dev/rst0* by default) specified will print information about the sizes of records and tape files. If a destination is specified a copy will be made of the source tape. The blocking on the destination tape will be identical to that used on the source tape. Copying a tape will yield the same output as if just printing the sizes.

Options:

-c Copy *src* to *dest* and then verify that the two tapes are identical.

-s *maxblk*
Specify a maximum block size, *maxblk*.

-v Given the two tapes, *src* and *dest* verify that they are identical.

-x Output all informational messages to the standard error. This option is useful when *dest* is /dev/stdout.

SEE ALSO

mtio(4)

HISTORY

The **tcopy** command appeared in 4.3BSD.

NAME

tee — pipe fitting

SYNOPSIS

tee [**-ai**] [*file ...*]

DESCRIPTION

The **tee** utility copies standard input to standard output, making a copy in zero or more files. The output is unbuffered.

The following options are available:

-a Append the output to the files rather than overwriting them.

-i Ignore the SIGINT signal.

The following operands are available:

file A pathname of an output *file*.

The **tee** utility takes the default action for all signals, except in the event of the **-i** option.

The **tee** utility exits 0 on success, and >0 if an error occurs.

STANDARDS

The **tee** function is expected to be POSIX IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

telnet — user interface to the TELNET protocol

SYNOPSIS

telnet [**-78EFKLacdfrx**] [**-S** *tos*] [**-X** *authtype*] [**-e** *escapechar*] [**-k** *realm*] [**-l** *user*] [**-n** *tracefile*] [*host* [port]]

DESCRIPTION

The **telnet** command is used to communicate with another host using the TELNET protocol. If **telnet** is invoked without the *host* argument, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command with those arguments.

Options:

-8 Specifies an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.

-7 Do not try to negotiate TELNET BINARY option.

-E Stops any character from being recognized as an escape character.

-F If Kerberos V5 authentication is being used, the **-F** option allows the local credentials to be forwarded to the remote system, including any credentials that have already been forwarded into the local environment.

-K Specifies no automatic login to the remote system.

-L Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-S *tos*
Sets the IP type-of-service (TOS) option for the telnet connection to the value *tos*, which can be a numeric TOS value or, on systems that support it, a symbolic TOS name found in the /etc/iptos file.

-X *atype*
Disables the *atype* type of authentication.

-a Attempt automatic login. Currently, this sends the user name via the USER variable of the ENVIRON option if supported by the remote system. The name used is that of the current user as returned by getlogin(2) if it agrees with the current user ID, otherwise it is the name associated with the user ID.

-c Disables the reading of the user's .telnetrc file. (See the **toggle skiprc** command on this man page.)

-d Sets the initial value of the **debug** toggle to TRUE

-e *escape char*
Sets the initial **telnet telnet** escape character to *escape char*. If *escape char* is omitted, then there will be no escape character.

-f If Kerberos V5 authentication is being used, the **-f** option allows the local credentials to be forwarded to the remote system.

-k *realm*
If Kerberos authentication is being used, the **-k** option requests that telnet obtain tickets for the remote host in realm realm instead of the remote host's realm, as determined by krb_realmofhost(3).

-l *user*
When connecting to the remote system, if the remote system understands the ENVIRON option, then *user* will be sent to the remote system as the value for the variable USER. This option implies the **-a** option. This option may also be used with the **open** command.

-n *tracefile*
Opens *tracefile* for recording trace information. See the **set tracefile** command below.

-r Specifies a user interface similar to rlogin(1). In this mode, the escape character is set to the tilde (~) character, unless modified by the -e option.

-x Turn on encryption of the data stream. When this option is turned on, will exit with an error if authentication cannot be negotiated or if encryption cannot be turned on.

host Indicates the official name, an alias, or the Internet address of a remote host.

port Indicates a port number (address of an application). If a number is not specified, the default **telnet** port is used.

When in rlogin mode, a line of the form ~. disconnects from the remote host; ~ is the telnet escape character. Similarly, the line ~^Z suspends the telnet session. The line ~^] escapes to the normal telnet escape prompt.

Once a connection has been opened, **telnet** will attempt to enable the TELNET LINEMODE option. If this fails, then **telnet** will revert to one of two input modes: either "character at a time" or "old line by line" depending on what the remote system supports.

When LINEMODE is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing is to be disabled, the remote system will relay that information. The remote system will also relay changes to any special characters that happen on the remote system, so that they can take effect on the local system.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "old line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially "^E") may be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

If the LINEMODE option is enabled, or if the **localchars** toggle is TRUE (the default for "old line by line"; see below), the user's **quit**, **intr**, and **flush** characters are trapped locally, and sent as TELNET protocol sequences to the remote side. If LINEMODE has ever been enabled, then the user's **susp** and **eof** are also sent as TELNET protocol sequences, and **quit** is sent as a TELNET ABORT instead of BREAK There are options (see **toggle autoflush** and **toggle autosynch** below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of **quit** and **intr**).

While connected to a remote host, **telnet** command mode may be entered by typing the **telnet** "escape character" (initially "^]"). When in command mode, the normal terminal editing conventions are available.

The following **telnet** commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the **mode**, **set**, **toggle**, **unset**, **slc**, **environ**, and **display** commands).

auth *argument ...*
The auth command manipulates the information sent through the TELNET AUTHENTICATE option. Valid arguments for the auth command are as follows:

disable *type* Disables the specified type of authentication. To obtain a list of available types, use the **auth disable ?** command.

enable *type* Enables the specified type of authentication. To obtain a list of available types, use the **auth enable ?** command.

status Lists the current status of the various types of authentication.

close Close a TELNET session and return to command mode.

display *argument ...*
Displays all, or some, of the **set** and **toggle** values (see below).

encrypt *argument ...*
The encrypt command manipulates the information sent through the TELNET ENCRYPT option.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside of the United States and Canada.

Valid arguments for the encrypt command are as follows:

disable *type* [**input** | **output**]
Disables the specified type of encryption. If you omit the input and output, both input and output are disabled. To obtain a list of available types, use the **encrypt disable ?** command.

enable *type* [**input** | **output**]
Enables the specified type of encryption. If you omit input and output, both input and output are enabled. To obtain a list of available types, use the **encrypt enable ?** command.

input This is the same as the **encrypt start input** command.

-input This is the same as the **encrypt stop input** command.

output This is the same as the **encrypt start output** command.

-output This is the same as the **encrypt stop output** command.

start [**input** | **output**]
Attempts to start encryption. If you omit **input** and **output**, both input and output are enabled. To obtain a list of available types, use the **encrypt enable ?** command.

status Lists the current status of encryption.

stop [**input** | **output**]
Stops encryption. If you omit input and output, encryption is on both input and output.

type *type* Sets the default type of encryption to be used with later **encrypt start** or **encrypt stop** commands.

environ *arguments ...*
The **environ** command is used to manipulate the the variables that my be sent through the TELNET ENVIRON option. The initial set of variables is taken from the users environment, with only the DISPLAY and PRINTER variables being exported by default. The USER variable is also exported if the **-a** or **-l** options are used.

Valid arguments for the **environ** command are:

define *variable value*
Define the variable *variable* to have a value of *value*. Any variables defined by this command are automatically exported. The *value* may be enclosed in single or double quotes so that tabs and spaces may be included.

undefine *variable*
Remove *variable* from the list of environment variables.

export *variable*
Mark the variable *variable* to be exported to the remote side.

unexport *variable*
Mark the variable *variable* to not be exported unless explicitly asked for by the remote side.

list List the current set of environment variables. Those marked with a * will be sent automatically, other variables will only be sent if explicitly requested.

? Prints out help information for the **environ** command.

logout Sends the TELNET LOGOUT option to the remote side. This command is similar to a **close** command; however, if the remote side does not support the LOGOUT option, nothing happens. If, however, the remote side does support the LOGOUT option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user's session for later reattachment, the logout argument indicates that you should terminate the session immediately.

mode *type* *Type* is one of several options, depending on the state of the TELNET session. The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

character Disable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then enter "character at a time" mode.

line Enable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then attempt to enter "old-line-by-line" mode.

isig (**-isig**)
Attempt to enable (disable) the TRAPSIG mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

edit (**-edit**)
Attempt to enable (disable) the EDIT mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

softtabs (**-softtabs**)
Attempt to enable (disable) the SOFT_TAB mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

litecho (**-litecho**)
Attempt to enable (disable) the LIT_ECHO mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

? Prints out help information for the **mode** command.

open *host* [**-l** *user*] [[**-**]*port*]
Open a connection to the named host. If no port number is specified, **telnet** will attempt to contact a TELNET server at the default port. The host specification may be either a host name

(see `hosts`(5)) or an Internet address specified in the "dot notation" (see `inet`(3)). The [**-l**] option may be used to specify the user name to be passed to the remote system via the `ENVIRON` option. When connecting to a non-standard port, **telnet** omits any automatic initiation of TELNET options. When the port number is preceded by a minus sign, the initial option negotiation is done. After establishing a connection, the file `.telnetrc` in the users home directory is opened. Lines beginning with a # are comment lines. Blank lines are ignored. Lines that begin without white space are the start of a machine entry. The first thing on the line is the name of the machine that is being connected to. The rest of the line, and successive lines that begin with white space are assumed to be **telnet** commands and are processed as if they had been typed in manually to the **telnet** command prompt.

quit Close any open TELNET session and exit **telnet**. An end of file (in command mode) will also close a session and exit.

send *arguments*

Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

abort Sends the `TELNET ABORT` (Abort processes) sequence.

ao Sends the `TELNET AO` (Abort Output) sequence, which should cause the remote system to flush all output *from* the remote system *to* the user's terminal.

ayt Sends the `TELNET AYT` (Are You There) sequence, to which the remote system may or may not choose to respond.

brk Sends the `TELNET BRK` (Break) sequence, which may have significance to the remote system.

ec Sends the `TELNET EC` (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the `TELNET EL` (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

eof Sends the `TELNET EOF` (End Of File) sequence.

eor Sends the `TELNET EOR` (End of Record) sequence.

escape Sends the current **telnet** escape character (initially "^").

ga Sends the `TELNET GA` (Go Ahead) sequence, which likely has no significance to the remote system.

getstatus

If the remote side supports the `TELNET STATUS` command, **getstatus** will send the subnegotiation to request that the server send its current option status.

ip Sends the `TELNET IP` (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

nop Sends the `TELNET NOP` (No OPeration) sequence.

susp Sends the `TELNET SUSP` (SUSPend process) sequence.

synch Sends the `TELNET SYNCH` sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2BSD system -- if it doesn't work, a lower case "r" may be echoed on the terminal).

do *cmd*

dont *cmd*

will *cmd*

wont *cmd*

Sends the TELNET DO *cmd* sequence. *Cmd* can be either a decimal number between 0 and 255, or a symbolic name for a specific TELNET command. *Cmd* can also be either **help** or **?** to print out help information, including a list of known symbolic names.

? Prints out help information for the **send** command.

set *argument value*

unset *argument value*

The **set** command will set any one of a number of **telnet** variables to a specific value or to TRUE. The special value **off** turns off the function associated with the variable, this is equivalent to using the **unset** command. The **unset** command will disable or set to FALSE any of the specified functions. The values of variables may be interrogated with the **display** command. The variables which may be set or unset, but not toggled, are listed here. In addition, any of the variables for the **toggle** command may be explicitly set or unset using the **set** and **unset** commands.

ayt If TELNET is in localchars mode, or LINEMODE is enabled, and the status character is typed, a TELNET AYT sequence (see **send ayt** preceding) is sent to the remote host. The initial value for the "Are You There" character is the terminal's status character.

echo This is the value (initially "^E") which, when in "line by line" mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

eof If **telnet** is operating in LINEMODE or "old line by line" mode, entering this character as the first character on a line will cause this character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's **eof** character.

erase If **telnet** is in **localchars** mode (see **toggle localchars** below), **and** if **telnet** is operating in "character at a time" mode, then when this character is typed, a TELNET EC sequence (see **send ec** above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

escape This is the **telnet** escape character (initially "^[") which causes entry into **telnet** command mode (when connected to a remote system).

flushoutput

If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **flushoutput** character is typed, a TELNET AO sequence (see **send ao** above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's **flush** character.

forw1

forw2 If TELNET is operating in LINEMODE, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's eol and eol2 characters.

interrupt
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **interrupt** character is typed, a TELNET IP sequence (see **send ip** above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

kill If **telnet** is in **localchars** mode (see **toggle localchars** below), **and** if **telnet** is operating in "character at a time" mode, then when this character is typed, a TELNET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

lnext If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **lnext** character. The initial value for the lnext character is taken to be the terminal's **lnext** character.

quit If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **quit** character is typed, a TELNET BRK sequence (see **send brk** above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

reprint
If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **reprint** character. The initial value for the reprint character is taken to be the terminal's **reprint** character.

rlogin This is the rlogin escape character. If set, the normal TELNET escape character is ignored unless it is preceded by this character at the beginning of a line. This character, at the beginning of a line followed by a "." closes the connection; when followed by a ^Z it suspends the telnet command. The initial state is to disable the rlogin escape character.

start If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's **start** character. The initial value for the kill character is taken to be the terminal's **start** character.

stop If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's **stop** character. The initial value for the kill character is taken to be the terminal's **stop** character.

susp If **telnet** is in **localchars** mode, or LINEMODE is enabled, and the **suspend** character is typed, a TELNET SUSP sequence (see **send susp** above) is sent to the remote host. The initial value for the suspend character is taken to be the terminal's **suspend** character.

tracefile
This is the file to which the output, caused by **netdata** or **option** tracing being TRUE, will be written. If it is set to "**-**", then tracing information will be written to standard output (the default).

worderase
If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **worderase** character. The initial value for the worderase character is taken to be the terminal's **worderase** character.

? Displays the legal **set** (**unset**) commands.

slc *state* The **slc** command (Set Local Characters) is used to set or change the state of the the special characters when the TELNET LINEMODE option has been enabled. Special characters are characters that get mapped to TELNET commands sequences (like **ip** or **quit**) or line editing characters (like **erase** and **kill**). By default, the local special characters are exported.

check Verify the current settings for the current special characters. The remote side is requested to send all the current special character settings, and if there are any discrepancies with the local side, the local side will switch to the remote value.

export Switch to the local defaults for the special characters. The local default characters are those of the local terminal at the time when **telnet** was started.

import Switch to the remote defaults for the special characters. The remote default characters are those of the remote system at the time when the TELNET connection was established.

? Prints out help information for the **slc** command.

status Show the current status of **telnet**. This includes the peer one is connected to, as well as the current mode.

toggle *arguments ...*

Toggle (between TRUE and FALSE) various flags that control how **telnet** responds to events. These flags may be set explicitly to TRUE or FALSE using the **set** and **unset** commands listed above. More than one argument may be specified. The state of these flags may be interrogated with the **display** command. Valid arguments are:

authdebug Turns on debugging information for the authentication code.

autoflush If **autoflush** and **localchars** are both TRUE, then when the **ao**, or **quit** characters are recognized (and transformed into TELNET sequences; see **set** above for details), **telnet** refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET TIMING MARK option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE if the terminal user had not done an "stty noflsh", otherwise FALSE (see stty(1)).

autodecrypt

When the TELNET ENCRYPT option is negotiated, by default the actual encryption (decryption) of the data stream does not start automatically. The autoencrypt (autodecrypt) command states that encryption of the output (input) stream should be enabled as soon as possible.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside the United States and Canada.

autologin If the remote side supports the TELNET AUTHENTICATION option TELNET attempts to use it to perform automatic authentication. If the AUTHENTICATION option is not supported, the user's login name are propagated through the TELNET ENVIRON option. This command is the same as specifying *a* option on the **open** command.

autosynch If **autosynch** and **localchars** are both TRUE, then when either the **intr** or **quit** characters is typed (see **set** above for descriptions of the **intr** and **quit** characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure **should** cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of

this toggle is FALSE.

binary Enable or disable the TELNET BINARY option on both input and output.

inbinary Enable or disable the TELNET BINARY option on input.

outbinary Enable or disable the TELNET BINARY option on output.

crlf If this is TRUE, then carriage returns will be sent as <CR><LF>. If this is FALSE, then carriage returns will be send as <CR><NUL>. The initial value for this toggle is FALSE.

crmod Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug Toggles socket level debugging (useful only to the **super user**). The initial value for this toggle is FALSE.

encdebug Turns on debugging information for the encryption code.

localchars If this is TRUE, then the **flush**, **interrupt**, **quit**, **erase**, and **kill** characters (see **set** above) are recognized locally, and transformed into (hopefully) appropriate TELNET control sequences (respectively **ao**, **ip**, **brk**, **ec**, and **el**; see **send** above). The initial value for this toggle is TRUE in "old line by line" mode, and FALSE in "character at a time" mode. When the LINEMODE option is enabled, the value of **localchars** is ignored, and assumed to always be TRUE. If LINEMODE has ever been enabled, then **quit** is sent as **abort**, and **eof** and **suspend** are sent as **eof** and **susp**, see **send** above).

netdata Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

options Toggles the display of some internal **telnet** protocol processing (having to do with TELNET options). The initial value for this toggle is FALSE.

prettydump When the **netdata** toggle is enabled, if **prettydump** is enabled the output from the **netdata** command will be formatted in a more user readable format. Spaces are put between each character in the output, and the beginning of any TELNET escape sequence is preceded by a '*' to aid in locating them.

skiprc When the skiprc toggle is TRUE, TELNET skips the reading of the .telnetrc file in the users home directory when connections are opened. The initial value for this toggle is FALSE.

termdata Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is FALSE.

verbose_encrypt
When the **verbose_encrypt** toggle is TRUE, TELNET prints out a message each time encryption is enabled or disabled. The initial value for this toggle is FALSE. Note: Because of export controls, data encryption is not supported outside of the United States and Canada.

? Displays the legal **toggle** commands.

z Suspend **telnet**. This command only works when the user is using the csh(1).

! [*command*]
Execute a single command in a subshell on the local system. If **command** is omitted, then an interactive subshell is invoked.

? [*command*]
Get help. With no arguments, **telnet** prints a help summary. If a command is specified, **telnet** will print the help information for just that command.

ENVIRONMENT

Telnet uses at least the HOME, SHELL, DISPLAY, and TERM environment variables. Other environment variables may be propagated to the other side via the TELNET ENVIRON option.

FILES

~/.telnetrc user customized telnet startup values

HISTORY

The **Telnet** command appeared in 4.2BSD.

NOTES

On some remote systems, echo has to be turned off manually when in "old line by line" mode.

In "old line by line" mode or LINEMODE the terminal's **eof** character is only recognized (and sent to the remote system) when it is the first character on a line.

NAME

telnet — user interface to the TELNET protocol

SYNOPSIS

telnet [**-4**] [**-6**] [**-8**] [**-E**] [**-F**] [**-K**] [**-L**] [**-N**] [**-S** *tos*] [**-X** *authtype*] [**-a**] [**-c**] [**-d**] [**-e** *escapechar*] [**-f**] [**-k** *realm*] [**-l** *user*] [**-n** *tracefile*] [**-P** *policy*] [**-r**] [**-x**] [*host* [*port*]]

DESCRIPTION

The **telnet** command is used to communicate with another host using the TELNET protocol. If **telnet** is invoked without the *host* argument, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. If it is invoked with arguments, it performs an **open** command with those arguments.

Options:

-4 Forces **telnet** to use IPv4 addresses only.

-6 Forces **telnet** to use IPv6 addresses only.

-8 Specifies an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.

-E Stops any character from being recognized as an escape character.

-F If Kerberos V5 authentication is being used, the **-F** option allows the local credentials to be forwarded to the remote system, including any credentials that have already been forwarded into the local environment.

-K Specifies no automatic login to the remote system.

-L Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-N Numeric host address. No attempt will be made to look up symbolic names for host addresses.

-S *tos*

Sets the IP type-of-service (TOS) option for the telnet connection to the value *tos*, which can be a numeric TOS value or, on systems that support it, a symbolic TOS name found in the /etc/iptos file.

-X *atype*

Disables the *atype* type of authentication.

-a Attempt automatic login. Currently, this sends the user name via the USER variable of the ENVIRON option if supported by the remote system. The name used is that of the current user as returned by getlogin(2) if it agrees with the current user ID, otherwise it is the name associated with the user ID.

-c Disables the reading of the user's .telnetrc file. (See the **toggle skiprc** command on this man page.)

-d Sets the initial value of the **debug** toggle to TRUE.

-e *escape char*

Sets the initial **telnet** escape character to *escape char*. If *escape char* is omitted, then there will be no escape character.

-f If Kerberos V5 authentication is being used, the **-f** option allows the local credentials to be forwarded to the remote system.

-k *realm*
If Kerberos authentication is being used, the **-k** option requests that telnet obtain tickets for the remote host in realm *realm* instead of the remote host's realm, as determined by krb_realmofhost(3).

-l *user*
When connecting to the remote system, if the remote system understands the ENVIRON option, then *user* will be sent to the remote system as the value for the variable USER. This option implies the **-a** option. This option may also be used with the **open** command.

-n *tracefile*
Opens *tracefile* for recording trace information. See the **set tracefile** command below.

-P *policy*
Use IPsec policy specification string *policy*, for the connections. See ipsec_set_policy(3) for details.

-r Specifies a user interface similar to rlogin(1). In this mode, the escape character is set to the tilde (~) character, unless modified by the **-e** option.

-x Turns on encryption of the data stream if possible. This option is not available outside of the United States and Canada.

host Indicates the official name, an alias, or the Internet address of a remote host.

port Indicates a port number (address of an application). If a number is not specified, the default **telnet** port is used.

When in rlogin mode, a line of the form ~. disconnects from the remote host; ~ is the telnet escape character. Similarly, the line ~^Z suspends the telnet session. The line ~^] escapes to the normal telnet escape prompt.

Once a connection has been opened, **telnet** will attempt to enable the TELNET LINEMODE option. If this fails, then **telnet** will revert to one of two input modes: either "character at a time" or "old line by line" depending on what the remote system supports.

When LINEMODE is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing is to be disabled, the remote system will relay that information. The remote system will also relay changes to any special characters that happen on the remote system, so that they can take effect on the local system.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "old line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially "^E") may be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

If the LINEMODE option is enabled, or if the **localchars** toggle is TRUE (the default for "old line by line"; see below), the user's **quit**, **intr**, and **flush** characters are trapped locally, and sent as TELNET protocol sequences to the remote side. If LINEMODE has ever been enabled, then the user's **susp** and **eof** are also sent as TELNET protocol sequences, and **quit** is sent as a TELNET ABORT instead of BREAK. There are options (see **toggle autoflush** and **toggle autosynch** below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of **quit** and **intr**).

While connected to a remote host, **telnet** command mode may be entered by typing the **telnet** "escape character" (initially "^]"). When in command mode, the normal terminal editing conventions are available.

The following **telnet** commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the **mode**, **set**, **toggle**, **unset**, **slc**, **environ**, and

display commands).

auth *argument ...*

The auth command manipulates the information sent through the TELNET AUTHENTICATE option. Valid arguments for the auth command are as follows:

disable *type* Disables the specified type of authentication. To obtain a list of available types, use the **auth disable ?** command.

enable *type* Enables the specified type of authentication. To obtain a list of available types, use the **auth enable ?** command.

status Lists the current status of the various types of authentication.

close Close a TELNET session and return to command mode.

display *argument ...*

Displays all, or some, of the **set** and **toggle** values (see below).

encrypt *argument ...*

The encrypt command manipulates the information sent through the TELNET ENCRYPT option.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside of the United States and Canada.

Valid arguments for the encrypt command are:

disable *type* **[input|output]**

Disables the specified type of encryption. If you omit the input and output, both input and output are disabled. To obtain a list of available types, use the **encrypt disable ?** command.

enable *type* **[input|output]**

Enables the specified type of encryption. If you omit input and output, both input and output are enabled. To obtain a list of available types, use the **encrypt enable ?** command.

input This is the same as the **encrypt start input** command.

-input This is the same as the **encrypt stop input** command.

output This is the same as the **encrypt start output** command.

-output This is the same as the **encrypt stop output** command.

start [input|output]

Attempts to start encryption. If you omit **input** and **output**, both input and output are enabled. To obtain a list of available types, use the **encrypt enable ?** command.

status Lists the current status of encryption.

stop [input|output]

Stops encryption. If you omit input and output, encryption is on both input and output.

type *type* Sets the default type of encryption to be used with later **encrypt start** or **encrypt stop** commands.

environ *arguments ...*

The **environ** command is used to manipulate the variables that may be sent through the TELNET ENVIRON option. The initial set of variables is taken from the users environment, with only the DISPLAY and PRINTER variables being exported by default. The USER variable is also exported if the **-a** or **-l** options are used.

Valid arguments for the **environ** command are:

define *variable value*
Define the variable *variable* to have a value of *value*. Any variables defined by this command are automatically exported. The *value* may be enclosed in single or double quotes so that tabs and spaces may be included.

undefine *variable*
Remove *variable* from the list of environment variables.

export *variable*
Mark the variable *variable* to be exported to the remote side.

unexport *variable*
Mark the variable *variable* to not be exported unless explicitly asked for by the remote side.

list List the current set of environment variables. Those marked with a * will be sent automatically, other variables will only be sent if explicitly requested.

? Prints out help information for the **environ** command.

logout Sends the TELNET LOGOUT option to the remote side. This command is similar to a **close** command; however, if the remote side does not support the LOGOUT option, nothing happens. If, however, the remote side does support the LOGOUT option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user's session for later reattachment, the logout argument indicates that you should terminate the session immediately.

mode *type* *Type* is one of several options, depending on the state of the TELNET session. The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered.

character Disable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then enter "character at a time" mode.

line Enable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then attempt to enter "old-line-by-line" mode.

isig (**-isig**)
Attempt to enable (disable) the TRAPSIG mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

edit (**-edit**)
Attempt to enable (disable) the EDIT mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

softtabs (**-softtabs**)
Attempt to enable (disable) the SOFT_TAB mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

litecho (**-litecho**)
Attempt to enable (disable) the LIT_ECHO mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

? Prints out help information for the **mode** command.

open *host* [**-l** *user*] [**-a**] [[-]*port*]
Open a connection to the named host. If no port number is specified, **telnet** will attempt to contact a TELNET server at the default port. The host specification may be either a host name (see hosts(5)) or an Internet address specified in the "dot notation" (see inet(3)). The **-l** option may be used to specify the user name to be passed to the remote system via the ENVIRON option. If a port is specified **telnet** omits any automatic initialisation of TELNET options. When the port number is preceded by a minus sign, the initial option negotiation is done.

After establishing a connection, the file .telnetrc in the user's home directory is read. Lines beginning with a # are comment lines. Blank lines are ignored. Lines that begin without white space are the start of a machine entry. The first thing on such a line is a string identifying the machine that is being connected to. It may be the hostname or numeric address specified as the argument *host*, the canonical name of that string as determined by getaddrinfo(3), or the string "DEFAULT" indicating all hosts. The rest of the line, and successive lines that begin with white space are assumed to be **telnet** commands and are processed as if they had been typed in manually to the **telnet** command prompt.

quit Close any open TELNET session and exit **telnet**. An end of file (in command mode) will also close a session and exit.

send *arguments*
Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

abort Sends the TELNET ABORT (Abort processes) sequence.

ao Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output *from* the remote system *to* the user's terminal.

ayt Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

brk Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ec Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

eof Sends the TELNET EOF (End Of File) sequence.

eor Sends the TELNET EOR (End of Record) sequence.

escape Sends the current **telnet** escape character (initially "^").

ga Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

getstatus
If the remote side supports the TELNET STATUS command, **getstatus** will send the subnegotiation to request that the server send its current option status.

ip Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

nop Sends the TELNET NOP (No OPeration) sequence.

susp Sends the TELNET SUSP (SUSPend process) sequence.

synch Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2BSD system -- if it doesn't work, a lower case "r" may be echoed on the terminal).

do *cmd*

dont *cmd*

will *cmd*

wont *cmd*

Sends the TELNET DO *cmd* sequence. *Cmd* can be either a decimal number between 0 and 255, or a symbolic name for a specific TELNET command. *Cmd* can also be either **help** or **?** to print out help information, including a list of known symbolic names.

? Prints out help information for the **send** command.

set *argument value*

unset *argument value*

The **set** command will set any one of a number of **telnet** variables to a specific value or to TRUE. The special value **off** turns off the function associated with the variable, this is equivalent to using the **unset** command. The **unset** command will disable or set to FALSE any of the specified functions. The values of variables may be interrogated with the **display** command. The variables which may be set or unset, but not toggled, are listed here. In addition, any of the variables for the **toggle** command may be explicitly set or unset using the **set** and **unset** commands.

ayt If TELNET is in localchars mode, or LINEMODE is enabled, and the status character is typed, a TELNET AYT sequence (see **send ayt** above) is sent to the remote host. The initial value for the "Are You There" character is the terminal's status character.

echo This is the value (initially "^E") which, when in "line by line" mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

eof If **telnet** is operating in LINEMODE or "old line by line" mode, entering this character as the first character on a line will cause this character to be sent to the remote system. The initial value of the eof character is taken to be the terminal's **eof** character.

erase If **telnet** is in **localchars** mode (see **toggle localchars** below), **and** if **telnet** is operating in "character at a time" mode, then when this character is typed, a TELNET EC sequence (see **send ec** above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

escape This is the **telnet** escape character (initially "^[") which causes entry into **telnet** command mode (when connected to a remote system).

flushoutput
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **flushoutput** character is typed, a TELNET AO sequence (see **send ao** above) is sent to the remote host. The initial value for the flush character is taken to be the terminal's **flush** character.

forw1

forw2 If TELNET is operating in LINEMODE, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's eol and eol2 characters.

interrupt
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **interrupt** character is typed, a TELNET IP sequence (see **send ip** above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

kill If **telnet** is in **localchars** mode (see **toggle localchars** below), **and** if **telnet** is operating in "character at a time" mode, then when this character is typed, a TELNET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

lnext If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **lnext** character. The initial value for the lnext character is taken to be the terminal's **lnext** character.

quit If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **quit** character is typed, a TELNET BRK sequence (see **send brk** above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

reprint
If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **reprint** character. The initial value for the reprint character is taken to be the terminal's **reprint** character.

rlogin This is the rlogin escape character. If set, the normal TELNET escape character is ignored unless it is preceded by this character at the beginning of a line. This character, at the beginning of a line followed by a "." closes the connection; when followed by a ^Z it suspends the **telnet** command. The initial state is to disable the rlogin escape character.

start If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's **start** character. The initial value for the start character is taken to be the terminal's **start** character.

stop If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then this character is taken to be the terminal's **stop** character. The initial value for the stop character is taken to be the terminal's **stop** character.

susp If **telnet** is in **localchars** mode, or LINEMODE is enabled, and the **suspend** character is typed, a TELNET SUSP sequence (see **send susp** above) is sent to the remote host. The initial value for the suspend character is taken to be the terminal's **suspend** character.

tracefile
This is the file to which the output, caused by **netdata** or **option** tracing being TRUE, will be written. If it is set to "**-**", then tracing information will be written to standard output (the default).

worderase
If **telnet** is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal's **worderase** character. The initial value for the worderase character is taken to be the terminal's **worderase** character.

? Displays the legal **set** (**unset**) commands.

slc *state* The **slc** command (Set Local Characters) is used to set or change the state of the special characters when the TELNET LINEMODE option has been enabled. Special characters are characters that get mapped to TELNET commands sequences (like **ip** or **quit**) or line editing characters (like **erase** and **kill**). By default, the local special characters are exported.

check Verify the current settings for the current special characters. The remote side is requested to send all the current special character settings, and if there are any discrepancies with the local side, the local side will switch to the remote value.

export Switch to the local defaults for the special characters. The local default characters are those of the local terminal at the time when **telnet** was started.

import Switch to the remote defaults for the special characters. The remote default characters are those of the remote system at the time when the TELNET connection was established.

? Prints out help information for the **slc** command.

status Show the current status of **telnet**. This includes the peer one is connected to, as well as the current mode.

toggle *arguments ...*
Toggle (between TRUE and FALSE) various flags that control how **telnet** responds to events. These flags may be set explicitly to TRUE or FALSE using the **set** and **unset** commands listed above. More than one argument may be specified. The state of these flags may be interrogated with the **display** command. Valid arguments are:

authdebug Turns on debugging information for the authentication code.

autoflush If **autoflush** and **localchars** are both TRUE, then when the **ao**, or **quit** characters are recognized (and transformed into TELNET sequences; see **set** above for details), **telnet** refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET TIMING MARK option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE if the terminal user had not done an "stty noflsh", otherwise FALSE (see stty(1)).

autodecrypt
When the TELNET ENCRYPT option is negotiated, by default the actual encryption (decryption) of the data stream does not start automatically. The autoencrypt (autodecrypt) command states that encryption of the output (input) stream should be enabled as soon as possible.

Note: Because of export controls, the TELNET ENCRYPT option is not supported outside the United States and Canada.

autologin If the remote side supports the TELNET AUTHENTICATION option TELNET attempts to use it to perform automatic authentication. If the AUTHENTICATION option is not supported, the user's login name are propagated through the TELNET ENVIRON option. This command is the same as specifying the **-a** option on the **open** command.

autosynch If **autosynch** and **localchars** are both TRUE, then when either the **intr** or **quit** characters is typed (see **set** above for descriptions of the **intr** and **quit** characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure **should** cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

binary Enable or disable the TELNET BINARY option on both input and output.

inbinary Enable or disable the TELNET BINARY option on input.

outbinary Enable or disable the TELNET BINARY option on output.

crlf If this is TRUE, then carriage returns will be sent as <CR><LF>. If this is FALSE, then carriage returns will be send as <CR><NUL>. The initial value for this toggle is FALSE.

crmod Toggle carriage return mode. When this mode is enabled, most carriage return characters received from the remote host will be mapped into a carriage return followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is not very useful unless the remote host only sends carriage return, but never line feed. The initial value for this toggle is FALSE.

debug Toggles socket level debugging (useful only to the **super user**). The initial value for this toggle is FALSE.

encdebug Turns on debugging information for the encryption code.

localchars If this is TRUE, then the **flush**, **interrupt**, **quit**, **erase**, and **kill** characters (see **set** above) are recognized locally, and transformed into (hopefully) appropriate TELNET control sequences (respectively **ao**, **ip**, **brk**, **ec**, and **el**; see **send** above). The initial value for this toggle is TRUE in "old line by line" mode, and FALSE in "character at a time" mode. When the LINEMODE option is enabled, the value of **localchars** is ignored, and assumed to always be TRUE. If LINEMODE has ever been enabled, then **quit** is sent as **abort**, and **eof and suspend** are sent as **eof and susp** (see **send** above).

netdata Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

options Toggles the display of some internal **telnet** protocol processing (having to do with TELNET options). The initial value for this toggle is FALSE.

prettydump When the **netdata** toggle is enabled, if **prettydump** is enabled the output from the **netdata** command will be formatted in a more user readable format. Spaces are put between each character in the output, and the beginning of any TELNET escape sequence is preceded by a '*' to aid in locating them.

skiprc When the skiprc toggle is TRUE, TELNET skips the reading of the `.telnetrc` file in the users home directory when connections are opened. The initial value for this toggle is FALSE.

termdata Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is FALSE.

verbose_encrypt
When the **verbose_encrypt** toggle is TRUE, **telnet** prints out a message each time encryption is enabled or disabled. The initial value for this toggle is FALSE. Note: Because of export controls, data encryption is not supported outside of the United States and Canada.

? Displays the legal **toggle** commands.

z Suspend **telnet**. This command only works when the user is using the csh(1).

! [*command*]
Execute a single command in a subshell on the local system. If *command* is omitted, then an interactive subshell is invoked.

? [*command*]
Get help. With no arguments, **telnet** prints a help summary. If a command is specified, **telnet** will print the help information for just that command.

ENVIRONMENT

telnet uses at least the HOME, SHELL, DISPLAY, and TERM environment variables. Other environment variables may be propagated to the other side via the TELNET ENVIRON option.

FILES

`~/.telnetrc` user customized telnet startup values

HISTORY

The **telnet** command appeared in 4.2BSD. IPsec support was added by WIDE/KAME project, in 1999.

NOTES

On some remote systems, echo has to be turned off manually when in "old line by line" mode.

In "old line by line" mode or LINEMODE the terminal's **eof** character is only recognized (and sent to the remote system) when it is the first character on a line.

NAME

tenletxr — forward X-connections backwards.

SYNOPSIS

tenletxr [**-l** *username*] [**-k**] *host* [*port*]

DESCRIPTION

The **tenletxr** program enables forwarding of X-connections from this machine to host *host*. If *port* is given, that port will be used instead of the default.

The supported options are:

-l Log in on the remote host as user *username*

-k Disables keep-alives.

EXAMPLE

To login from host *foo* to host *bar* (where your display is), you might do the following.

1. On foo: **tenletxr** *bar*
2. You will get a new shell where you will be able to start X clients that will show their windows on *bar*.

BUGS

It currently checks if you have permission to run it by checking if you own `/dev/console` on the remote host.

SEE ALSO

`kx`(1), `rxtelnet`(1), `rxterm`(1), `telnet`(1), `kxd`(8)

NAME

test, **[** — condition evaluation utility

SYNOPSIS

test *expression*
[*expression*]

DESCRIPTION

The **test** utility evaluates the expression and, if it evaluates to true, returns a zero (true) exit status; otherwise it returns 1 (false). If there is no expression, test also returns 1 (false).

All operators and flags are separate arguments to the **test** utility.

The following primaries are used to construct expression:

-b *file* True if *file* exists and is a block special file.

-c *file* True if *file* exists and is a character special file.

-d *file* True if *file* exists and is a directory.

-e *file* True if *file* exists (regardless of type).

-f *file* True if *file* exists and is a regular file.

-g *file* True if *file* exists and its set group ID flag is set.

-h *file* True if *file* exists and is a symbolic link.

-k *file* True if *file* exists and its sticky bit is set.

-n *string* True if the length of *string* is nonzero.

-p *file* True if *file* exists and is a named pipe (FIFO).

-r *file* True if *file* exists and is readable.

-s *file* True if *file* exists and has a size greater than zero.

-t *file_descriptor*
True if the file whose file descriptor number is *file_descriptor* is open and is associated with a terminal.

-u *file* True if *file* exists and its set user ID flag is set.

-w *file* True if *file* exists and is writable. True indicates only that the write flag is on. The file is not writable on a read-only file system even if this test indicates true.

-x *file* True if *file* exists and is executable. True indicates only that the execute flag is on. If *file* is a directory, true indicates that *file* can be searched.

-z *string* True if the length of *string* is zero.

-L *file* True if *file* exists and is a symbolic link. This operator is retained for compatibility with previous versions of this program. Do not rely on its existence; use **-h** instead.

-O *file* True if *file* exists and its owner matches the effective user id of this process.

-G *file* True if *file* exists and its group matches the effective group id of this process.

-S *file* True if *file* exists and is a socket.

file1 **-nt** *file2*
True if *file1* exists and is newer than *file2*.

file1 **-ot** *file2*
True if *file1* exists and is older than *file2*.

file1 **-ef** *file2*
True if *file1* and *file2* exist and refer to the same file.

string True if *string* is not the null string.

s1 **=** *s2* True if the strings *s1* and *s2* are identical.

s1 **!=** *s2* True if the strings *s1* and *s2* are not identical.

s1 **<** *s2* True if string *s1* comes before *s2* based on the ASCII value of their characters.

s1 **>** *s2* True if string *s1* comes after *s2* based on the ASCII value of their characters.

n1 **-eq** *n2* True if the integers *n1* and *n2* are algebraically equal.

n1 **-ne** *n2* True if the integers *n1* and *n2* are not algebraically equal.

n1 **-gt** *n2* True if the integer *n1* is algebraically greater than the integer *n2*.

n1 **-ge** *n2* True if the integer *n1* is algebraically greater than or equal to the integer *n2*.

n1 **-lt** *n2* True if the integer *n1* is algebraically less than the integer *n2*.

n1 **-le** *n2* True if the integer *n1* is algebraically less than or equal to the integer *n2*.

These primaries can be combined with the following operators:

! *expression*
True if *expression* is false.

expression1 **-a** *expression2*
True if both *expression1* and *expression2* are true.

expression1 **-o** *expression2*
True if either *expression1* or *expression2* are true.

(*expression***)**
True if expression is true.

The **-a** operator has higher precedence than the **-o** operator.

GRAMMAR AMBIGUITY

The **test** grammar is inherently ambiguous. In order to assure a degree of consistency, the cases described in IEEE Std 1003.2 (“POSIX.2”) section 4.62.4, are evaluated consistently according to the rules specified in the standards document. All other cases are subject to the ambiguity in the command semantics.

EXIT STATUS

The **test** utility exits with one of the following values:

0 expression evaluated to true.

1 expression evaluated to false or expression was missing.

>1 An error occurred.

STANDARDS

The **test** utility implements a superset of the IEEE Std 1003.2 (“POSIX.2”) specification.

NAME

texi2dvi − print Texinfo documents

SYNOPSIS

texi2dvi [*OPTION*]... *FILE*...

DESCRIPTION

Run each Texinfo or LaTeX FILE through TeX in turn until all cross-references are resolved, building all indices. The directory containing each FILE is searched for included files. The suffix of FILE is used to determine its language (LaTeX or Texinfo).

Makeinfo is used to perform Texinfo macro expansion before running TeX when needed.

Operation modes:

−b, **−−batch**
no interaction

−c, **−−clean**
remove all auxiliary files

−D, **−−debug**
turn on shell debugging (set **−x**)

−h, **−−help**
display this help and exit successfully

−o, **−−output**=*OFILE*
leave output in OFILE (implies **−−clean**); Only one input FILE may be specified in this case

−q, **−−quiet**
no output unless errors (implies **−−batch**)

−s, **−−silent**
same as **−−quiet**

−v, **−−version**
display version information and exit successfully

−V, **−−verbose**
report on what is done

TeX tuning:

-@ use @input instead of \input; for preloaded Texinfo

−e, **−E**, **−−expand**
force macro expansion using makeinfo

−I DIR search DIR for Texinfo files

−l, **−−language**=*LANG*
specify the LANG of FILE (LaTeX or Texinfo)

−p, **−−pdf**
use pdftex or pdflatex for processing

−t, **−−command**=*CMD*
insert CMD in copy of input file;

or **−−texinfo**=*CMD*
multiple values accumulate

The values of the BIBTEX, LATEX (or PDFLATEX), MAKEINDEX, MAKEINFO, TEX (or PDFTEX), TEXINDEX, and THUMBPDF environment variables are used to run those commands, if they are set. Any CMD strings are added after @setfilename for Texinfo input, in the first line for LaTeX input.

REPORTING BUGS

Email bug reports to <bug-texinfo@gnu.org>, general questions and discussion to <help-texinfo@gnu.org>. Texinfo home page: http://www.gnu.org/software/texinfo/

COPYRIGHT

SEE ALSO

The full documentation for **texi2dvi** is maintained as a Texinfo manual. If the **info** and **texi2dvi** programs are properly installed at your site, the command

info texi2dvi

should give you access to the complete manual.

NAME

texindex − sort Texinfo index files

SYNOPSIS

texindex [*OPTION*]... *FILE*...

DESCRIPTION

Generate a sorted index for each TeX output FILE. Usually FILE... is specified as ‘foo.??’ for a document ‘foo.texi’.

OPTIONS

−h, **−−help**
display this help and exit

−o, **−−output** FILE
send output to FILE

−−version
display version information and exit

REPORTING BUGS

Email bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org. Texinfo home page: http://www.gnu.org/software/texinfo/

COPYRIGHT

SEE ALSO

The full documentation for **texindex** is maintained as a Texinfo manual. If the **info** and **texindex** programs are properly installed at your site, the command

info texindex

should give you access to the complete manual.

NAME

tftp — trivial file transfer program

SYNOPSIS

tftp [**-e**] [*host*] [*port*]

DESCRIPTION

tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote *host* (and optional *port*) may be specified on the command line, in which case **tftp** uses *host* (and *port*) as the default for future transfers (see the **connect** command below).

The optional **-e** argument sets a binary transfer mode as well as setting the extended options as if **tout**, **tsize**, and **blksize 65464**, had been given.

The Multicast TFTP option is supported in open-loop (i.e., "slave-only") mode based on IETF draft-dion-tftp-multicast-option-01.txt (May 2002), which in turn was based on RFC2026.

COMMANDS

Once **tftp** is running, it issues the prompt `tftp>` and recognizes the following commands:

? *command-name ...*
: Print help information.

ascii Shorthand for "mode ascii"

binary Shorthand for "mode binary"

blksize *blk-size*
: Set the tftp blksize option to *blk-size* octets (8-bit bytes). Since the number of blocks in a tftp **get** or **put** is 65535, the default block size of 512 bytes only allows a maximum of just under 32 megabytes to be transferred. The value given for *blk-size* must be between 8 and 65464, inclusive. Note that many servers will not respect this option.

connect *host-name* [*port*]
: Set the *host* (and optionally *port*) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the **connect** command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the **connect** command; the remote host can be specified as part of the **get** or **put** commands.

get *filename*
get *remotename localname*
get *file1 file2 ... fileN*
: Get a file or set of files from the specified *sources*. *Source* can be in one of two forms: a filename on the remote host, if the host has already been specified, or a string of the form *hosts:filename* to specify both a host and filename at the same time. If the latter form is used, the last hostname specified becomes the default for future transfers.

mode *transfer-mode*
: Set the mode for transfers; *transfer-mode* may be one of *ascii* or *binary*. The default is *ascii*.

put *file*

put *localfile remotefile*
put *file1 file2 ... fileN remote-directory*
Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a filename on the remote host, if the host has already been specified, or a string of the form *hosts:filename* to specify both a host and filename at the same time. If the latter form is used, the hostname specified becomes the default for future transfers. If the remote-directory form is used, the remote host is assumed to be a UNIX machine. If you need to specify IPv6 numeric address to *hosts*, wrap them using square bracket like *[hosts]:filename* to disambiguate the colon.

quit Exit **tftp**. An end of file also exits.

rexmt *retransmission-timeout*
Set the per-packet retransmission timeout, in seconds.

status Show current status.

timeout *total-transmission-timeout*
Set the total transmission timeout, in seconds.

tout Toggle the tftp "timeout" option. If enabled, the client will pass its *retransmission-timeout* to the server. Note that many servers will not respect this option.

trace Toggle packet tracing.

tsize Toggle the tftp "tsize" option. If enabled, the client will pass and request the filesize of a file at the beginning of a file transfer. Note that many servers will not respect this option.

verbose Toggle verbose mode.

HISTORY

The **tftp** command appeared in 4.3BSD. IPv6 support was implemented by WIDE/KAME project in 1999. TFTP options were implemented by Wasabi Systems, Inc., in 2003, and first appeared in NetBSD 2.0. Multicast TFTP was implemented by Jared D. McNeill in 2006, and first appeared in NetBSD 4.0.

SECURITY CONSIDERATIONS

Because there is no user-login or validation within the TFTP protocol, the remote site will probably have some sort of file-access restrictions in place. The exact methods are specific to each site and therefore difficult to document here.

NAME

time — time command execution

SYNOPSIS

time [**-lp**] *utility* [*argument ...*]

DESCRIPTION

The **time** utility executes and times *utility*. After the *utility* finishes, **time** writes the total time elapsed, the time consumed by system overhead, and the time used to execute *utility* to the standard error stream. Times are reported in seconds.

Available options:

-l Lists resource utilization information. The contents of the *utility* process' *rusage* structure are printed, see below.

-p The output is formatted as specified by IEEE Std 1003.2-1992 ("POSIX.2").

Some shells, such as csh(1) and ksh(1), have their own and syntactically different builtin version of **time**. The utility described here is available as /usr/bin/time to users of these shells.

Resource Utilization

If the **-l** option is given, the following resource usage informations are displayed in addition to the timing information:

maximum resident set size
average shared memory size
average unshared data size
average unshared stack size
page reclaims
page faults
swaps
block input operations
block output operations
messages sent
messages received
signals received
voluntary context switches
involuntary context switches

EXIT STATUS

The **time** utility exits with one of the following values:

1-125 An error occurred in the **time** utility.

126 The *utility* was found but could not be invoked.

127 The *utility* could not be found.

Otherwise, the exit status of **time** will be that of *utility*.

FILES

<sys/resource.h>

SEE ALSO

`csh`(1), `ksh`(1), `getrusage`(2)

STANDARDS

The **`time`** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

BUGS

The granularity of seconds on microprocessors is crude and can result in times being reported for CPU usage which are too large by a second.

NAME

tip, **cu** — connect to a remote system

SYNOPSIS

tip [**-v**] **-***speed system-name*
tip [**-v**] **-***speed phone-number*
cu [*options*] *phone-number*
cu [*options*] "*dir*"
cu --help

DESCRIPTION

tip and **cu** are used to connect to another system over a serial link. In the era before modern networks, they were typically used to connect to a modem in order to dial in to a remote host. They are now frequently used for tasks such as attaching to the serial console of another machine for administrative or debugging purposes.

The following option is available for **tip**:

-v Set verbose mode.

The following options are available for **cu**:

-a *acu*
Set the ACU port.

-c *number*
Call this number.

-E *char*
Use this escape character.

-e Use even parity.

-F *flow*
Set flow control to *hard*, *soft*, or *none*.

-f Use no flow control.

-h Echo characters locally (half-duplex mode).

-l *line*
Specify the line to use. Either of the forms like `tty00` or `/dev/tty00` are permitted.

-n No escape (disable tilde).

-o Use odd parity.

-P *parity*
Set parity to *even* or *odd*.

-p *acu*
Set the ACU port.

-s *speed*
Set the speed of the connection. Defaults to 9600.

-t Connect via a hard-wired connection to a host on a dial-up line.

For **cu**, if both **-e** and **-o** are given, then no parity is used. This is the default behaviour.

If *speed* is specified it will override any baudrate specified in the system description being used.

If neither *speed* nor *system-name* are specified, *system-name* will be set to the value of the HOST environment variable.

If *speed* is specified but *system-name* is not, *system-name* will be set to a value of "tip" with *speed* appended. e.g. **tip -1200** will set *system-name* to "tip1200".

Typed characters are normally transmitted directly to the remote machine (which does the echoing as well). A tilde ('~') appearing as the first character of a line is an escape signal; the following are recognized:

~^D or **~**.
Drop the connection and exit (you may still be logged in on the remote machine).

~c [*name*]
Change directory to *name* (no argument implies change to your home directory).

~! Escape to a shell (exiting the shell will return you to tip).

~> Copy file from local to remote. **tip** prompts for the name of a local file to transmit.

~< Copy file from remote to local. **tip** prompts first for the name of the file to be sent, then for a command to be executed on the remote machine.

~p *from* [*to*]
Send a file to a remote UNIX host. The put command causes the remote UNIX system to run the command string "cat > 'to'", while **tip** sends it the "from" file. If the "to" file isn't specified the "from" file name is used. This command is actually a UNIX specific version of the "~>" command.

~t *from* [*to*]
Take a file from a remote UNIX host. As in the put command the "to" file defaults to the "from" file name if it isn't specified. The remote host executes the command string "cat 'from';echo ^A" to send the file to **tip**.

~| Pipe the output from a remote command to a local UNIX process. The command string sent to the local UNIX system is processed by the shell.

~$ Pipe the output from a local UNIX process to the remote host. The command string sent to the local UNIX system is processed by the shell.

~C Fork a child process on the local system to perform special protocols such as XMODEM. The child program will be run with the following arrangement of file descriptors:

0 <-> remote tty in
1 <-> remote tty out
2 <-> local tty out

~+ Synonym for ~C, provided for compatibility with other versions of **cu**.

~# Send a BREAK to the remote system. For systems which don't support the necessary *ioctl* call the break is simulated by a sequence of line speed changes and DEL characters.

~s Set a variable (see the discussion below).

~^Z Stop **tip** (only available with job control).

~^Y Stop only the "local side" of **tip** (only available with job control); the "remote side" of **tip**, the side that displays output from the remote host, is left running.

~? Get a summary of the tilde escapes

tip uses the file /etc/remote to find how to reach a particular system and to find out how it should operate while talking to the system; refer to remote(5) for a full description. Each system has a default baud rate with which to establish a connection. If this value is not suitable, the baud rate to be used may be specified on the command line, e.g. tip -300 mds.

When **tip** establishes a connection it sends out a connection message to the remote system; the default value, if any, is defined in /etc/remote (see remote(5)).

When **tip** prompts for an argument (e.g. during setup of a file transfer) the line typed may be edited with the standard erase and kill characters. A null line in response to a prompt, or an interrupt, will abort the dialogue and return you to the remote machine.

tip guards against multiple users connecting to a remote system by opening modems and terminal lines with exclusive access, and by honoring the locking protocol used by uucico(8).

During file transfers **tip** provides a running count of the number of lines transferred. When using the ~> and ~< commands, the ''eofread'' and ''eofwrite'' variables are used to recognize end-of-file when reading, and specify end-of-file when writing (see below). File transfers normally depend on tandem mode for flow control. If the remote system does not support tandem mode, ''echocheck'' may be set to indicate **tip** should synchronize with the remote system on the echo of each transmitted character.

When **tip** must dial a phone number to connect to a system it will print various messages indicating its actions. **tip** supports the DEC DN-11 and Racal-Vadic 831 auto-call-units; the DEC DF02 and DF03, Ventel 212+, Racal-Vadic 3451, and Bizcomp 1031 and 1032 integral call unit/modems.

VARIABLES

tip maintains a set of *variables* which control its operation. Some of these variables are read-only to normal users (root is allowed to change anything of interest). Variables may be displayed and set through the ''s'' escape. The syntax for variables is patterned after vi(1) and Mail(1). Supplying ''all'' as an argument to the set command displays all variables readable by the user. Alternatively, the user may request display of a particular variable by attaching a '?' to the end. For example ''escape?'' displays the current escape character.

Variables are numeric, string, character, or boolean values. Boolean variables are set merely by specifying their name; they may be reset by prepending a '!' to the name. Other variable types are set by concatenating an '=' and the value. The entire assignment must not have any blanks in it. A single set command may be used to interrogate as well as set a number of variables. Variables may be initialized at run time by placing set commands (without the ''~s'' prefix in a file .tiprc in one's home directory). The **-v** option causes **tip** to display the sets as they are made. Certain common variables have abbreviations. The following is a list of common variables, their abbreviations, and their default values.

beautify (bool) Discard unprintable characters when a session is being scripted; abbreviated *be*.

baudrate (num) The baud rate at which the connection was established; abbreviated *ba*.

dialtimeout
(num) When dialing a phone number, the time (in seconds) to wait for a connection to be established; abbreviated *dial*.

echocheck (bool) Synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted; default is *off*.

eofread (str) The set of characters which signify an end-of-transmission during a ~< file transfer command; abbreviated *eofr*.

eofwrite (str) The string sent to indicate end-of-transmission during a ~> file transfer command; abbreviated *eofw*.

eol (str) The set of characters which indicate an end-of-line. **tip** will recognize escape characters only after an end-of-line.

escape (char) The command prefix (escape) character; abbreviated *es*; default value is ‘~’.

exceptions (str) The set of characters which should not be discarded due to the beautification switch; abbreviated *ex*; default value is ‘‘\t\n\f\b’’.

force (char) The character used to force literal data transmission; abbreviated *fo*; default value is ‘^P’.

framesize (num) The amount of data (in bytes) to buffer between file system writes when receiving files; abbreviated *fr*.

host (str) The name of the host to which you are connected; abbreviated *ho*.

prompt (char) The character which indicates an end-of-line on the remote host; abbreviated *pr*; default value is ‘\n’. This value is used to synchronize during data transfers. The count of lines transferred during a file transfer command is based on receipt of this character.

raise (bool) Upper case mapping mode; abbreviated *ra*; default value is *off*. When this mode is enabled, all lower case letters will be mapped to upper case by **tip** for transmission to the remote machine.

raisechar (char) The input character used to toggle upper case mapping mode; abbreviated *rc*; default value is ‘^A’.

record (str) The name of the file in which a session script is recorded; abbreviated *rec*; default value is ‘‘tip.record’’.

script (bool) Session scripting mode; abbreviated *sc*; default is *off*. When *script* is true, **tip** will record everything transmitted by the remote machine in the script record file specified in *record*. If the *beautify* switch is on, only printable ASCII characters will be included in the script file (those characters between 040 and 0177). The variable *exceptions* is used to indicate characters which are an exception to the normal beautification rules.

tabexpand (bool) Expand tabs to spaces during file transfers; abbreviated *tab*; default value is *false*. Each tab is expanded to 8 spaces.

tandem (bool) Use XON/XOFF flow control to throttle data from the remote host; abbreviated *ta*. The default value is *true* unless the *nt* capability has been specified in /etc/remote, in which case the default value is *false*.

verbose (bool) Verbose mode; abbreviated *verb*; default is *true*. When verbose mode is enabled, **tip** prints messages while dialing, shows the current number of lines transferred during a file transfer operations, and more.

ENVIRONMENT

tip uses the following environment variables:

SHELL (str) The name of the shell to use for the ~! command; default value is ‘‘/bin/sh’’, or taken from the environment.

HOME (str) The home directory to use for the ~c command; default value is taken from the environment.

HOST Check for a default host if none specified.

The variables `${REMOTE}` and `${PHONES}` are also exported.

FILES

`/etc/remote`	Global system descriptions.
`/etc/phones`	Global phone number data base.
${REMOTE}	Private system descriptions.
${PHONES}	Private phone numbers.
~/.tiprc	Initialization file.
`tip.record`	Record file.

DIAGNOSTICS

Diagnostics are, hopefully, self explanatory.

SEE ALSO

`phones`(5), `remote`(5)

HISTORY

The **`tip`** command appeared in 4.2BSD.

BUGS

The full set of variables is undocumented and should, probably, be pared down.

NAME

tn3270 — full-screen remote login to IBM VM/CMS

SYNOPSIS

tn3270 [**-d**] [**-n** *filename*] [**-t** *commandname*] [*sysname* [port]]

DESCRIPTION

tn3270 permits a full-screen, full-duplex connection from a UNIX machine to an IBM (or compatible) machine. **tn3270** gives the appearance of being logged in to the remote machine from an IBM 3270 terminal. Of course, you must have an account on the machine to which you connect in order to log in. **tn3270** looks to the user in many respects like the Yale ASCII Terminal Communication System II. **tn3270** is actually a modification of the ARPANET TELNET user interface (see telnet(1)) which will, in certain circumstances, interpret and generate raw 3270 control streams.

The flags to **tn3270** are as follows:

-d Turn on socket-level tracing (for super-user only)

-n *filename*
Specify a file to receive network trace data output (from commands "toggle netdata" and "toggle options", see telnet(1)); the default is for output to be directed to the standard error file.

-t *commandname*
Specify a UNIX command to process IBM 4994 style transparent mode data received from the remote IBM machine.

sysname
The name of the remote system. If the remote name is NOT specified, the user will be prompted for a command (see below).

port The port to connect to on the remote system. Normally, **tn3270** attempts to connect to the standard TELNET port (port 23) on the remote machine.

When **tn3270** first connects to the remote system, it will negotiate to go into 3270 mode. Part of this negotiation involves telling the remote system what model 3270 it is emulating. In all cases, **tn3270** emulates a 3278 terminal. To decide which specific model, **tn3270** looks at the number of lines and columns on the actual terminal (as defined in the TERM environment variable; see termcap(5)). The terminal (or window in which **tn3270** is running, on multiple window systems) must have at least 80 columns and 24 lines, or **tn3270** will not go into emulation mode. If the terminal does have at least 80 columns and at least 24 lines, the following table describes the emulation:

minimum_size (rows*columns)	emulated terminal
27*132	3278 model 5
43*80	3278 model 4
32*80	3278 model 3
24*80	3278 model 2.

Emulation of the 3270 terminal is done in the UNIX process. This emulation involves mapping 3270-style commands from the host into appropriate sequences to control the user's terminal screen. **tn3270** uses curses(3) and the /usr/share/misc/termcap file to do this. The emulation also involves simulating the special 3270 keyboard keys (program function keys, etc.) by mapping sequences of keystrokes from the ASCII keyboard into appropriate 3270 control strings. This mapping is terminal dependent and is specified in a description file, /usr/share/misc/map3270, (see map3270(5)) or in an environment variable MAP3270 (and, if necessary, MAP3270A, MAP3270B, and so on - see mset(1)). Any special function

keys on the ASCII keyboard are used whenever possible. If an entry for the user's terminal is not found, **tn3270** looks for an entry for the terminal type *unknown*. If this is not found, **tn3270** uses a default keyboard mapping (see map3270(5)).

The first character of each special keyboard mapping sequence is either an ASCII escape (ESC), a control character, or an ASCII delete (DEL). If the user types an unrecognized function key sequence, **tn3270** sends an ASCII bell (BEL), or a visual bell if defined in the user's termcap entry, to the user's terminal and nothing is sent to the IBM host.

If **tn3270** is invoked without specifying a remote host system name, it enters local command mode, indicated by the prompt "tn3270> ". In this mode, **tn3270** accepts and executes all the commands of telnet(1), plus one additional command:

transcom Specify UNIX command for IBM 4994 style transparent mode processing.

tn3270 command mode may also be entered, after connecting to a host, by typing a special escape sequence. If **tn3270** has succeeded in negotiating 3270 mode with the remote host, the escape sequence will be as defined by the map3270 (see map3270(5)) entry for the user's terminal type (typically control-C); otherwise the escape sequence will initially be set to the single character '^]' (control right square bracket).

While in command mode, any host login session is still alive but temporarily suspended. The host login session may be resumed by entering an empty line (press the RETURN key) in response to the command prompt. A session may be terminated by logging off the foreign host, or by typing ''quit'' or ''close'' while in local command mode.

FILES

/usr/share/misc/termcap
/usr/share/misc/map3270

NOTES

The IBM 4994 style transparent mode command is invoked when **tn3270** receives IBM 4994 style transparent output from the remote host. Output and input pipes are created for communication between the two processes. The pipes are closed when a 3270 clear command is received from the remote hosts, signaling the end of transparent mode output. Transparent mode is necessary for sending ASCII control characters over the 3270 terminal connection; ASCII graphics terminal support is accomplished this way. Developers of **transcom** commands should note that the **transcom** stdin pipe end will be in CBREAK mode, with ECHO and CRMOD turned off.

ENVIRONMENT

tn3270 checks the following environment variables: TERM, MAP3270, MAP3270[A...]. Information on these can be found in mset(1). **tn3270** also checks SHELL, KEYBD and API3270.

SEE ALSO

mset(1), telnet(1), curses(3), termcap(3), map3270(5), termcap(5)

"Yale ASCII Terminal Communication", *System II Program Description/Operator's Manual*, IBM SB30-1911.

HISTORY

The **tn3270** command appeared in 4.3BSD.

BUGS

tn3270 is slow and uses system resources prodigiously.

Not all 3270 functions are supported, nor all Yale enhancements.

Error conditions (attempting to enter data in a protected field, for example) should cause a message to be sent to the user’s terminal instead of just ringing a bell.

NAME

top − display and update information about the top CPU processes

SYNOPSIS

top [**−bIinqStuv**] [**−d***count*] [**−o** *field*] [**−s***time*] [**−U***username*] [*number*]

DESCRIPTION

Top displays the top processes on the system and periodically updates this information. If standard output is an intelligent terminal (see below) then as many processes as will fit on the terminal screen are displayed by default. Otherwise, a good number of them are shown (around 20). Raw CPU percentage is used to rank the processes. If *number* is given, then the top *number* processes will be displayed instead of the default.

Top makes a distinction between terminals that support advanced capabilities and those that do not. This distinction affects the choice of defaults for certain options. In the remainder of this document, an ‘‘intelligent’’ terminal is one that supports cursor addressing, clear screen, and clear to end of line. Conversely, a ‘‘dumb’’ terminal is one that does not support such features. If the output of *top* is redirected to a file, it acts as if it were being run on a dumb terminal.

OPTIONS

−S Toggle displaying of system processes. Normally, system processes such as the pager and the swapper are shown.

−b Use ‘‘batch’’ mode. In this mode, all input from the terminal is ignored. Interrupt characters (such as ^C and ^\) still have an effect. This is the default on a dumb terminal, or when the output is not a terminal.

−i Use ‘‘interactive’’ mode. In this mode, any input is immediately read for processing. See the section on ‘‘Interactive Mode’’ for an explanation of which keys perform what functions. After the command is processed, the screen will immediately be updated, even if the command was not understood. This mode is the default when standard output is an intelligent terminal.

−I Do not display idle processes. By default, top displays both active and idle processes.

−n Use ‘‘non-interactive’’ mode. This is identical to ‘‘batch’’ mode.

−q Renice *top* to -20 so that it will run faster. This can be used when the system is being very sluggish to improve the possibility of discovering the problem. This option can only be used by root.

−t Display threads (lightweight processes). The default is to display processes.

−u Do not take the time to map UID numbers to usernames. Normally, *top* will read as much of the file ‘‘/etc/passwd’’ as is necessary to map all the user id numbers it encounters into login names. This option disables all that, while possibly decreasing execution time. The UID numbers are displayed instead of the names.

−v Write version number information to stderr then exit immediately. No other processing takes place when this option is used. To see current revision information while top is running, use the help command ‘‘?’’.

−d*count*
Show only *count* displays, then exit. A display is considered to be one update of the screen. This option allows the user to select the number of displays he wants to see before *top* automatically exits. For intelligent terminals, no upper limit is set. The default is 1 for dumb terminals.

−s*time* Set the delay between screen updates to *time* seconds. The default delay between updates is 2 seconds.

−o *field* Sort the process display area on the specified field. The field name is the name of the column as seen in the output, but in lower case. Likely values are ‘‘cpu’’, ‘‘size’’, ‘‘res’’, and ‘‘time’’, but may vary on different operating systems. Note that not all operating systems support this option.

−U*username*

Show only those processes owned by *username*. This option currently only accepts usernames and will not understand UID numbers.

Both *count* and *number* fields can be specified as ‘‘infinite’’, indicating that they can stretch as far as possible. This is accomplished by using any proper prefix of the keywords ‘‘infinity’’, ‘‘maximum’’, or ‘‘all’’. The default for *count* on an intelligent terminal is, in fact, **infinity**.

The environment variable **TOP** is examined for options before the command line is scanned. This enables a user to set his or her own defaults. The number of processes to display can also be specified in the environment variable **TOP**. The options **−I**, **−S**, and **−u** are actually toggles. A second specification of any of these options will negate the first. Thus a user who has the environment variable **TOP** set to ‘‘−I’’ may use the command ‘‘top −I’’ to see idle processes.

INTERACTIVE MODE

When *top* is running in ‘‘interactive mode’’, it reads commands from the terminal and acts upon them accordingly. In this mode, the terminal is put in ‘‘CBREAK’’, so that a character will be processed as soon as it is typed. Almost always, a key will be pressed when *top* is between displays; that is, while it is waiting for *time* seconds to elapse. If this is the case, the command will be processed and the display will be updated immediately thereafter (reflecting any changes that the command may have specified). This happens even if the command was incorrect. If a key is pressed while *top* is in the middle of updating the display, it will finish the update and then process the command. Some commands require additional information, and the user will be prompted accordingly. While typing this information in, the user’s erase and kill keys (as set up by the command *stty*) are recognized, and a newline terminates the input.

These commands are currently recognized (ˆL refers to control-L):

ˆL Redraw the screen.

h or **?** Display a summary of the commands (help screen). Version information is included in this display.

q Quit *top.*

d Change the number of displays to show (prompt for new number). Remember that the next display counts as one, so typing **d1** will make *top* show one final display and then immediately exit.

n or # Change the number of processes to display (prompt for new number).

s Change the number of seconds to delay between displays (prompt for new number).

S Toggle between showing and not showing system processes.

t Toggle between showing threads and showing processes.

k Send a signal (‘‘kill’’ by default) to a list of processes. This acts similarly to the command *kill*(1)).

r Change the priority (the ‘‘nice’’) of a list of processes. This acts similarly to the command *renice*(8)).

u Display only processes owned by a specific username (prompt for username). If the username specified is simply ‘‘+’’, then processes belonging to all users will be displayed.

o Change the order in which the display is sorted. This command is not available on all systems. The sort key names vary from system to system but usually include: ‘‘cpu’’, ‘‘res’’, ‘‘size’’, ‘‘time’’. The default is cpu.

p Select a single process to display. Useful when showing threads.

e Display a list of system errors (if any) generated by the last **k**ill or **r**enice command.

i (or **I)** Toggle the display of idle processes.

THE DISPLAY

The actual display varies depending on the specific variant of Unix that the machine is running. This description may not exactly match what is seen by top running on this particular machine. Differences are

listed at the end of this manual entry.

The top few lines of the display show general information about the state of the system, including the last process id assigned to a process (on most systems), the three load averages, the uptime, the current time, the number of existing processes, the number of processes in each state (sleeping, running, starting, zombies, and stopped), and a percentage of time spent in each of the processor states (user, nice, system, and idle). It also includes information about physical and virtual memory allocation.

The remainder of the screen displays information about individual processes. This display is similar in spirit to *ps*(1) but it is not exactly the same. PID is the process id, USERNAME is the name of the process's owner (if **−u** is specified, a UID column will be substituted for USERNAME), PRI is the current priority of the process, NICE is the nice amount (in the range −20 to 20), SIZE is the total size of the process (text, data, and stack), RES is the current amount of resident memory (both SIZE and RES are given in kilobytes), STATE is the current state (one of ''START'', ''RUN'', ''STOP'', ''ZOMB'', ''DEAD'', or ''CPU'') or wait channel if the state is SLEEP, TIME is the number of system and user CPU seconds that the process has used, WCPU, when displayed, is the weighted CPU percentage (this is the same value that *ps*(1) displays as CPU), CPU is the raw percentage and is the field that is sorted to determine the order of the processes, and COMMAND is the name of the command that the process is currently running (if the process is swapped out, this column is marked ''<swapped>''). On multi-processor systems, the STATE field may be followed by a slash and CPU number.

NOTES

The ''ABANDONED'' state (known in the kernel as ''SWAIT'') was abandoned, thus the name. A process should never end up in this state.

AUTHOR

William LeFebvre, EECS Department, Northwestern University

ENVIRONMENT

TOP user-configurable defaults for options.

FILES

/dev/kmem	kernel memory
/dev/mem	physical memory
/etc/passwd	used to map UID numbers to user names
/netbsd	system image

BUGS

Don't shoot me, but the default for **−I** has changed once again. So many people were confused by the fact that *top* wasn't showing them all the processes that I have decided to make the default behavior show idle processes, just like it did in version 2. But to appease folks who can't stand that behavior, I have added the ability to set ''default'' options in the environment variable **TOP** (see the OPTIONS section). Those who want the behavior that version 3.0 had need only set the environment variable **TOP** to ''−I''.

The command name for swapped processes should be tracked down, but this would make the program run slower.

As with *ps*(1), things can change while *top* is collecting information for an update. The picture it gives is only a close approximation to reality.

SEE ALSO

kill(1), ps(1), stty(1), systat(1), mem(4), renice(8)

NAME

touch — change file access and modification times

SYNOPSIS

touch [**-acfhm**] [**-r** *file*] [**-t** *[[CC]YY]MMDDhhmm[.SS]*] *file ...*

DESCRIPTION

The **touch** utility sets the modification and access times of files to the current time of day. If the file doesn't exist, it is created with default permissions.

The following options are available:

-a Change the access time of the file. The modification time of the file is not changed unless the **-m** flag is also specified.

-c Do not create the file if it does not exist. The **touch** utility does not treat this as an error. No error messages are displayed and the exit value is not affected.

-f Attempt to force the update, even if the file permissions do not currently permit it.

-h If *file* is a symbolic link, access and/or modification time of the link is changed. This option implies **-c**.

-m Change the modification time of the file. The access time of the file is not changed unless the **-a** flag is also specified.

-r Use the access and modifications times from the specified file instead of the current time of day.

-t Change the access and modification times to the specified time. The argument should be in the form "[[CC]YY]MMDDhhmm[.SS]" where each pair of letters represents the following:

CC The first two digits of the year (the century).

YY The second two digits of the year. If "YY" is specified, but "CC" is not, a value for "YY" between 69 and 99 results in a "CC" value of 19. Otherwise, a "CC" value of 20 is used.

MM The month of the year, from 1 to 12.

DD The day of the month, from 1 to 31.

hh The hour of the day, from 0 to 23.

mm The minute of the hour, from 0 to 59.

SS The second of the minute, from 0 to 61.

If the "CC" and "YY" letter pairs are not specified, the values default to the current year. If the "SS" letter pair is not specified, the value defaults to 0.

The **touch** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

utimes(2)

COMPATIBILITY

The obsolescent form of **touch**, where a time format is specified as the first argument, is supported. When no **-r** or **-t** option is specified, there are at least two arguments, and the first argument is a string of digits either eight or ten characters in length, the first argument is interpreted as a time specification of the form "MMDDhhmm[YY]".

The "MM", "DD", "hh" and "mm" letter pairs are treated as their counterparts specified to the **-t** option. If the "YY" letter pair is in the range 69 to 99, the year is set to 1969 to 1999, otherwise, the year is set in the 21st century.

STANDARDS

The **touch** utility is expected to be a superset of the IEEE Std 1003.2 (“POSIX.2”) specification.

HISTORY

A **touch** utility appeared in Version 7 AT&T UNIX.

BUGS

A symbolic link can’t be a reference file of access and/or modification time.

NAME

tput, **clear** — terminal capability interface

SYNOPSIS

tput [**-T** *term*] *attribute* [*attribute-args*] . . .

DESCRIPTION

tput makes terminal-dependent information available to users or shell applications. The options are as follows:

-T The terminal name as specified in the termcap(5) database, for example, “vt100” or “xterm”. If not specified, **tput** retrieves the “TERM” variable from the environment.

tput outputs a string if the *attribute* is of type string; a number if it is of type integer. Otherwise, **tput** exits 0 if the terminal has the capability and 1 if it does not, without further action.

If the *attribute* is of type string, and takes arguments (e.g. cursor movement, the termcap “cm” sequence) the arguments are taken from the command line immediately following the attribute.

The following special attributes are available:

clear	Clear the screen (the termcap(5) “cl” sequence).
init	Initialize the terminal (the termcap(5) “is” sequence).
longname	Print the descriptive name of the user’s terminal type.
reset	Reset the terminal (the termcap(5) “rs” sequence).

EXIT STATUS

The exit status of **tput** is based on the last attribute specified. If the attribute is of type string or of type integer, **tput** exits 0 if the attribute is defined for this terminal type and 1 if it is not. If the attribute is of type boolean, **tput** exits 0 if the terminal has this attribute, and 1 if it does not. **tput** exits 2 if any error occurred.

EXAMPLES

`tput cl cm 5 10` clear the screen and goto line 5 column 10

`tput cm 6 11 DC 6` goto line 6 column 11 and delete 6 characters

SEE ALSO

termcap(3), termcap(5)

HISTORY

The **tput** command appears in 4.4BSD.

BUGS

tput can’t really distinguish between different types of attributes.

Not all termcap entries contain the reset sequence, so using the init sequence may be more useful.

NAME

tr — translate characters

SYNOPSIS

tr [**-cs**] *string1 string2*
tr [**-c**] **-d** *string1*
tr [**-c**] **-s** *string1*
tr [**-c**] **-ds** *string1 string2*

DESCRIPTION

The **tr** utility copies the standard input to the standard output with substitution or deletion of selected characters.

The following options are available:

-c Complements the set of characters in *string1*, that is **-c** *ab* includes every character except for 'a' and 'b'.

-d The **-d** option causes characters to be deleted from the input.

-s The **-s** option squeezes multiple occurrences of the characters listed in the last operand (either *string1* or *string2*) in the input into a single instance of the character. This occurs after all deletion and translation is completed.

In the first synopsis form, the characters in *string1* are translated into the characters in *string2* where the first character in *string1* is translated into the first character in *string2* and so on. If *string1* is longer than *string2*, the last character found in *string2* is duplicated until *string1* is exhausted.

In the second synopsis form, the characters in *string1* are deleted from the input.

In the third synopsis form, the characters in *string1* are compressed as described for the **-s** option.

In the fourth synopsis form, the characters in *string1* are deleted from the input, and the characters in *string2* are compressed as described for the **-s** option.

The following conventions can be used in *string1* and *string2* to specify sets of characters:

character Any character not described by one of the following conventions represents itself.

\octal A backslash followed by 1, 2 or 3 octal digits represents a character with that encoded value. To follow an octal sequence with a digit as a character, left zero-pad the octal sequence to the full 3 octal digits.

\character A backslash followed by certain special characters maps to special values.

\a	<alert character>
\b	<backspace>
\f	<form-feed>
\n	<newline>
\r	<carriage return>
\t	<tab>
\v	<vertical tab>

A backslash followed by any other character maps to that character.

c-c Represents the range of characters between the range endpoints, inclusively.

[:class:] Represents all characters belonging to the defined character class. Class names are:

alnum	<alphanumeric characters>
alpha	<alphabetic characters>
blank	<blank characters>
cntrl	<control characters>
digit	<numeric characters>
graph	<graphic characters>
lower	<lower-case alphabetic characters>
print	<printable characters>
punct	<punctuation characters>
space	<space characters>
upper	<upper-case characters>
xdigit	<hexadecimal characters>

With the exception of the “upper” and “lower” classes, characters in the classes are in unspecified order. In the “upper” and “lower” classes, characters are entered in ascending order.

For specific information as to which ASCII characters are included in these classes, see `ctype`(3) and related manual pages.

[=equiv=] Represents all characters or collating (sorting) elements belonging to the same equivalence class as *`equiv`*. If there is a secondary ordering within the equivalence class, the characters are ordered in ascending sequence. Otherwise, they are ordered after their encoded values. An example of an equivalence class might be “c” and “ch” in Spanish; English has no equivalence classes.

[#*n] Represents *`n`* repeated occurrences of the character represented by *#*. This expression is only valid when it occurs in *`string2`*. If *`n`* is omitted or is zero, it is interpreted as large enough to extend *`string2`* sequence to the length of *`string1`*. If *`n`* has a leading zero, it is interpreted as an octal value, otherwise, it’s interpreted as a decimal value.

EXIT STATUS

`tr` exits 0 on success, and >0 if an error occurs.

EXAMPLES

The following examples are shown as given to the shell:

Create a list of the words in *`file1`*, one per line, where a word is taken to be a maximal string of letters:

```
tr -cs "[:alpha:]" "\n" < file1
```

Translate the contents of *`file1`* to upper-case:

```
tr "[:lower:]" "[:upper:]" < file1
```

Strip out non-printable characters from *`file1`*:

```
tr -cd "[:print:]" < file1
```

COMPATIBILITY

AT&T System V UNIX has historically implemented character ranges using the syntax “[c-c]” instead of the “c-c” used by historic BSD implementations and standardized by POSIX. AT&T System V UNIX shell scripts should work under this implementation as long as the range is intended to map in another range, i.e. the com-

mand

tr [a-z] [A-Z]

will work as it will map the '[' character in *string1* to the '[' character in *string2*. However, if the shell script is deleting or squeezing characters as in the command

tr -d [a-z]

the characters '[' and ']' will be included in the deletion or compression list which would not have happened under an historic AT&T System V UNIX implementation. Additionally, any scripts that depended on the sequence "a-z" to represent the three characters 'a', '-', and 'z' will have to be rewritten as "a\-z".

The **tr** utility has historically not permitted the manipulation of NUL bytes in its input and, additionally, stripped NUL's from its input stream. This implementation has removed this behavior as a bug.

The **tr** utility has historically been extremely forgiving of syntax errors, for example, the **-c** and **-s** options were ignored unless two strings were specified. This implementation will not permit illegal syntax.

STANDARDS

The **tr** utility is expected to be IEEE Std 1003.2 ("POSIX.2") compatible. It should be noted that the feature wherein the last character of *string2* is duplicated if *string2* has less characters than *string1* is permitted by POSIX but is not required. Shell scripts attempting to be portable to other POSIX systems should use the "[#*]" convention instead of relying on this behavior.

BUGS

tr was originally designed to work with US-ASCII. Its use with character sets that do not share all the properties of US-ASCII, e.g., a symmetric set of upper and lower case characters that can be algorithmically converted one to the other, may yield unpredictable results.

tr should be internationalized.

NAME

true — return true value

SYNOPSIS

true

DESCRIPTION

The **true** utility always returns with exit code zero.

SEE ALSO

csh(1), false(1), sh(1)

STANDARDS

The **true** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

NAME

tset, **reset** — terminal initialization

SYNOPSIS

tset [**-EIQrSs**] [**-**] [**-e** *ch*] [**-i** *ch*] [**-k** *ch*] [**-m** *mapping*] [*terminal*]
reset [**-EIQrSs**] [**-**] [**-e** *ch*] [**-i** *ch*] [**-k** *ch*] [**-m** *mapping*] [*terminal*]

DESCRIPTION

tset initializes terminals. **tset** first determines the type of terminal that you are using. This determination is done as follows, using the first terminal type found.

- The *terminal* argument specified on the command line.
- The value of the TERM environmental variable.
- The terminal type associated with the standard error output device in the /etc/ttys file.
- The default terminal type, “unknown”.

If the terminal type was not specified on the command-line, the **-m** option mappings are then applied (see below for more information). Then, if the terminal type begins with a question mark (“?”), the user is prompted for confirmation of the terminal type. An empty response confirms the type, or, another type can be entered to specify a new type. Once the terminal type has been determined, the termcap entry for the terminal is retrieved. If no termcap entry is found for the type, the user is prompted for another terminal type.

Once the termcap entry is retrieved, the window size, backspace, interrupt and line kill characters (among many other things) are set and the terminal and tab initialization strings are sent to the standard error output. Finally, if the erase, interrupt and line kill characters have changed, or are not set to their default values, their values are displayed to the standard error output.

When invoked as **reset**, **tset** sets cooked and echo modes, turns off cbreak and raw modes, turns on newline translation and resets any unset special characters to their default values before doing the terminal initialization described above. This is useful after a program dies leaving a terminal in a abnormal state. Note, you may have to type “<LF>reset<LF>” (the line-feed character is normally control-J) to get the terminal to work, as carriage-return may no longer work in the abnormal state. Also, the terminal will often not echo the command.

The options are as follows:

- The terminal type is displayed to the standard output, and the terminal is not initialized in any way.

-E Emit the extended termcap entry. By default the termcap entry is truncated to 1024 bytes, this flag specifies the untruncated termcap entry is to be output. Using this flag may cause problems with some shells.

-e Set the erase character to *ch*.

-I Do not send the terminal or tab initialization strings to the terminal.

-i Set the interrupt character to *ch*.

-k Set the line kill character to *ch*.

-m Specify a mapping from a port type to a terminal. See below for more information.

-Q Don’t display any values for the erase, interrupt and line kill characters.

-r Print the terminal type to the standard error output.

-S Print the terminal type and the termcap entry to the standard output. See the section below on setting the environment for details.

-s Print the sequence of shell commands to initialize the environment variables TERM and TERMCAP to the standard output. See the section below on setting the environment for details.

The arguments for the **-e**, **-i** and **-k** options may either be entered as actual characters or by using the "hat" notation, i.e. control-h may be specified as "^H" or "^h".

SETTING THE ENVIRONMENT

It is often desirable to enter the terminal type and information about the terminal's capabilities into the shell's environment. This is done using the **-S** and **-s** options.

When the **-S** option is specified, the terminal type and the termcap entry are written to the standard output, separated by a space and without a terminating newline. This can be assigned to an array by csh(1) and ksh(1) users and then used like any other shell array.

When the **-s** option is specified, the commands to enter the information into the shell's environment are written to the standard output. If the SHELL environmental variable ends in "csh", the commands are for the csh(1), otherwise, they are for sh(1). Note, the csh(1) commands **set** and **unset** the shell variable "noglob", leaving it unset. The following line in the .login or .profile files will initialize the environment correctly:

```
eval `tset -s options ... `
```

To demonstrate a simple use of the **-S** option, the following lines in the .login file have an equivalent effect:

```
set noglob
set term=(‘tset -S options ...‘)
setenv TERM $term[1]
setenv TERMCAP "$term[2]"
unset term
unset noglob
```

TERMINAL TYPE MAPPING

When the terminal is not hardwired into the system (or the current system information is incorrect) the terminal type derived from the /etc/ttys file or the TERM environmental variable is often something generic like "network", "dialup", or "unknown". When **tset** is used in a startup script (.profile for sh(1) users or .login for csh(1) users) it is often desirable to provide information about the type of terminal used on such ports. The purpose of the **-m** option is to "map" from some set of conditions to a terminal type, that is, to tell **tset** ''If I'm on this port at a particular speed, guess that I'm on that kind of terminal''.

The argument to the **-m** option consists of an optional port type, an optional operator, an optional baud rate specification, an optional colon (''':'') character and a terminal type. The port type is a string (delimited by either the operator or the colon character). The operator may be any combination of: "**>**", "**<**", "**@**", and "**!**"; "**>**" means greater than, "**<**" means less than, "**@**" means equal to and "**!**" inverts the sense of the test. The baud rate is specified as a number and is compared with the speed of the standard error output (which should be the control terminal). The terminal type is a string.

If the terminal type is not specified on the command line, the **-m** mappings are applied to the terminal type. If the port type and baud rate match the mapping, the terminal type specified in the mapping replaces the current type. If more than one mapping is specified, the first applicable mapping is used.

For example, consider the following mapping: "dialup>9600:vt100". The port type is "dialup", the operator is ">", the baud rate specification is "9600", and the terminal type is "vt100". The result of this mapping is to specify that if the terminal type is "dialup", and the baud rate is greater than 9600 baud, a terminal type of "vt100" will be used.

If no port type is specified, the terminal type will match any port type, for example, "`-m dialup:vt100 -m :?xterm`" will cause any dialup port, regardless of baud rate, to match the terminal type "`vt100`", and any non-dialup port type to match the terminal type "`?xterm`". Note, because of the leading question mark, the user will be queried on a default port as to whether they are actually using an *xterm* terminal.

No whitespace characters are permitted in the **-m** option argument. Also, to avoid problems with metacharacters, it is suggested that the entire **-m** option argument be placed within single quote characters, and that csh(1) users insert a backslash character ("\") before any exclamation marks ("!").

ENVIRONMENT

The **tset** command uses the SHELL and TERM environment variables.

FILES

/etc/ttys	system port name to terminal type mapping database
/usr/share/misc/termcap	terminal capability database

SEE ALSO

csh(1), sh(1), stty(1), tty(4), termcap(5), ttys(5), environ(7)

HISTORY

The **tset** command appeared in 3.0 BSD.

COMPATIBILITY

The **-A**, **-E**, **-h**, **-u** and **-v** options have been deleted from the **tset** utility. None of them were documented in 4.3 BSD and all are of limited utility at best. The **-a**, **-d** and **-p** options are similarly not documented or useful, but were retained as they appear to be in widespread use. It is strongly recommended that any usage of these three options be changed to use the **-m** option instead. The **-n** option remains, but has no effect. It is still permissible to specify the **-e**, **-i** and **-k** options without arguments, although it is strongly recommended that such usage be fixed to explicitly specify the character.

Executing **tset** as **reset** no longer implies the **-Q** option. Also, the interaction between the **-** option and the *terminal* argument in some historic implementations of **tset** has been removed.

Finally, the **tset** implementation has been completely redone (as part of the addition to the system of a IEEE Std 1003.1-1988 ("POSIX.1") compliant terminal interface) and will no longer compile on systems with older terminal interfaces.

NAME

tsort — topological sort of a directed graph

SYNOPSIS

tsort [**-l**] [**-q**] [*file*]

DESCRIPTION

tsort takes a list of pairs of node names representing directed arcs in a graph and prints the nodes in topological order on standard output. Input is taken from the named *file*, or from standard input if no file is given.

Node names in the input are separated by white space and there must be an even number of node names.

Presence of a node in a graph can be represented by an arc from the node to itself. This is useful when a node is not connected to any other nodes.

If the graph contains a cycle (and therefore cannot be properly sorted), one of the arcs in the cycle is ignored and the sort continues. Cycles are reported on standard error.

The options are as follows:

-l Search for and display the longest cycle. Can take a very long time.

-q Do not display informational messages about cycles. This is primarily intended for building libraries, where optimal ordering is not critical, and cycles occur often.

SEE ALSO

ar(1)

HISTORY

A **tsort** command appeared in Version 7 AT&T UNIX. This **tsort** command and manual page are derived from sources contributed to Berkeley by Michael Rendell of Memorial University of Newfoundland.

NAME

tty — return user's terminal name

SYNOPSIS

tty [**-s**]

DESCRIPTION

The **tty** utility writes the name of the terminal attached to standard input to standard output. The name that is written is the string returned by ttyname(3). If the standard input is not a terminal, the message ''not a tty'' is written. The options are as follows:

-s Don't write the terminal name; only the exit status is affected when this option is specified. The **-s** option is deprecated in favor of the "`test -t 0`" command.

tty exits 0 if the standard input is a terminal, 1 if the standard input is not a terminal, and >1 if an error occurs.

SEE ALSO

test(1), ttyname(3)

STANDARDS

The **tty** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2").

NAME

tvctrl — control display-TV for X680x0

SYNOPSIS

tvctrl *control_number*

DESCRIPTION

tvctrl is display-TV control program for X680x0.

BUGS

Errors are not checked.

NAME

ul — do underlining

SYNOPSIS

ul [**-i**] [**-t** *terminal*] [*name ...*]

DESCRIPTION

ul reads the named files (or standard input if none are given) and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable TERM. The file /usr/share/misc/termcap is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode then that is used instead. If the terminal can overstrike, or handles underlining automatically, **ul** degenerates to cat(1). If the terminal cannot underline, underlining is ignored.

The following options are available:

-i Underlining is indicated by a separate line containing appropriate dashes '−'; this is useful when you want to look at the underlining which is present in an nroff(1) output stream on a crt-terminal.

-t *terminal*
Overrides the terminal type specified in the environment with *terminal*.

ENVIRONMENT

The following environment variable is used:

TERM The TERM variable is used to relate a tty device with its device capability description (see termcap(5)). TERM is set at login time, either by the default terminal type specified in /etc/ttys or as set during the login process by the user in their login file (see for example csh(1)'s **setenv**).

SEE ALSO

colcrt(1), man(1), nroff(1)

HISTORY

The **ul** command appeared in 3.0 BSD.

BUGS

nroff(1) usually outputs a series of backspaces and underlines intermixed with the text to indicate underlining. No attempt is made to optimize the backward motion.

NAME

uname — Print operating system name

SYNOPSIS

uname [**-amnprsv**]

DESCRIPTION

The **uname** utility writes symbols representing one or more system characteristics to the standard output.

The following options are available:

-a Behave as though all of the options **-mnrsv** were specified.

-m print the machine hardware name.

-n print the nodename (the nodename may be a name that the system is known by to a communications network).

-p print the machine processor architecture name.

-r print the operating system release.

-s print the operating system name.

-v print the operating system version.

If no options are specified, **uname** prints the operating system name as if the **-s** option had been specified.

SEE ALSO

hostname(1), machine(1), uname(3)

STANDARDS

The **uname** utility conforms to IEEE Std 1003.2-1992 ("POSIX.2"). The **-p** option is an extension to the standard.

NAME

unifdef, **unifdefall** — remove preprocessor conditionals from code

SYNOPSIS

unifdef [**-ceklst**] [**-I***path*] [**-D***sym*[=*val*]] [**-U***sym*] [**-iD***sym*[=*val*]] [**-iU***sym*] . . . [*file*]
unifdefall [**-I***path*] . . . *file*

DESCRIPTION

The **unifdef** utility selectively processes conditional cpp(1) directives. It removes from a file both the directives and any additional text that they specify should be removed, while otherwise leaving the file alone.

The **unifdef** utility acts on **#if**, **#ifdef**, **#ifndef**, **#elif**, **#else**, and **#endif** lines, and it understands only the commonly-used subset of the expression syntax for **#if** and **#elif** lines. It handles integer values of symbols defined on the command line, the **defined**() operator applied to symbols defined or undefined on the command line, the operators **!**, **<**, **>**, **<=**, **>=**, **==**, **!=**, **&&**, **||**, and parenthesized expressions. Anything that it does not understand is passed through unharmed. It only processes **#ifdef** and **#ifndef** directives if the symbol is specified on the command line, otherwise they are also passed through unchanged. By default, it ignores **#if** and **#elif** lines with constant expressions, or they may be processed by specifying the **-k** flag on the command line.

The **unifdef** utility also understands just enough about C to know when one of the directives is inactive because it is inside a comment, or affected by a backslash-continued line. It spots unusually-formatted preprocessor directives and knows when the layout is too odd to handle.

A script called **unifdefall** can be used to remove all conditional cpp(1) directives from a file. It uses **unifdef -s** and **cpp -dM** to get lists of all the controlling symbols and their definitions (or lack thereof), then invokes **unifdef** with appropriate arguments to process the file.

Available options:

-D*sym*[=*val*]
: Specify that a symbol is defined, and optionally specify what value to give it for the purpose of handling **#if** and **#elif** directives.

-U*sym*
: Specify that a symbol is undefined. If the same symbol appears in more than one argument, the last occurrence dominates.

-c
: If the **-c** flag is specified, then the operation of **unifdef** is complemented, i.e., the lines that would have been removed or blanked are retained and vice versa.

-e
: Because **unifdef** processes its input one line at a time, it cannot remove preprocessor directives that span more than one line. The most common example of this is a directive with a multi-line comment hanging off its right hand end. By default, if **unifdef** has to process such a directive, it will complain that the line is too obfuscated. The **-e** option changes the behaviour so that, where possible, such lines are left unprocessed instead of reporting an error.

-k
: Process **#if** and **#elif** lines with constant expressions. By default, sections controlled by such lines are passed through unchanged because they typically start "`#if 0`" and are used as a kind of comment to sketch out future or past development. It would be rude to strip them out, just as it would be for normal comments.

-l
: Replace removed lines with blank lines instead of deleting them.

-s
: Instead of processing the input file as usual, this option causes **unifdef** to produce a list of symbols that appear in expressions that **unifdef** understands. It is useful in conjunction with the **-dM** option of cpp(1) for creating **unifdef** command lines.

-t
: Disables parsing for C comments and line continuations, which is useful for plain text.

-iD*sym*[=*val*]
-iU*sym*

Ignore **#ifdef**s. If your C code uses **#ifdef**s to delimit non-C lines, such as comments or code which is under construction, then you must tell **unifdef** which symbols are used for that purpose so that it will not try to parse comments and line continuations inside those **#ifdef**s. One specifies ignored symbols with **-iD***sym*[=*val*] and **-iU***sym* similar to **-D***sym*[=*val*] and **-U***sym* above.

-I*path*

Specifies to **unifdefall** an additional place to look for **#include** files. This option is ignored by **unifdef** for compatibility with cpp(1) and to simplify the implementation of **unifdefall**.

The **unifdef** utility copies its output to *stdout* and will take its input from *stdin* if no *file* argument is given.

The **unifdef** utility works nicely with the **-D***sym* option of diff(1).

DIAGNOSTICS

Too many levels of nesting.

Inappropriate **#elif**, **#else** or **#endif**.

Obfuscated preprocessor control line.

Premature EOF (with the line number of the most recent unterminated **#if**).

EOF in comment.

The **unifdef** utility exits 0 if the output is an exact copy of the input, 1 if not, and 2 if in trouble.

SEE ALSO

cpp(1), diff(1)

HISTORY

The **unifdef** command appeared in 4.3BSD. ANSI C support was added in FreeBSD 4.7.

BUGS

Expression evaluation is very limited.

Preprocessor control lines split across more than one physical line (because of comments or backslash-newline) cannot be handled in every situation.

Trigraphs are not recognized.

There is no support for symbols with different definitions at different points in the source file.

The text-mode and ignore functionality does not correspond to modern cpp(1) behaviour.

NAME

uniq — report or filter out repeated lines in a file

SYNOPSIS

uniq [**-c** | **-d** | **-u**] [**-f** *fields*] [**-s** *chars*] [*input_file* [*output_file*]]

DESCRIPTION

The **uniq** utility reads the standard input comparing adjacent lines, and writes a copy of each unique input line to the standard output. The second and succeeding copies of identical adjacent input lines are not written. Repeated lines in the input will not be detected if they are not adjacent, so it may be necessary to sort the files first.

The following options are available:

-c Precede each output line with the count of the number of times the line occurred in the input, followed by a single space.

-d Don't output lines that are not repeated in the input.

-f *fields*
Ignore the first *fields* in each input line when doing comparisons. A field is a string of non-blank characters separated from adjacent fields by blanks. Field numbers are one based, i.e. the first field is field one.

-s *chars*
Ignore the first *chars* characters in each input line when doing comparisons. If specified in conjunction with the **-f** option, the first *chars* characters after the first *fields* fields will be ignored. Character numbers are one based, i.e. the first character is character one.

-u Don't output lines that are repeated in the input.

If additional arguments are specified on the command line, the first such argument is used as the name of an input file, the second is used as the name of an output file.

The **uniq** utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

The historic +*number* and **-***number* options have been deprecated but are still supported in this implementation.

SEE ALSO

sort(1)

STANDARDS

The **uniq** utility is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

NAME

units — conversion program

SYNOPSIS

units [**-f** *filename*] [**-qv**] [[*count*] *from-unit to-unit*]

DESCRIPTION

units converts quantities expression in various scales to their equivalents in other scales. **units** can only handle multiplicative scale changes. It cannot convert Centigrade to Fahrenheit, for example.

The following options are supported:

-f *filename* Specifies the name of the units data file to load.

-q Suppresses prompting of the user for units and the display of statistics about the number of units loaded.

-v Prints the version number.

from-unit to-unit Allows a single unit conversion to be done directly from the command line. No prompting will occur. **units** will print out only the result of this single conversion. The *count* argument can be prepended to the *from-unit* or it can be separate.

units works interactively by prompting the user for input:

```
You have: meters
You want: feet
        * 3.2808399
        / 0.3048

You have: cm^3
You want: gallons
        * 0.00026417205
        / 3785.4118
```

Powers of units can be specified using the "^" character as shown in the example, or by simple concatenation: "cm3" is equivalent to "cm^3". Multiplication of units can be specified by using spaces, a dash or an asterisk. Division of units is indicated by the slash ('/'). Note that multiplication has a higher precedence than division, so "m/s/s" is the same as "m/s^2" or "m/s s". If the user enters incompatible unit types, the **units** program will print a message indicating that the units are not conformable and it will display the reduced form for each unit:

```
You have: ergs/hour
You want: fathoms kg^2 / day
conformability error
        2.7777778e-11 kg m^2 / sec^3
        2.1166667e-05 kg^2 m / sec
```

The conversion information is read from a units data file. The default file includes definitions for most familiar units, abbreviations and metric prefixes. Some constants of nature included are:

pi	ratio of circumference to diameter
c	speed of light
e	charge on an electron

g	acceleration of gravity
force	same as g
mole	Avogadro's number
water	pressure per unit height of water
mercury	pressure per unit height of mercury
au	astronomical unit

"pound" is a unit of mass. Compound names are run together so "poundforce" is a unit of force. British units that differ from their US counterparts are prefixed with "br", and currency is prefixed with its country name: "belgiumfranc", "britainpound". When searching for a unit, if the specified string does not appear exactly as a unit name, then the **units** program will try to remove a trailing "s" or a trailing "es" and check again for a match.

All of these definitions can be read in the standard units file, or you can supply your own file. A unit is specified on a single line by giving its name and an equivalence. One should be careful to define new units in terms of old ones so that a reduction leads to the primitive units which are marked with '!' characters. **units** will not detect infinite loops that could be caused by careless unit definitions.

Prefixes are defined in the same was as standard units, but with a trailing dash at the end of the prefix name.

FILES

`/usr/share/misc/units.lib` the standard units library

AUTHORS

Adrian Mariano ⟨adrian@cam.cornell.edu⟩ or ⟨mariano@geom.umn.edu⟩

BUGS

The effect of including a '/' in a prefix is surprising.

Exponents entered by the user can be only one digit. You can work around this by multiplying several terms.

The user must use '|' to indicate division of numbers and '/' to indicate division of symbols. This distinction should not be necessary.

The program contains various arbitrary limits on the length of the units converted and on the length of the data file.

The program should use a hash table to store units so that it doesn't take so long to load the units list and check for duplication.

NAME

unvis — revert a visual representation of data back to original form

SYNOPSIS

unvis [**-h**] [*file ...*]

DESCRIPTION

unvis is the inverse function of vis(1). It reverts a visual representation of data back to its original form on standard output.

The options are as follows:

-h Decode using the URI encoding from RFC 1808. (VIS_HTTPSTYLE)

SEE ALSO

vis(1), unvis(3), vis(3)

HISTORY

The **unvis** command appears in 4.4BSD.

NAME

uptime — show how long system has been running

SYNOPSIS

uptime

DESCRIPTION

The **uptime** utility displays the current time, the length of time the system has been up, the number of users, and the load average of the system over the last 1, 5, and 15 minutes.

FILES

`/netbsd` system name list

SEE ALSO

w(1)

HISTORY

The **uptime** command appeared in 3.0BSD.

NAME

usbhidaction — perform actions according to USB HID controls

SYNOPSIS

usbhidaction **-c** *config-file* [**-d**] [**-i**] **-f** *device* [**-t** *table*] [**-v**] [*arg ...*]

DESCRIPTION

usbhidaction can be used to execute commands when certain values appear on HID controls. The normal operation for this program is to read the configuration file and then become a daemon and execute commands as the HID items specify. If a read from the HID device fails the program dies; this will make it die when the USB device is unplugged.

The options are as follows:

-c *config-file*
: Specify a path name for the config file. When running as a daemon this needs to be an absolute path for the HUP signal to work.

-d
: Toggle the daemon flag.

-i
: Ignore HID items in the config file that do not exist in the device.

-f *device*
: Specify a path name for the device to operate on. If *device* is numeric, it is taken to be the USB HID device number. If it is a relative path, it is taken to be the name of the device under `/dev`. An absolute path is taken to be the literal device pathname.

-t *table*
: Specify a path name for the HID usage table file.

-v
: Be verbose, and do not become a daemon.

The config file will be re-read if the process gets a HUP signal.

CONFIGURATION

The configuration file has a very simple format. Each line describes an action; if a line begins with a whitespace it is considered a continuation of the previous line. Lines beginning with '#' are considered as comments.

Each line has three parts: a name of a USB HID item, a value for that item, and an action. There must be whitespace between the parts.

The item names are similar to those used by `usbhidctl`(1), but each part must be prefixed by its page name (use the **-v** flag to `usbhidctl`(1) to see the page name). Replace spaces in the item name by underscores.

The value is simply a numeric value. When the item reports this value the action will be performed. If the value is '*' it will match any value.

The action is a normal command that is executed with `system`(3). Before it is executed some substitution will occur: '$n' will be replaced by the nth argument on the command line, '$V' will be replaced by the numeric value of the HID item, '$N' will be replaced by the name of the control, and '$H' will be replaced by the name of the HID device.

FILES

`/usr/share/misc/usb_hid_usages` The HID usage table.

EXAMPLES

The following configuration file can be used to control a pair of Philips USB speakers with the HID controls on the speakers.

```
# Configuration for various Philips USB speakers
Consumer:Consumer_Control.Consumer:Volume_Up                    1
        mixerctl -f $1 -n -w outputs.master++
Consumer:Consumer_Control.Consumer:Volume_Down                      1
        mixerctl -f $1 -n -w outputs.master--
Consumer:Consumer_Control.Consumer:Mute                             1
        mixerctl -f $1 -n -w outputs.mute++
Consumer:Consumer_Control.Consumer:Channel_Top.Microsoft:Base_Up    1
        mixerctl -f $1 -n -w outputs.bass++
Consumer:Consumer_Control.Consumer:Channel_Top.Microsoft:Base_Down  1
        mixerctl -f $1 -n -w outputs.bass--
```

A sample invocation using this configuration would be

```
usbhidaction -f /dev/uhid1 -c conf /dev/mixer1
```

This configuration file can be used for various keyboards with extra keys:

```
# Configuration for extra keyboard keys
Consumer:Consumer_Control.Consumer:Volume_Up                1
        mixerctl -n -w outputs.master++
Consumer:Consumer_Control.Consumer:Volume_Down                      1
        mixerctl -n -w outputs.master--
Consumer:Consumer_Control.Consumer:Mute                             1
        mixerctl -n -w outputs.mute++
Consumer:Consumer_Control.Consumer:Pause/Play                       1
        xmms -p
Consumer:Consumer_Control.Consumer:Stop                             1
        xmms -s
Consumer:Consumer_Control.Consumer:Scan_Previous_Track              1
        xmms -r
Consumer:Consumer_Control.Consumer:Scan_Next_Track          1
        xmms -f
```

And this configuration can be used with, e.g.,

```
usbhidaction -f /dev/uhid0 -c conf -i
```

SEE ALSO

`usbhidctl`(1), `usbhid`(3), `uhid`(4), `usb`(4)

HISTORY

The **`usbhidaction`** command first appeared in NetBSD 1.6.

NAME

usbhidctl — manipulate USB HID devices

SYNOPSIS

usbhidctl -f *device* [**-t** *table*] [**-lv**] **-a**
usbhidctl -f *device* [**-t** *table*] [**-v**] **-r**
usbhidctl -f *device* [**-t** *table*] [**-lnv**] *item* [...]
usbhidctl -f *device* [**-t** *table*] [**-z**] **-w** *item=value* [...]

DESCRIPTION

usbhidctl can be used to output or modify the state of a USB HID (Human Interface Device). If a list of items is present on the command line, then **usbhidctl** prints the current value of those items for the specified device. If the **-w** flag is specified **usbhidctl** attempts to set the specified items to the given values.

The options are as follows:

-a Show all items and their current values. This option fails if the device does not support the GET_REPORT command.

-f *device*
Specify a path name for the device to operate on. If *device* is numeric, it is taken to be the USB HID device number. If it is a relative path, it is taken to be the name of the device under /dev. An absolute path is taken to be the literal device pathname.

-l Loop and dump the device data every time it changes. Only 'input' items are displayed in this mode.

-n Suppress printing of the item name when querying specific items. Only output the current value.

-r Dump the USB HID report descriptor.

-t *table*
Specify a path name for the HID usage table file.

-v Be verbose. Repeating this option increases verbosity.

-w Change item values. Only 'output' and 'feature' kinds can be set with this option.

-z Reset all feature and output flags to zero before attempting to change them. May be required for changing item values (via **-w**) on devices that don't implement GET_REPORT.

FILES

/usr/share/misc/usb_hid_usages The default HID usage table.

SYNTAX

usbhidctl parses the names of items specified on the command line against the human interface items reported by the USB device. Each human interface item is mapped from its native form to a human readable name, using the HID usage table file. Command line items are compared with the generated item names, and the USB HID device is operated on when a match is found.

Each human interface item is named by the "page" it appears in, the "usage" within that page, and the list of "collections" containing the item. Each collection in turn is also identified by page, and the usage within that page.

On the **usbhidctl** command line the page name is separated from the usage name with the character '**:**'. The collections are separated by the character '**.**'.

As an alternative notation in items on the command line, the native numeric value for the page name or usage can be used instead of the full human readable page name or usage name. Numeric values can be specified in decimal, octal or hexadecimal.

Some devices give the same name to more than one item. **usbhidctl** supports isolating each item by appending a '**#**'. character and a decimal item instance number, starting at zero.

EXAMPLES

On a standard USB mouse the item

```
Generic_Desktop:Mouse.Generic_Desktop:Pointer.Button:Button_2
```

reflects the current status of button 2. The "button 2" item is encapsulated within two collections, the "Mouse" collection in the "Generic Desktop" page, and the "Pointer" collection in the "Generic Desktop" page. The item itself is the usage "Button_2" in the "Button" page.

An item can generally be named by omitting one or more of the page names. For example the "button 2" item would usually just be referred to on the command line as:

```
usbhidctl -f /dev/mouse Mouse.Pointer.Button_2
```

Items can also be named by referring to parts of the item name with the numeric representation of the native HID usage identifiers. This is most useful when items are missing from the HID usage table. The page identifier for the "Generic Desktop" page is 1, and the usage identifier for the usage "Button_2" is 2, so the following can be used to refer to the "button 2" item:

```
usbhidctl -f /dev/mouse 1:Mouse.1:Pointer.Button:2
```

Devices with human interface outputs can be manipulated with the **-w** option. For example, some USB mice have a Light Emitting Diode under software control as usage 2 under page 0xffff, in the "Mouse" collection. The following can be used to switch this LED off:

```
usbhidctl -f /dev/mouse -w Mouse.0xffff:2=0
```

The output below is from a device that uses the same name repeatedly.

```
% usbhidctl -f /dev/uhid0 -a
Consumer_Control.Volume_Up=0
Consumer_Control.Volume_Down=0
Consumer_Control.Mute=0
Consumer_Control.Unassigned=0
Consumer_Control.Unassigned=0
```

The "Consumer_Control.Unassigned" name is used twice. Each can be individually accessed by providing an instance number. For example, to set the value for the first item:

```
usbhidctl -f /dev/uhid0 -w 'Consumer_Control.Unassigned#0=1'
```

SEE ALSO

usbhidaction(1), usbhid(3), uhid(4), usb(4)

HISTORY

The **usbhidctl** command first appeared in NetBSD 1.4.

AUTHORS

David Sainty ⟨David.Sainty@dtsp.co.nz⟩

NAME

users — list current users

SYNOPSIS

users

DESCRIPTION

users lists the login names of the users currently on the system, in sorted order, space separated, on a single line.

FILES

`/var/run/utmp`

SEE ALSO

`finger`(1), `last`(1), `who`(1), `utmp`(5)

HISTORY

The **users** command appeared in 3.0BSD.

NAME

utoppya — Topfield TF5000PVR file manipulation program

SYNOPSIS

utoppya [**-f** *device*] *command ...*

DESCRIPTION

utoppya is the userland interface to the utoppy(4) device driver.

The options are as follows:

-f Communicate with *device* instead of the default /dev/utoppy0.

Using the services of the utoppy(4) driver, **utoppya** can perform the following operations, specified by the *command* operand:

df Display disk size and free space

ls [*directory*]
List the files in the specified *directory*. Default is the Toppy's root directory.

mkdir ⟨*directory*⟩
Make the specified *directory*. You must supply the full pathname to *directory*.

rm ⟨*pathname*⟩
Delete the file or directory specified by *pathname*.

rename ⟨*source*⟩⟨*target*⟩
Rename the file or directory specified by the *source* operand to the destination file or directory specified by the *target* operand.

get [**-prt**]⟨*toppyfile*⟩[*localfile*]
Copy *toppyfile* from the Toppy to *localfile* on the local filesystem. If *localfile* is omitted, the file will be copied into the current directory and will be named using the last component of the *toppyfile* operand. If *localfile* is '-', then *toppyfile* will be copied to the standard output.

The following options are available for the **get** command:

-p Display a progress bar.

-r This option is useful if you wish to resume a previously interrupted **get** command. Instead of restarting from the beginning of the file, the transfer will resume where it left off.

-t Enable Turbo mode. This instructs the Toppy to drop everything and concentrate on transferring the file as quickly as possible. You will be able to watch live TV, but all other functions, including changing channel via the remote control, will be inoperative.

put [**-prt**]⟨*localfile*⟩⟨*toppyfile*⟩
Copy *localfile* on the local filesystem to *toppyfile* on the Toppy. If *toppyfile* specifies a directory on the Toppy, the last component of the *localfile* operand will be appended to the *toppyfile* operand.

The options described for the **get** command (above) also apply to the **put** command.

FILES

/dev/utoppy0 The default Topfield TF5000PVR instance.

SEE ALSO

usb(4), utoppy(4)

HISTORY

The **utoppya** command first appeared in NetBSD 4.0 and was inspired by ftpd-topfield written by Steve Bennett ⟨msteveb at ozemail.com.au⟩ and puppy written by Peter Urbanec ⟨toppy at urbanec.net⟩.

AUTHORS

Steve C. Woodford ⟨scw@netbsd.org⟩

NAME

uuencode, **uudecode** — encode/decode a binary file

SYNOPSIS

uuencode [**-m**] [*file*] *name*
uudecode [**-m** | **-p**] [*file ...*]

DESCRIPTION

uuencode and **uudecode** are used to transmit binary files over transmission mediums that do not support other than simple ASCII data.

The following options are available:

-m Use base64 encoding.

uuencode reads *file* (or by default the standard input) and writes an encoded version to the standard output. The encoding uses only printing ASCII characters and includes the mode of the file and the operand *name* for use by **uudecode**.

uudecode transforms *uuencoded* files (or by default, the standard input) into the original form. The resulting file is named *name* and will have the mode of the original file except that setuid and execute bits are not retained; if the **-p** option is specified, the data will be written to the standard output. **uudecode** ignores any leading and trailing lines.

EXIT STATUS

The **uudecode** and **uuencode** utilities exits 0 on success, and >0 if an error occurs.

EXAMPLES

The following example packages up a source tree, compresses it, uuencodes it and mails it to a user on another system. When **uudecode** is run on the target system, the file ‘‘src_tree.tar.Z’’ will be created which may then be uncompressed and extracted into the original tree.

```
tar cf - src_tree | compress |
uuencode src_tree.tar.Z | mail sys1!sys2!user
```

SEE ALSO

compress(1), mail(1), uucp(1), uuencode(5)

STANDARDS

The **uudecode** and **uuencode** utilities conform to IEEE Std 1003.2-1992 (“POSIX.2”).

HISTORY

The **uudecode** and **uuencode** utilities appeared in 4.0BSD.

BUGS

The encoded form of the file is expanded by 35% (3 bytes become 4 plus control information).

NAME

uuidgen — generate universally unique identifiers

SYNOPSIS

uuidgen [**-1s**] [**-n** *count*] [**-o** *filename*]

DESCRIPTION

The **uuidgen** utility by default generates a single universally unique identifier (UUID), also known as a globally unique identifier (GUID). By default, **uuidgen** generates a single UUID and outputs it in the standard string representation to stdout. The following options can be used to change the behavior of **uuidgen**:

-1 This option only has effect if multiple identifiers are to be generated and instructs **uuidgen** to not generate them in batch, but one at a time.

-n *count* This option controls the number of identifiers generated. By default, multiple identifiers are generated in batch.

-o *filename* Redirect output to *filename* instead of stdout.

-s Output UUIDs as initialized C structures, rather than in the standard string format.

Batched generation yields a dense set of identifiers in such a way that there is no identifier that is larger than the smallest identifier in the set and smaller than the largest identifier in the set and that is not already in the set.

When generating the identifiers one at a time, the identifiers will be close to each other, but operating system latency and processing time will be reflected in the distance between two successive identifiers.

DIAGNOSTICS

The **uuidgen** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

uuidgen(2), uuid(3)

HISTORY

The **uuidgen** command first appeared in NetBSD 3.0.

NAME

vacation — return “I am not here” indication

SYNOPSIS

vacation **-dIi** [**-f** *databasefile*] [**-m** *messagefile*] [**-r** *interval*] [**-t** *interval*]
vacation **-dj** [**-a** *alias*] [**-F** *F|R|S*] [**-f** *databasefile*] [**-m** *messagefile*]
[**-s** *sender*] [**-T** *A|D*] *login*

DESCRIPTION

vacation returns a message to the sender of a message telling them that you are currently not reading your mail. The intended use is in a `.forward` file. For example, your `.forward` file might have:

```
\eric, "|/usr/bin/vacation -a allman eric"
```

which would send messages to you (assuming your login name was eric) and reply to any messages for “eric” or “allman”.

Available options:

-a *alias*
: Handle messages for *alias* in the same manner as those received for the user’s login name.

-d
: Turn debugging on; don’t send an actual message, but print it on stdout.

-f *database_file*
: Use the specified *database_file* prefix and append `.db` to it instead of `$HOME/.vacation.db`.

-F *F|R|S*
: Make **vacation** additionally look in From: (F), Return-Path: (R), or Sender: (S) headers to determine the From: field.

-i

-I
: Initialize the vacation database files. It should be used before you modify your `.forward` file.

-j
: Do not check if the recipient is present in the `To:` or `Cc:` lines. Usage of this option is strongly discouraged because it will result in **vacation** replying to mailing lists or other inappropriate places (e.g., messages that you have been `Bcc` to).

-m *message_file*
: Use *message_file* instead of `$HOME/.vacation.msg`.

-s *sender*
: Reply to *sender* instead of the the value read from the message.

-r *interval*

-t *interval*
: Set the reply interval to *interval* days. If the *interval* number is followed by `w`, `d`, `h`, `m`, or `s` then the number is interpreted as weeks, days, hours, minutes, or seconds respectively. The default *interval* is one week. An *interval* of “0” means that a reply is sent to each message, and an interval of “`infinite`” (actually, any non-numeric character) will never send more than one reply. It should be noted that intervals of “`0`” are quite dangerous, as it allows mailers to get into “I am on vacation” loops.

-T *A|D*
: Make **vacation** additionally look in Apparently-To: (A) or Delivered-To: (D) headers to determine the To: field.

No message will be sent unless *login* (or an *alias* supplied using the **-a** option) is part of either the "To:" or "Cc:" headers of the mail. No messages from "???-REQUEST", "Postmaster", "UUCP", "MAILER", or "MAILER-DAEMON" will be replied to (where these strings are case insensitive) nor is a notification sent if a "Precedence: bulk" "Precedence: list" or "Precedence: junk" line is included in the mail headers. The people who have sent you messages are maintained as a db(3) database in the file `.vacation.db` in your home directory.

vacation expects a file `.vacation.msg`, in your home directory, containing a message to be sent back to each sender. It should be an entire message (including headers). If the message contains the string `$SUBJECT` then it will will be replaced with the subject of the original message. For example, it might contain:

```
From: eric@CS.Berkeley.EDU (Eric Allman)
Subject: I am on vacation
Delivered-By-The-Graces-Of: The Vacation program
Precedence: bulk

I am on vacation until July 22.
Your mail regarding "$SUBJECT" will be read when I return.
If you have something urgent, please contact Keith Bostic
<bostic@CS.Berkeley.EDU>.
--eric
```

vacation reads the first line from the standard input for a UNIX "From" line to determine the sender. `sendmail`(8) includes this "From" line automatically.

Fatal errors, such as calling **vacation** with incorrect arguments, or with non-existent *logins*, are logged in the system log file, using `syslog`(3).

FILES

`~/.vacation.db` database file
`~/.vacation.msg` message to send

SEE ALSO

`syslog`(3), `sendmail`(8)

HISTORY

The **vacation** command appeared in 4.3BSD.

BUGS

Adding **-t** *A* or **-t** *D* should only be done for misconfigured or non-compliant MTAs. Doing so may auto-respond to messages that were not supposed to be replied to.

NAME

vgrind — grind nice listings of programs

SYNOPSIS

vgrind [**-**] [**-W**] [**-d** *file*] [**-f**] [**-h** *header*] [**-l** *language*] [**-n**] [**-sn**] [**-t**] [**-x**] *file ...*

DESCRIPTION

vgrind formats the program sources which are arguments in a nice style using troff(1). Comments are placed in italics, keywords in bold face, and the name of the current function is listed down the margin of each page as it is encountered.

vgrind runs in two basic modes, filter mode (see the **-f** option) or regular mode. In filter mode **vgrind** acts as a filter in a manner similar to tbl(1). The standard input is passed directly to the standard output except for lines bracketed by the *troff-like* macros:

.vS starts processing

.vE ends processing

These lines are formatted as described above. The output from this filter can be passed to troff(1) for output. There need be no particular ordering with eqn(1) or tbl(1).

In regular mode **vgrind** accepts input files, processes them, and passes them to troff(1) for output.

In both modes **vgrind** passes any lines beginning with a decimal point without conversion.

The options are:

- forces input to be taken from standard input (default if **-f** is specified)

-W forces output to the (wide) Versatec printer rather than the (narrow) Varian

-d *file* specifies an alternative language definitions file (default is /usr/share/misc/vgrindefs)

-f forces filter mode

-h *header* specifies a particular header to put on every output page (default is the file name)

-l specifies the language to use. Currently known are PASCAL (**-l***p*), MODEL (**-l***m*), C (**-l***c* or the default), CSH (**-l***csh*), SHELL (**-l***sh*), RATFOR (**-l***r*), MODULA2 (**-l***mod2*), YACC (**-l***yacc*), LISP (**-l***isp*), and ICON (**-l***I*).

-n forces no keyword bolding

-s specifies a point size to use on output (exactly the same as the argument of a .ps)

-t similar to the same option in troff(1) causing formatted text to go to the standard output

-x outputs the index file in a ‘‘pretty’’ format. The index file itself is produced whenever **vgrind** is run with a file called index in the current directory. The index of function definitions can then be run off by giving **vgrind** the **-x** option and the file index as argument.

FILES

index	file where source for index is created
/usr/share/tmac/vgrind.tmac	macro package

`/usr/libexec/vfontedpr`	preprocessor
`/usr/share/misc/vgrindefs`	language descriptions

SEE ALSO

`lpr`(1), `troff`(1), `getcap`(3), `vgrindefs`(5)

HISTORY

The **`vgrind`** command appeared in 3.0 BSD.

BUGS

Vfontedpr assumes that a certain programming style is followed:

For C − function names can be preceded on a line only by spaces, tabs, or an asterisk. The parenthesized arguments must also be on the same line.

For PASCAL − function names need to appear on the same line as the keywords *function* or *procedure*.

For MODEL − function names need to appear on the same line as the keywords *is beginproc*.

If these conventions are not followed, the indexing and marginal function name comment mechanisms will fail.

More generally, arbitrary formatting styles for programs mostly look bad. The use of spaces to align source code fails miserably; if you plan to **`vgrind`** your program you should use tabs. This is somewhat inevitable since the font used by **`vgrind`** is variable width.

The mechanism of `ctags`(1) in recognizing functions should be used here.

Filter mode does not work in documents using the **`-me`** or **`-ms`** macros. (So what use is it anyway?)

NAME

ex, vi, view − text editors

SYNOPSIS

ex [**-eFRrSsv**] [**-c** *cmd*] [**-t** *tag*] [**-w** *size*] [file ...]
vi [**-eFlRrSv**] [**-c** *cmd*] [**-t** *tag*] [**-w** *size*] [file ...]
view [**-eFRrSv**] [**-c** *cmd*] [**-t** *tag*] [**-w** *size*] [file ...]

LICENSE

DESCRIPTION

Vi is a screen oriented text editor. *Ex* is a line-oriented text editor. *Ex* and *vi* are different interfaces to the same program, and it is possible to switch back and forth during an edit session. *View* is the equivalent of using the **−R** (read-only) option of *vi*.

This manual page is the one provided with the *nex/nvi* versions of the *ex/vi* text editors. *Nex/nvi* are intended as bug-for-bug compatible replacements for the original Fourth Berkeley Software Distribution (4BSD) *ex* and *vi* programs. For the rest of this manual page, *nex/nvi* is used only when it's necessary to distinguish it from the historic implementations of *ex/vi*.

This manual page is intended for users already familiar with *ex/vi*. Anyone else should almost certainly read a good tutorial on the editor before this manual page. If you're in an unfamiliar environment, and you absolutely have to get work done immediately, read the section after the options description, entitled ''Fast Startup''. It's probably enough to get you going.

The following options are available:

−c Execute **cmd** immediately after starting the edit session. Particularly useful for initial positioning in the file, however **cmd** is not limited to positioning commands. This is the POSIX 1003.2 interface for the historic ''+cmd'' syntax. *Nex/nvi* supports both the old and new syntax.

−e Start editing in ex mode, as if the command name were *ex*.

−F Don't copy the entire file when first starting to edit. (The default is to make a copy in case someone else modifies the file during your edit session.)

−l Start editing with the lisp and showmatch options set.

−R Start editing in read-only mode, as if the command name was *view*, or the **readonly** option was set.

−r Recover the specified files, or, if no files are specified, list the files that could be recovered. If no recoverable files by the specified name exist, the file is edited as if the **−r** option had not been specified.

−S Run with the **secure** edit option set, disallowing all access to external programs.

−s Enter batch mode; applicable only to *ex* edit sessions. Batch mode is useful when running *ex* scripts. Prompts, informative messages and other user oriented message are turned off, and no startup files or environmental variables are read. This is the POSIX 1003.2 interface for the historic ''−'' argument. *Nex/nvi* supports both the old and new syntax.

−t Start editing at the specified tag. (See *ctags*(1)).

−w Set the initial window size to the specified number of lines.

−v Start editing in vi mode, as if the command name was *vi* or *view*.

Command input for *ex/vi* is read from the standard input. In the *vi* interface, it is an error if standard input is not a terminal. In the *ex* interface, if standard input is not a terminal, *ex* will read commands from it regardless, however, the session will be a batch mode session, exactly as if the **−s** option had been specified.

Ex/vi exits 0 on success, and greater than 0 if an error occurs.

FAST STARTUP

This section will tell you the minimum amount that you need to do simple editing tasks using *vi*. If you've never used any screen editor before, you're likely to have problems even with this simple introduction. In that case you should find someone that already knows *vi* and have them walk you through this section.

Vi is a screen editor. This means that it takes up almost the entire screen, displaying part of the file on each screen line, except for the last line of the screen. The last line of the screen is used for you to give commands to *vi*, and for *vi* to give information to you.

The other fact that you need to understand is that *vi* is a modeful editor, i.e. you are either entering text or you are executing commands, and you have to be in the right mode to do one or the other. You will be in command mode when you first start editing a file. There are commands that switch you into input mode. There is only one key that takes you out of input mode, and that is the <escape> key. (Key names are written using less-than and greater-than signs, e.g. <escape> means the ''escape'' key, usually labeled ''esc'' on your terminal's keyboard.) If you're ever confused as to which mode you're in, keep entering the <escape> key until *vi* beeps at you. (Generally, *vi* will beep at you if you try and do something that's not allowed. It will also display error messages.)

To start editing a file, enter the command ''vi file_name<carriage-return>''. The command you should enter as soon as you start editing is '':set verbose showmode<carriage-return>''. This will make the editor give you verbose error messages and display the current mode at the bottom of the screen.

The commands to move around the file are:

h Move the cursor left one character.

j Move the cursor down one line.

k Move the cursor up one line.

l Move the cursor right one character.

<cursor-arrows>
The cursor arrow keys should work, too.

/text<carriage-return>
Search for the string ''text'' in the file, and move the cursor to its first character.

The commands to enter new text are:

a Append new text, *after* the cursor.

i Insert new text, *before* the cursor.

o Open a new line below the line the cursor is on, and start entering text.

O Open a new line above the line the cursor is on, and start entering text.

<escape>
Once you've entered input mode using the one of the **a**, **i**, **O** or **o** commands, use **<escape>** to quit entering text and return to command mode.

The commands to copy text are:

yy Copy the line the cursor is on.

p Append the copied line after the line the cursor is on.

The commands to delete text are:

dd Delete the line the cursor is on.

x Delete the character the cursor is on.

The commands to write the file are:

:w<carriage-return>
Write the file back to the file with the name that you originally used as an argument on the *vi* command line.

:w file_name<carriage-return>
Write the file back to the file with the name ‘‘file_name’’.

The commands to quit editing and exit the editor are:

:q<carriage-return>
Quit editing and leave vi (if you’ve modified the file, but not saved your changes, *vi* will refuse to quit).

:q!<carriage-return>
Quit, discarding any modifications that you may have made.

One final caution. Unusual characters can take up more than one column on the screen, and long lines can take up more than a single screen line. The above commands work on ‘‘physical’’ characters and lines, i.e. they affect the entire line no matter how many screen lines it takes up and the entire character no matter how many screen columns it takes up.

VI COMMANDS

The following section describes the commands available in the command mode of the *vi* editor. In each entry below, the tag line is a usage synopsis for the command character.

[count] <control-A>
Search forward *count* times for the current word.

[count] <control-B>
Page backwards *count* screens.

[count] <control-D>
Scroll forward *count* lines.

[count] <control-E>
Scroll forward *count* lines, leaving the current line and column as is, if possible.

[count] <control-F>
Page forward *count* screens.

<control-G>
Display the file information.

<control-H>

[count] h
Move the cursor back *count* characters in the current line.

[count] <control-J>

[count] <control-N>

[count] j
Move the cursor down *count* lines without changing the current column.

<control-L>

<control-R>
Repaint the screen.

[count] <control-M>

[count] +
Move the cursor down *count* lines to the first nonblank character of that line.

[count] <control-P>

[count] k
Move the cursor up *count* lines, without changing the current column.

<control-T>
Return to the most recent tag context.

<control-U>
Scroll backwards *count* lines.

<control-W>
Switch to the next lower screen in the window, or, to the first screen if there are no lower screens in the window.

<control-Y>
Scroll backwards *count* lines, leaving the current line and column as is, if possible.

<control-Z>
Suspend the current editor session.

<escape>
Execute *ex* commands or cancel partial commands.

<control-]>
Push a tag reference onto the tag stack.

<control-^>
Switch to the most recently edited file.

[count] <space>

[count] l
Move the cursor forward *count* characters without changing the current line.

[count] ! motion shell-argument(s)
Replace text with results from a shell command.

[count] # #|+|-
Increment or decrement the cursor number.

[count] $
Move the cursor to the end of a line.

% Move to the matching character.

& Repeat the previous substitution command on the current line.

’<character>

‘<character>
Return to a context marked by the character <*character*>.

[count] (
Back up *count* sentences.

[count])
Move forward *count* sentences.

[count] ,
Reverse find character *count* times.

[count] -
Move to first nonblank of the previous line, *count* times.

[count] .
Repeat the last *vi* command that modified text.

/RE<carriage-return>

/RE/ [offset]<carriage-return>

?RE<carriage-return>

?RE? [offset]<carriage-return>

N

n Search forward or backward for a regular expression.

0 Move to the first character in the current line.

: Execute an ex command.

[count] ;
Repeat the last character find *count* times.

[count] < motion

[count] > motion
Shift lines left or right.

@ buffer
Execute a named buffer.

[count] A
Enter input mode, appending the text after the end of the line.

[count] B
Move backwards *count* bigwords.

[buffer] [count] C
Change text from the current position to the end-of-line.

[buffer] D
Delete text from the current position to the end-of-line.

[count] E
Move forward *count* end-of-bigwords.

[count] F <character>
Search *count* times backward through the current line for <*character*>.

[count] G
Move to line *count*, or the last line of the file if *count* not specified.

[count] H
Move to the screen line *count* - *1* lines below the top of the screen.

[count] I
Enter input mode, inserting the text at the beginning of the line.

[count] J
Join lines.

[count] L
Move to the screen line *count* - *1* lines above the bottom of the screen.

M Move to the screen line in the middle of the screen.

[count] O
Enter input mode, appending text in a new line above the current line.

[buffer] P
Insert text from a buffer.

Q Exit *vi* (or visual) mode and switch to *ex* mode.

[count] R
Enter input mode, replacing the characters in the current line.

[buffer] [count] S
Substitute *count* lines.

[count] T <character>
Search backwards, *count* times, through the current line for the character *after* the specified <*character*>.

U Restore the current line to its state before the cursor last moved to it.

[count] W
Move forward *count* bigwords.

[buffer] [count] X
Delete *count* characters before the cursor.

[buffer] [count] Y
Copy (or ‘‘yank’’) *count* lines into the specified buffer.

ZZ Write the file and exit *vi*.

[count] [[
Back up *count* section boundaries.

[count]]]
Move forward *count* section boundaries.

^ Move to first nonblank character on the current line.

[count] _
Move down *count - 1* lines, to the first nonblank character.

[count] a
Enter input mode, appending the text after the cursor.

[count] b
Move backwards *count* words.

[buffer] [count] c motion
Change a region of text.

[buffer] [count] d motion
Delete a region of text.

[count] e
Move forward *count* end-of-words.

[count] f<character>
Search forward, *count* times, through the rest of the current line for <*character*>.

[count] i
Enter input mode, inserting the text before the cursor.

m <character>
Save the current context (line and column) as <*character*>.

[count] o
Enter input mode, appending text in a new line under the current line.

[buffer] p
Append text from a buffer.

[count] r <character>
Replace *count* characters.

[buffer] [count] s
Substitute *count* characters in the current line starting with the current character.

[count] t <character>
Search forward, *count* times, through the current line for the character immediately *before* <*character*>.

u Undo the last change made to the file.

[count] w
Move forward *count* words.

[buffer] [count] x
Delete *count* characters.

[buffer] [count] y motion
Copy (or ‘‘yank’’) a text region specified by the *count* and motion into a buffer.

[count1] z [count2] -|.|+|^|<carriage-return>
Redraw, optionally repositioning and resizing the screen.

[count] {
Move backward *count* paragraphs.

[count] |
Move to a specific *column* position on the current line.

[count] }
Move forward *count* paragraphs.

[count] ~
Reverse the case of the next *count* character(s).

[count] ~ motion
Reverse the case of the characters in a text region specified by the *count* and *motion*.

<interrupt>
Interrupt the current operation.

VI TEXT INPUT COMMANDS

The following section describes the commands available in the text input mode of the *vi* editor.

<nul> Replay the previous input.

<control-D>
Erase to the previous **shiftwidth** column boundary.

^<control-D>
Erase all of the autoindent characters, and reset the autoindent level.

0<control-D>
Erase all of the autoindent characters.

<control-T>
Insert sufficient <*tab*> and <*space*> characters to move forward to the next **shiftwidth** column boundary.

<erase>

<control-H>
Erase the last character.

<literal next>
Quote the next character.

<escape>
Resolve all text input into the file, and return to command mode.

<line erase>
Erase the current line.

<control-W>

<word erase>
Erase the last word. The definition of word is dependent on the **altwerase** and **ttywerase** options.

<control-X>[0-9A-Fa-f]+
Insert a character with the specified hexadecimal value into the text.

<interrupt>
Interrupt text input mode, returning to command mode.

EX COMMANDS

The following section describes the commands available in the *ex* editor. In each entry below, the tag line is a usage synopsis for the command.

<end-of-file>
Scroll the screen.

! argument(s)

[range]! argument(s)
Execute a shell command, or filter lines through a shell command.

" A comment.

[range] nu[mber] [count] [flags]

[range] # [count] [flags]
Display the selected lines, each preceded with its line number.

@ buffer

*** buffer**
Execute a buffer.

[line] a[ppend][!]
The input text is appended after the specified line.

[range] c[hange][!] [count]
The input text replaces the specified range.

cs[cope] add | find | help | kill | reset
Execute a Cscope command.

[range] d[elete] [buffer] [count] [flags]
Delete the lines from the file.

di[splay] b[uffers] | c[onnections] | s[creens] | t[ags]
Display buffers, Cscope connections, screens or tags.

[Ee][dit][!] [+cmd] [file]

[Ee]x[!] [+cmd] [file]
Edit a different file.

exu[sage] [command]
Display usage for an *ex* command.

f[ile] [file]
Display and optionally change the file name.

[Ff]g [name]
Vi mode only. Foreground the specified screen.

[range] g[lobal] /pattern/ [commands]

[range] v /pattern/ [commands]
Apply commands to lines matching (or not matching) a pattern.

he[lp] Display a help message.

[line] i[nsert][!]
The input text is inserted before the specified line.

[range] j[oin][!] [count] [flags]
Join lines of text together.

[range] l[ist] [count] [flags]
Display the lines unambiguously.

map[!] [lhs rhs]
Define or display maps (for *vi* only).

[line] ma[rk] <character>

[line] k <character>
Mark the line with the mark <*character*>.

[range] m[ove] line
Move the specified lines after the target line.

mk[exrc][!] file
Write the abbreviations, editor options and maps to the specified file.

[Nn][ext][!] [file ...]
Edit the next file from the argument list.

[line] o[pen] /pattern/ [flags]
Enter open mode.

pre[serve]
Save the file in a form that can later be recovered using the *ex* **−r** option.

[Pp]rev[ious][!]
Edit the previous file from the argument list.

[range] p[rint] [count] [flags]
Display the specified lines.

[line] pu[t] [buffer]
Append buffer contents to the current line.

q[uit][!]
End the editing session.

[line] r[ead][!] [file]
Read a file.

rec[over] file
Recover *file* if it was previously saved.

res[ize] [+|-]size
Vi mode only. Grow or shrink the current screen.

rew[ind][!]
Rewind the argument list.

se[t] [option[=[value]] ...] [nooption ...] [option? ...] [all]
Display or set editor options.

sh[ell] Run a shell program.

so[urce] file
Read and execute *ex* commands from a file.

[range] s[ubstitute] [/pattern/replace/] [options] [count] [flags]

[range] & [options] [count] [flags]

[range] ~ [options] [count] [flags]
Make substitutions.

su[spend][!]

st[op][!]

<suspend>
Suspend the edit session.

[Tt]a[g][!] tagstring
Edit the file containing the specified tag.

tagn[ext][!]
Edit the file containing the next context for the current tag.

tagp[op][!] [file | number]
Pop to the specified tag in the tags stack.

tagp[rev][!]
Edit the file containing the previous context for the current tag.

unm[ap][!] lhs
Unmap a mapped string.

ve[rsion]
Display the version of the *ex/vi* editor.

[line] vi[sual] [type] [count] [flags]
Ex mode only. Enter *vi*.

[Vi]i[sual][!] [+cmd] [file]
Vi mode only. Edit a new file.

viu[sage] [command]
Display usage for a *vi* command.

[range] w[rite][!] [>>] [file]

[range] w[rite] [!] [file]

[range] wn[!] [>>] [file]

[range] wq[!] [>>] [file]
Write the file.

[range] x[it][!] [file]
Write the file if it has been modified.

[range] ya[nk] [buffer] [count]
Copy the specified lines to a buffer.

[line] z [type] [count] [flags]
Adjust the window.

SET OPTIONS

There are a large number of options that may be set (or unset) to change the editor's behavior. This section describes the options, their abbreviations and their default values.

In each entry below, the first part of the tag line is the full name of the option, followed by any equivalent abbreviations. The part in square brackets is the default value of the option. Most of the options are boolean, i.e. they are either on or off, and do not have an associated value.

Options apply to both *ex* and *vi* modes, unless otherwise specified.

altwerase [off]
Vi only. Select an alternate word erase algorithm.

autoindent, ai [off]
Automatically indent new lines.

autoprint, ap [off]
Ex only. Display the current line automatically.

autowrite, aw [off]
Write modified files automatically when changing files.

backup [""]
Backup files before they are overwritten.

beautify, bf [off]
Discard control characters.

cdpath [environment variable CDPATH, or current directory]
The directory paths used as path prefixes for the **cd** command.

cedit [no default]
Set the character to edit the colon command-line history.

columns, co [80]
Set the number of columns in the screen.

comment [off]
Vi only. Skip leading comments in shell, C and C++ language files.

directory, dir [environment variable TMPDIR, or /tmp]
The directory where temporary files are created.

edcompatible, ed [off]
Remember the values of the ''c'' and ''g'' suffices to the **substitute** commands, instead of initializing them as unset for each new command.

errorbells, eb [off]
Ex only. Announce error messages with a bell.

exrc, ex [off]
Read the startup files in the local directory.

extended [off]
Regular expressions are extended (i.e. *egrep*(1)−style) expressions.

filec [no default]
Set the character to perform file path completion on the colon command line.

flash [on]
Flash the screen instead of beeping the keyboard on error.

hardtabs, ht [8]
Set the spacing between hardware tab settings.

iclower [off]
Makes all Regular Expressions case-insensitive, as long as an upper-case letter does not appear in the search string.

ignorecase, ic [off]
Ignore case differences in regular expressions.

keytime [6]
The 10th's of a second *ex/vi* waits for a subsequent key to complete a key mapping.

leftright [off]
Vi only. Do left-right scrolling.

lines, li [24]
Vi only. Set the number of lines in the screen.

lisp [off]
Vi only. Modify various search commands and options to work with Lisp. *This option is not yet implemented.*

list [off]
Display lines in an unambiguous fashion.

lock [on]
Attempt to get an exclusive lock on any file being edited, read or written.

magic [on]
Treat certain characters specially in regular expressions.

matchtime [7]
Vi only. The 10th's of a second *ex/vi* pauses on the matching character when the **showmatch** option is set.

mesg [on]
Permit messages from other users.

modelines, modeline [off]
Read the first and last few lines of each file for *ex* commands. *This option will never be implemented.*

noprint [""]
Characters that are never handled as printable characters.

number, nu [off]
Precede each line displayed with its current line number.

octal [off]
Display unknown characters as octal numbers, instead of the default hexadecimal.

open [on]
Ex only. If this option is not set, the **open** and **visual** commands are disallowed.

optimize, opt [on]
Vi only. Optimize text throughput to dumb terminals. *This option is not yet implemented.*

paragraphs, para [IPLPPPQPP LIpplpipbp]
Vi only. Define additional paragraph boundaries for the **{** and **}** commands.

path [] Define additional directories to search for files being edited.

print [""]
Characters that are always handled as printable characters.

prompt [on]
Ex only. Display a command prompt.

readonly, ro [off]
Mark the file and session as read-only.

recdir [/var/tmp/vi.recover]
The directory where recovery files are stored.

redraw, re [off]
Vi only. Simulate an intelligent terminal on a dumb one. *This option is not yet implemented.*

remap [on]
Remap keys until resolved.

report [5]
Set the number of lines about which the editor reports changes or yanks.

ruler [off]
Vi only. Display a row/column ruler on the colon command line.

scroll, scr [window / 2]
Set the number of lines scrolled.

searchincr [off]
Makes the **/** and **?** commands incremental.

sections, sect [NHSHH HUnhsh]
Vi only. Define additional section boundaries for the **[[** and **]]** commands.

secure [off]
Turns off all access to external programs.

shell, sh [environment variable SHELL, or /bin/sh]
Select the shell used by the editor.

shellmeta [~{[*?$‘’"\]
Set the meta characters checked to determine if file name expansion is necessary.

shiftwidth, sw [8]
Set the autoindent and shift command indentation width.

showmatch, sm [off]
Vi only. Note matching ‘‘{’’ and ‘‘(’’ for ‘‘}’’ and ‘‘)’’ characters.

showmode, smd [off]
Vi only. Display the current editor mode and a ‘‘modified’’ flag.

sidescroll [16]
Vi only. Set the amount a left-right scroll will shift.

slowopen, slow [off]
Delay display updating during text input. *This option is not yet implemented.*

sourceany [off]
Read startup files not owned by the current user. *This option will never be implemented.*

tabstop, ts [8]
This option sets tab widths for the editor display.

taglength, tl [0]
Set the number of significant characters in tag names.

tags, tag [tags /var/db/libc.tags /sys/kern/tags]
Set the list of tags files.

term, ttytype, tty [environment variable TERM]
Set the terminal type.

terse [off]
This option has historically made editor messages less verbose. It has no effect in this implementation.

tildeop [off]
Modify the ~ command to take an associated motion.

timeout, to [on]
Time out on keys which may be mapped.

ttywerase [off]
Vi only. Select an alternate erase algorithm.

verbose [off]
Vi only. Display an error message for every error.

w300 [no default]
Vi only. Set the window size if the baud rate is less than 1200 baud.

w1200 [no default]
Vi only. Set the window size if the baud rate is equal to 1200 baud.

w9600 [no default]
Vi only. Set the window size if the baud rate is greater than 1200 baud.

warn [on]
Ex only. This option causes a warning message to the terminal if the file has been modified, since it was last written, before a **!** command.

window, w, wi [environment variable LINES]
Set the window size for the screen.

windowname [off]
Change the icon/window name to the current file name even if it can't be restored on editor exit.

wraplen, wl [0]
Vi only. Break lines automatically, the specified number of columns from the left-hand margin. If both the **wraplen** and **wrapmargin** edit options are set, the **wrapmargin** value is used.

wrapmargin, wm [0]
Vi only. Break lines automatically, the specified number of columns from the right-hand margin. If both the **wraplen** and **wrapmargin** edit options are set, the **wrapmargin** value is used.

wrapscan, ws [on]
Set searches to wrap around the end or beginning of the file.

writeany, wa [off]
Turn off file-overwriting checks.

ENVIRONMENT VARIABLES

COLUMNS
The number of columns on the screen. This value overrides any system or terminal specific values. If the *COLUMNS* environmental variable is not set when *ex/vi* runs, or the **columns** option is explicitly reset by the user, *ex/vi* enters the value into the environment.

EXINIT
A list of *ex* startup commands, read if the variable *NEXINIT* is not set.

HOME The user's home directory, used as the initial directory path for the startup ''$*HOME*/.nexrc'' and ''$*HOME*/.exrc'' files. This value is also used as the default directory for the *vi* **cd** command.

LINES The number of rows on the screen. This value overrides any system or terminal specific values. If the *LINES* environmental variable is not set when *ex/vi* runs, or the **lines** option is explicitly reset by the user, *ex/vi* enters the value into the environment.

NEXINIT
A list of *ex* startup commands.

SHELL The user's shell of choice (see also the **shell** option).

TERM The user's terminal type. The default is the type ''unknown''. If the *TERM* environmental variable is not set when *ex/vi* runs, or the **term** option is explicitly reset by the user, *ex/vi* enters the value into the environment.

TMPDIR
The location used to stored temporary files (see also the **directory** edit option).

ASYNCHRONOUS EVENTS

SIGALRM
Vi/ex uses this signal for periodic backups of file modifications and to display ''busy'' messages when operations are likely to take a long time.

SIGHUP

SIGTERM
If the current buffer has changed since it was last written in its entirety, the editor attempts to save the modified file so it can be later recovered. See the *vi/ex* Reference manual section entitled ''Recovery'' for more information.

SIGINT
When an interrupt occurs, the current operation is halted, and the editor returns to the command level. If interrupted during text input, the text already input is resolved into the file as if the text input had been normally terminated.

SIGWINCH
The screen is resized. See the *vi/ex* Reference manual section entitled ''Sizing the Screen'' for more information.

SIGCONT

SIGQUIT

SIGTSTP
Vi/ex ignores these signals.

FILES

/bin/sh The default user shell.

/etc/vi.exrc
System-wide vi startup file.

/tmp Temporary file directory.

/var/tmp/vi.recover
The default recovery file directory.

$HOME/.nexrc
1st choice for user's home directory startup file.

$HOME/.exrc
2nd choice for user's home directory startup file.

.nexrc 1st choice for local directory startup file.

.exrc 2nd choice for local directory startup file.

SEE ALSO

ctags(1), *more*(1), *curses*(3), *dbopen*(3)

The ''Vi Quick Reference'' card.

''An Introduction to Display Editing with Vi'', found in the ''UNIX User's Manual Supplementary Documents'' section of both the 4.3BSD and 4.4BSD manual sets. This document is the closest thing available to an introduction to the *vi* screen editor.

''Ex Reference Manual (Version 3.7)'', found in the ''UNIX User's Manual Supplementary Documents'' section of both the 4.3BSD and 4.4BSD manual sets. This document is the final reference for the *ex* editor, as distributed in most historic 4BSD and System V systems.

''Edit: A tutorial'', found in the ''UNIX User's Manual Supplementary Documents'' section of the 4.3BSD manual set. This document is an introduction to a simple version of the *ex* screen editor.

''Ex/Vi Reference Manual'', found in the ''UNIX User's Manual Supplementary Documents'' section of the 4.4BSD manual set. This document is the final reference for the *nex/nvi* text editors, as distributed in 4.4BSD and 4.4BSD-Lite.

Roff source for all of these documents is distributed with *nex/nvi* in the *nvi/USD.doc* directory of the *nex/nvi* source code.

The files ''autowrite'', ''input'', ''quoting'' and ''structures'' found in the *nvi/docs/internals* directory of the *nex/nvi* source code.

HISTORY

The *nex/nvi* replacements for the *ex/vi* editor first appeared in 4.4BSD.

STANDARDS

Nex/nvi is close to IEEE Std1003.2 (''POSIX''). That document differs from historical *ex/vi* practice in several places; there are changes to be made on both sides.

NAME

ex, vi, view − text editors

SYNOPSIS

ex [**-eFGRrSsv**] [**-c** *cmd*] [**-t** *tag*] [**-w** *size*] [file ...]
vi [**-eFGlRrSv**] [**-c** *cmd*] [**-t** *tag*] [**-w** *size*] [file ...]
view [**-eFGRrSv**] [**-c** *cmd*] [**-t** *tag*] [**-w** *size*] [file ...]

LICENSE

The vi program is freely redistributable. You are welcome to copy, modify and share it with others under the conditions listed in the LICENSE file. If any company (not individual!) finds vi sufficiently useful that you would have purchased it, or if any company wishes to redistribute it, contributions to the authors would be appreciated.

DESCRIPTION

Vi is a screen oriented text editor. *Ex* is a line-oriented text editor. *Ex* and *vi* are different interfaces to the same program, and it is possible to switch back and forth during an edit session. *View* is the equivalent of using the **−R** (read-only) option of *vi*.

This manual page is the one provided with the *nex/nvi* versions of the *ex/vi* text editors. *Nex/nvi* are intended as bug-for-bug compatible replacements for the original Fourth Berkeley Software Distribution (4BSD) *ex* and *vi* programs. For the rest of this manual page, *nex/nvi* is used only when it's necessary to distinguish it from the historic implementations of *ex/vi*.

This manual page is intended for users already familiar with *ex/vi*. Anyone else should almost certainly read a good tutorial on the editor before this manual page. If you're in an unfamiliar environment, and you absolutely have to get work done immediately, read the section after the options description, entitled ''Fast Startup''. It's probably enough to get you going.

The following options are available:

−c Execute **cmd** on the first file loaded. Particularly useful for initial positioning in the file, however **cmd** is not limited to positioning commands. This is the POSIX 1003.2 interface for the historic ''+cmd'' syntax. *Nex/nvi* supports both the old and new syntax.

−e Start editing in ex mode, as if the command name were *ex*.

−F Don't copy the entire file when first starting to edit. (The default is to make a copy in case someone else modifies the file during your edit session.)

−G Start editing in gtags mode, as if the gtagsmode option were set.

−l Start editing with the lisp and showmatch options set.

−R Start editing in read-only mode, as if the command name was *view*, or the **readonly** option was set.

−r Recover the specified files, or, if no files are specified, list the files that could be recovered. If no recoverable files by the specified name exist, the file is edited as if the **−r** option had not been specified. (Also see *virecover*(8) for more information.)

−S Run with the **secure** edit option set, disallowing all access to external programs.

−s Enter batch mode; applicable only to *ex* edit sessions. Batch mode is useful when running *ex* scripts. Prompts, informative messages and other user oriented message are turned off, and no startup files or environmental variables are read. This is the POSIX 1003.2 interface for the historic ''−'' argument. *Nex/nvi* supports both the old and new syntax.

−t Start editing at the specified tag. (See *ctags*(1)).

−w Set the initial window size to the specified number of lines.

−v Start editing in vi mode, as if the command name was *vi* or *view*.

Command input for *ex/vi* is read from the standard input. In the *vi* interface, it is an error if standard input is not a terminal. In the *ex* interface, if standard input is not a terminal, *ex* will read commands from it

regardless, however, the session will be a batch mode session, exactly as if the **−s** option had been specified.

Ex/vi exits 0 on success, and greater than 0 if an error occurs.

FAST STARTUP

This section will tell you the minimum amount that you need to do simple editing tasks using *vi*. If you've never used any screen editor before, you're likely to have problems even with this simple introduction. In that case you should find someone that already knows *vi* and have them walk you through this section.

Vi is a screen editor. This means that it takes up almost the entire screen, displaying part of the file on each screen line, except for the last line of the screen. The last line of the screen is used for you to give commands to *vi*, and for *vi* to give information to you.

The other fact that you need to understand is that *vi* is a modeful editor, i.e. you are either entering text or you are executing commands, and you have to be in the right mode to do one or the other. You will be in command mode when you first start editing a file. There are commands that switch you into input mode. There is only one key that takes you out of input mode, and that is the <escape> key. (Key names are written using less-than and greater-than signs, e.g. <escape> means the ''escape'' key, usually labeled ''esc'' on your terminal's keyboard.) If you're ever confused as to which mode you're in, keep entering the <escape> key until *vi* beeps at you. (Generally, *vi* will beep at you if you try and do something that's not allowed. It will also display error messages.)

To start editing a file, enter the command ''vi file_name<carriage-return>''. The command you should enter as soon as you start editing is '':set verbose showmode<carriage-return>''. This will make the editor give you verbose error messages and display the current mode at the bottom of the screen.

The commands to move around the file are:

h Move the cursor left one character.

j Move the cursor down one line.

k Move the cursor up one line.

l Move the cursor right one character.

<cursor-arrows>
The cursor arrow keys should work, too.

/text<carriage-return>
Search for the string ''text'' in the file, and move the cursor to its first character.

The commands to enter new text are:

a Append new text, *after* the cursor.

i Insert new text, *before* the cursor.

o Open a new line below the line the cursor is on, and start entering text.

O Open a new line above the line the cursor is on, and start entering text.

<escape>
Once you've entered input mode using the one of the **a**, **i**, **O** or **o** commands, use **<escape>** to quit entering text and return to command mode.

The commands to copy text are:

yy Copy the line the cursor is on.

p Append the copied line after the line the cursor is on.

The commands to delete text are:

dd Delete the line the cursor is on.

x Delete the character the cursor is on.

The commands to write the file are:

:w<carriage-return>
Write the file back to the file with the name that you originally used as an argument on the *vi* command line.

:w file_name<carriage-return>
Write the file back to the file with the name ‘‘file_name’’.

The commands to quit editing and exit the editor are:

:q<carriage-return>
Quit editing and leave vi (if you’ve modified the file, but not saved your changes, *vi* will refuse to quit).

:q!<carriage-return>
Quit, discarding any modifications that you may have made.

One final caution. Unusual characters can take up more than one column on the screen, and long lines can take up more than a single screen line. The above commands work on ‘‘physical’’ characters and lines, i.e. they affect the entire line no matter how many screen lines it takes up and the entire character no matter how many screen columns it takes up.

VI COMMANDS

The following section describes the commands available in the command mode of the *vi* editor. In each entry below, the tag line is a usage synopsis for the command character.

[count] <control-A>
Search forward *count* times for the current word.

[count] <control-B>
Page backwards *count* screens.

[count] <control-D>
Scroll forward *count* lines.

[count] <control-E>
Scroll forward *count* lines, leaving the current line and column as is, if possible.

[count] <control-F>
Page forward *count* screens.

<control-G>
Display the file information.

<control-H>

[count] h
Move the cursor back *count* characters in the current line.

[count] <control-J>

[count] <control-N>

[count] j
Move the cursor down *count* lines without changing the current column.

<control-L>

<control-R>
Repaint the screen.

[count] <control-M>

[count] +
Move the cursor down *count* lines to the first nonblank character of that line.

[count] <control-P>

[count] k
Move the cursor up *count* lines, without changing the current column.

<control-T>
Return to the most recent tag context.

<control-U>
Scroll backwards *count* lines.

<control-W>
Switch to the next lower screen in the window, or, to the first screen if there are no lower screens in the window.

<control-Y>
Scroll backwards *count* lines, leaving the current line and column as is, if possible.

<control-Z>
Suspend the current editor session.

<escape>
Execute *ex* commands or cancel partial commands.

<control-]>
Push a tag reference onto the tag stack. In gtags mode, if at the first column of line, locate function references otherwise function definitions.

<control-^>
Switch to the most recently edited file.

[count] <space>

[count] l
Move the cursor forward *count* characters without changing the current line.

[count] ! motion shell-argument(s)
Replace text with results from a shell command.

[count] # #|+|-
Increment or decrement the cursor number.

[count] $
Move the cursor to the end of a line.

% Move to the matching character.

& Repeat the previous substitution command on the current line.

'<character>

`<character>
Return to a context marked by the character <*character*>.

[count] (
Back up *count* sentences.

[count])
Move forward *count* sentences.

[count] ,
Reverse find character *count* times.

[count] -
Move to first nonblank of the previous line, *count* times.

[count] .
Repeat the last *vi* command that modified text.

/RE<carriage-return>

/RE/ [offset]<carriage-return>

?RE<carriage-return>

?RE? [offset]<carriage-return>

N

n Search forward or backward for a regular expression.

0 Move to the first character in the current line.

: Execute an ex command.

[count] ;
Repeat the last character find *count* times.

[count] < motion

[count] > motion
Shift lines left or right.

@ buffer
Execute a named buffer.

[count] A
Enter input mode, appending the text after the end of the line.

[count] B
Move backwards *count* bigwords.

[buffer] [count] C
Change text from the current position to the end-of-line.

[buffer] D
Delete text from the current position to the end-of-line.

[count] E
Move forward *count* end-of-bigwords.

[count] F <character>
Search *count* times backward through the current line for <*character*>.

[count] G
Move to line *count*, or the last line of the file if *count* not specified.

[count] H
Move to the screen line *count* - *1* lines below the top of the screen.

[count] I
Enter input mode, inserting the text at the beginning of the line.

[count] J
Join lines.

[count] L
Move to the screen line *count* - *1* lines above the bottom of the screen.

M Move to the screen line in the middle of the screen.

[count] O
Enter input mode, appending text in a new line above the current line.

[buffer] P
Insert text from a buffer.

Q Exit *vi* (or visual) mode and switch to *ex* mode.

[count] R
Enter input mode, replacing the characters in the current line.

[buffer] [count] S
Substitute *count* lines.

[count] T <character>
Search backwards, *count* times, through the current line for the character *after* the specified <*character*>.

U Restore the current line to its state before the cursor last moved to it.

[count] W
Move forward *count* bigwords.

[buffer] [count] X
Delete *count* characters before the cursor.

[buffer] [count] Y
Copy (or ‘‘yank’’) *count* lines into the specified buffer.

ZZ Write the file and exit *vi*.

[count] [[
Back up *count* section boundaries.

[count]]]
Move forward *count* section boundaries.

^ Move to first nonblank character on the current line.

[count] _
Move down *count - 1* lines, to the first nonblank character.

[count] a
Enter input mode, appending the text after the cursor.

[count] b
Move backwards *count* words.

[buffer] [count] c motion
Change a region of text.

[buffer] [count] d motion
Delete a region of text.

[count] e
Move forward *count* end-of-words.

[count] f<character>
Search forward, *count* times, through the rest of the current line for <*character*>.

[count] i
Enter input mode, inserting the text before the cursor.

m <character>
Save the current context (line and column) as <*character*>.

[count] o
Enter input mode, appending text in a new line under the current line.

[buffer] p
Append text from a buffer.

[count] r <character>
Replace *count* characters.

[buffer] [count] s
Substitute *count* characters in the current line starting with the current character.

[count] t <character>
Search forward, *count* times, through the current line for the character immediately *before* <*character*>.

u Undo the last change made to the file.

[count] w
Move forward *count* words.

[buffer] [count] x
Delete *count* characters.

[buffer] [count] y motion
Copy (or ‘‘yank’’) a text region specified by the *count* and motion into a buffer.

[count1] z [count2] -|.|+|^|<carriage-return>
Redraw, optionally repositioning and resizing the screen.

[count] {
Move backward *count* paragraphs.

[count] |
Move to a specific *column* position on the current line.

[count] }
Move forward *count* paragraphs.

[count] ~
Reverse the case of the next *count* character(s).

[count] ~ motion
Reverse the case of the characters in a text region specified by the *count* and *motion*.

<interrupt>
Interrupt the current operation.

VI TEXT INPUT COMMANDS

The following section describes the commands available in the text input mode of the *vi* editor.

<nul> Replay the previous input.

<control-D>
Erase to the previous **shiftwidth** column boundary.

^<control-D>
Erase all of the autoindent characters, and reset the autoindent level.

0<control-D>
Erase all of the autoindent characters.

<control-T>
Insert sufficient <*tab*> and <*space*> characters to move forward to the next **shiftwidth** column boundary.

<erase>

<control-H>
Erase the last character.

<literal next>
Quote the next character.

<escape>
Resolve all text input into the file, and return to command mode.

<line erase>
Erase the current line.

<control-W>

<word erase>
Erase the last word. The definition of word is dependent on the **altwerase** and **ttywerase** options.

<control-X>[0-9A-Fa-f]+
Insert a character with the specified hexadecimal value into the text.

<interrupt>
Interrupt text input mode, returning to command mode.

EX COMMANDS

The following section describes the commands available in the *ex* editor. In each entry below, the tag line is a usage synopsis for the command.

<end-of-file>
Scroll the screen.

! argument(s)

[range]! argument(s)
Execute a shell command, or filter lines through a shell command.

" A comment.

[range] nu[mber] [count] [flags]

[range] # [count] [flags]
Display the selected lines, each preceded with its line number.

@ buffer

*** buffer**
Execute a buffer.

[line] a[ppend][!]
The input text is appended after the specified line.

[range] c[hange][!] [count]
The input text replaces the specified range.

cs[cope] add | find | help | kill | reset
Execute a Cscope command.

[range] d[elete] [buffer] [count] [flags]
Delete the lines from the file.

di[splay] b[uffers] | c[onnections] | s[creens] | t[ags]
Display buffers, Cscope connections, screens or tags.

[Ee][dit][!] [+cmd] [file]

[Ee]x[!] [+cmd] [file]
Edit a different file.

exu[sage] [command]
Display usage for an *ex* command.

f[ile] [file]
Display and optionally change the file name.

[Ff]g [name]
Vi mode only. Foreground the specified screen.

[range] g[lobal] /pattern/ [commands]

[range] v /pattern/ [commands]
Apply commands to lines matching (or not matching) a pattern.

he[lp] Display a help message.

[line] i[nsert][!]
The input text is inserted before the specified line.

[range] j[oin][!] [count] [flags]
Join lines of text together.

[range] l[ist] [count] [flags]
Display the lines unambiguously.

map[!] [lhs rhs]
Define or display maps (for *vi* only).

[line] ma[rk] <character>

[line] k <character>
Mark the line with the mark <*character*>.

[range] m[ove] line
Move the specified lines after the target line.

mk[exrc][!] file
Write the abbreviations, editor options and maps to the specified file.

[Nn][ext][!] [file ...]
Edit the next file from the argument list.

[line] o[pen] /pattern/ [flags]
Enter open mode.

pre[serve]
Save the file in a form that can later be recovered using the *ex* **−r** option.

[Pp]rev[ious][!]
Edit the previous file from the argument list.

[range] p[rint] [count] [flags]
Display the specified lines.

[line] pu[t] [buffer]
Append buffer contents to the current line.

q[uit][!]
End the editing session.

[line] r[ead][!] [file]
Read a file.

rec[over] file
Recover *file* if it was previously saved.

res[ize] [+|-]size
Vi mode only. Grow or shrink the current screen.

rew[ind][!]
Rewind the argument list.

rta[g][!] tagstring
Edit the file refering to the specified tag. (only in gtags mode)

se[t] [option[=[value]] ...] [nooption ...] [option? ...] [all]
Display or set editor options.

sh[ell] Run a shell program.

so[urce] file
Read and execute *ex* commands from a file.

[range] s[ubstitute] [/pattern/replace/] [options] [count] [flags]

[range] & [options] [count] [flags]

[range] ~ [options] [count] [flags]
Make substitutions.

su[spend][!]

st[op][!]

<suspend>
Suspend the edit session.

[Tt]a[g][!] tagstring
Edit the file containing the specified tag.

tagn[ext][!]
Edit the file containing the next context for the current tag.

tagp[op][!] [file | number]
Pop to the specified tag in the tags stack.

tagp[rev][!]
Edit the file containing the previous context for the current tag.

unm[ap][!] lhs
Unmap a mapped string.

ve[rsion]
Display the version of the *ex/vi* editor.

[line] vi[sual] [type] [count] [flags]
Ex mode only. Enter *vi*.

[Vi]i[sual][!] [+cmd] [file]
Vi mode only. Edit a new file.

viu[sage] [command]
Display usage for a *vi* command.

[range] w[rite][!] [>>] [file]

[range] w[rite] [!] [file]

[range] wn[!] [>>] [file]

[range] wq[!] [>>] [file]
Write the file.

[range] x[it][!] [file]
Write the file if it has been modified.

[range] ya[nk] [buffer] [count]
Copy the specified lines to a buffer.

[line] z [type] [count] [flags]
Adjust the window.

SET OPTIONS

There are a large number of options that may be set (or unset) to change the editor's behavior. This section describes the options, their abbreviations and their default values.

In each entry below, the first part of the tag line is the full name of the option, followed by any equivalent abbreviations. The part in square brackets is the default value of the option. Most of the options are boolean, i.e. they are either on or off, and do not have an associated value.

Options apply to both *ex* and *vi* modes, unless otherwise specified.

altwerase [off]
Vi only. Select an alternative word erase algorithm.

autoindent, ai [off]
Automatically indent new lines.

autoprint, ap [off]
Ex only. Display the current line automatically.

autowrite, aw [off]
Write modified files automatically when changing files.

backup [""]
Backup files before they are overwritten.

beautify, bf [off]
Discard control characters.

cdpath [environment variable CDPATH, or current directory]
The directory paths used as path prefixes for the **cd** command.

cedit [no default]
Set the character to edit the colon command-line history.

columns, co [80]
Set the number of columns in the screen.

comment [off]
Vi only. Skip leading comments in shell, C and C++ language files.

directory, dir [environment variable TMPDIR, or /tmp]
The directory where temporary files are created.

edcompatible, ed [off]
Remember the values of the ''c'' and ''g'' suffixes to the **substitute** commands, instead of initializing them as unset for each new command.

errorbells, eb [off]
Ex only. Announce error messages with a bell.

escapetime [1]
The 10th's of a second *ex/vi* waits for a subsequent key to complete an <escape> key mapping. Over a link with high latency, the default may cause arrow and function keys to introduce artifacts. If that's the case, try increasing this to a value between 3 and 6.

exrc, ex [off]
Read the startup files in the local directory.

extended [off]
Regular expressions are extended (i.e. *egrep*(1)−style) expressions.

filec [no default]
Set the character to perform file path completion on the colon command line.

flash [on]
Flash the screen instead of beeping the keyboard on error.

gtagsmode, gt [off]
Use GTAGS and GRTAGS instead of tags.

hardtabs, ht [8]
Set the spacing between hardware tab settings.

iclower [off]
Makes all Regular Expressions case-insensitive, as long as an upper-case letter does not appear in the search string.

ignorecase, ic [off]
Ignore case differences in regular expressions.

keytime [6]
The 10th's of a second *ex/vi* waits for a subsequent key to complete a key mapping.

leftright [off]
Vi only. Do left-right scrolling.

lines, li [24]
Vi only. Set the number of lines in the screen.

lisp [off]
Vi only. Modify various search commands and options to work with Lisp. *This option is not yet implemented.*

list [off]
Display lines in an unambiguous fashion.

lock [on]
Attempt to get an exclusive lock on any file being edited, read or written.

magic [on]
Treat certain characters specially in regular expressions.

matchchars [[]{}()<>]
Character pairs looked for by the % command.

matchtime [7]
Vi only. The 10th's of a second *ex/vi* pauses on the matching character when the **showmatch** option is set.

mesg [on]
Permit messages from other users.

modelines, modeline [off]
Read the first and last few lines of each file for *ex* commands. *This option will never be implemented.*

msgcat [./]
This option selects a message catalog to be used to display error and informational messages in a specified language. If the value of this option ends with a '/', it is treated as the name of a directory that contains a message catalog

noprint [""]
Characters that are never handled as printable characters.

number, nu [off]
Precede each line displayed with its current line number.

octal [off]
Display unknown characters as octal numbers, instead of the default hexadecimal.

open [on]
Ex only. If this option is not set, the **open** and **visual** commands are disallowed.

optimize, opt [on]
Vi only. Optimize text throughput to dumb terminals. *This option is not yet implemented.*

paragraphs, para [IPLPPPQPP LIpplpipbp]
Vi only. Define additional paragraph boundaries for the **{** and **}** commands.

path [] Define additional directories to search for files being edited.

print [""]
Characters that are always handled as printable characters.

prompt [on]
Ex only. Display a command prompt.

readonly, ro [off]
Mark the file and session as read-only.

recdir [/var/tmp/vi.recover]
The directory where recovery files are stored.

redraw, re [off]
Vi only. Simulate an intelligent terminal on a dumb one. *This option is not yet implemented.*

remap [on]
Remap keys until resolved.

report [5]
Set the number of lines about which the editor reports changes or yanks.

ruler [off]
Vi only. Display a row/column ruler on the colon command line.

scroll, scr [window / 2]
Set the number of lines scrolled.

searchincr [off]
Makes the **/** and **?** commands incremental.

sections, sect [NHSHH HUnhsh]
Vi only. Define additional section boundaries for the **[[** and **]]** commands.

secure [off]
Turns off all access to external programs.

shell, sh [environment variable SHELL, or /bin/sh]
Select the shell used by the editor.

shellmeta [~{[*?$‘’"\]
Set the meta characters checked to determine if file name expansion is necessary.

shiftwidth, sw [8]
Set the autoindent and shift command indentation width.

showmatch, sm [off]
Vi only. Note matching ‘‘{’’ and ‘‘(’’ for ‘‘}’’ and ‘‘)’’ characters.

showmode, smd [off]
Vi only. Display the current editor mode and a ‘‘modified’’ flag.

sidescroll [16]
Vi only. Set the amount a left-right scroll will shift.

slowopen, slow [off]
Delay display updating during text input. *This option is not yet implemented.*

sourceany [off]
Read startup files not owned by the current user. *This option will never be implemented.*

tabstop, ts [8]
This option sets tab widths for the editor display.

taglength, tl [0]
Set the number of significant characters in tag names.

tags, tag [tags /var/db/libc.tags /sys/kern/tags]
Set the list of tags files.

term, ttytype, tty [environment variable TERM]
Set the terminal type.

terse [off]
This option has historically made editor messages less verbose. It has no effect in this implementation.

tildeop [off]
Modify the ~ command to take an associated motion.

timeout, to [on]
Time out on keys which may be mapped.

ttywerase [off]
Vi only. Select an alternative erase algorithm.

verbose [off]
Vi only. Display an error message for every error.

w300 [no default]
Vi only. Set the window size if the baud rate is less than 1200 baud.

w1200 [no default]
Vi only. Set the window size if the baud rate is equal to 1200 baud.

w9600 [no default]
Vi only. Set the window size if the baud rate is greater than 1200 baud.

warn [on]
Ex only. This option causes a warning message to the terminal if the file has been modified, since it was last written, before a **!** command.

window, w, wi [environment variable LINES]
Set the window size for the screen.

windowname [off]
Change the icon/window name to the current file name even if it can’t be restored on editor exit.

wraplen, wl [0]
Vi only. Break lines automatically, the specified number of columns from the left-hand margin. If both the **wraplen** and **wrapmargin** edit options are set, the **wrapmargin** value is used.

wrapmargin, wm [0]
Vi only. Break lines automatically, the specified number of columns from the right-hand margin. If both the **wraplen** and **wrapmargin** edit options are set, the **wrapmargin** value is used.

wrapscan, ws [on]
Set searches to wrap around the end or beginning of the file.

writeany, wa [off]
Turn off file-overwriting checks.

ENVIRONMENTAL VARIABLES

COLUMNS
The number of columns on the screen. This value overrides any system or terminal specific values. If the *COLUMNS* environmental variable is not set when *ex/vi* runs, or the **columns** option is explicitly reset by the user, *ex/vi* enters the value into the environment.

EXINIT
A list of *ex* startup commands, read if the variable *NEXINIT* is not set.

HOME The user's home directory, used as the initial directory path for the startup ''$*HOME*/.nexrc'' and ''$*HOME*/.exrc'' files. This value is also used as the default directory for the *vi* **cd** command.

LINES The number of rows on the screen. This value overrides any system or terminal specific values. If the *LINES* environmental variable is not set when *ex/vi* runs, or the **lines** option is explicitly reset by the user, *ex/vi* enters the value into the environment.

NEXINIT
A list of *ex* startup commands.

SHELL The user's shell of choice (see also the **shell** option).

TERM The user's terminal type. The default is the type ''unknown''. If the *TERM* environmental variable is not set when *ex/vi* runs, or the **term** option is explicitly reset by the user, *ex/vi* enters the value into the environment.

TMPDIR
The location used to stored temporary files (see also the **directory** edit option).

ASYNCHRONOUS EVENTS

SIGALRM
Vi/ex uses this signal for periodic backups of file modifications and to display ''busy'' messages when operations are likely to take a long time.

SIGHUP

SIGTERM
If the current buffer has changed since it was last written in its entirety, the editor attempts to save the modified file so it can be later recovered. See the *vi/ex* Reference manual section entitled ''Recovery'' for more information.

SIGINT
When an interrupt occurs, the current operation is halted, and the editor returns to the command level. If interrupted during text input, the text already input is resolved into the file as if the text input had been normally terminated.

SIGWINCH
The screen is resized. See the *vi/ex* Reference manual section entitled ''Sizing the Screen'' for more information.

SIGCONT

SIGQUIT

SIGTSTP
Vi/ex ignores these signals.

FILES

/bin/sh The default user shell.

/etc/vi.exrc
System-wide vi startup file.

/tmp Temporary file directory.

/var/tmp/vi.recover
The default recovery file directory.

$HOME/.nexrc
1st choice for user's home directory startup file.

$HOME/.exrc
2nd choice for user's home directory startup file.

.nexrc 1st choice for local directory startup file.

.exrc 2nd choice for local directory startup file.

SEE ALSO

ctags(1), *more*(3), *curses*(3), *dbopen*(3), *virecover*(8)

The ''Vi Quick Reference'' card.

''An Introduction to Display Editing with Vi'', found in the ''UNIX User's Manual Supplementary Documents'' section of both the 4.3BSD and 4.4BSD manual sets. This document is the closest thing available to an introduction to the *vi* screen editor.

''Ex Reference Manual (Version 3.7)'', found in the ''UNIX User's Manual Supplementary Documents'' section of both the 4.3BSD and 4.4BSD manual sets. This document is the final reference for the *ex* editor, as distributed in most historic 4BSD and System V systems.

''Edit: A tutorial'', found in the ''UNIX User's Manual Supplementary Documents'' section of the 4.3BSD manual set. This document is an introduction to a simple version of the *ex* screen editor.

''Ex/Vi Reference Manual'', found in the ''UNIX User's Manual Supplementary Documents'' section of the 4.4BSD manual set. This document is the final reference for the *nex/nvi* text editors, as distributed in 4.4BSD and 4.4BSD-Lite.

Roff source for all of these documents is distributed with *nex/nvi* in the *nvi/USD.doc* directory of the *nex/nvi* source code.

The files ''autowrite'', ''input'', ''quoting'' and ''structures'' found in the *nvi/docs/internals* directory of the *nex/nvi* source code.

HISTORY

The *nex/nvi* replacements for the *ex/vi* editor first appeared in 4.4BSD.

STANDARDS

Nex/nvi is close to IEEE Std1003.2 (''POSIX''). That document differs from historical *ex/vi* practice in several places; there are changes to be made on both sides.

NAME

vis — display non-printable characters in a visual format

SYNOPSIS

vis [**-bcfhlnostw**] [**-e** *extra*] [**-F** *foldwidth*] [*file ...*]

DESCRIPTION

vis is a filter for converting non-printable characters into a visual representation. It differs from `cat -v` in that the form is unique and invertible. By default, all non-graphic characters except space, tab, and newline are encoded. A detailed description of the various visual formats is given in `vis`(3).

The options are as follows:

-b Turns off prepending of backslash before up-arrow control sequences and meta characters, and disables the doubling of backslashes. This produces output which is neither invertible or precise, but does represent a minimum of change to the input. It is similar to "`cat -v`". (`VIS_NOSLASH`)

-c Request a format which displays a small subset of the non-printable characters using C-style backslash sequences. (`VIS_CSTYLE`)

-e *extra*
Also encode characters in *extra*, per `svis`(3).

-F *foldwidth*
Causes **vis** to fold output lines to foldwidth columns (default 80), like `fold`(1), except that a hidden newline sequence is used, (which is removed when inverting the file back to its original form with `unvis`(1)). If the last character in the encoded file does not end in a newline, a hidden newline sequence is appended to the output. This makes the output usable with various editors and other utilities which typically don't work with partial lines.

-f Same as **-F**.

-h Encode using the URI encoding from RFC 1808. (`VIS_HTTPSTYLE`)

-l Mark newlines with the visible sequence '`\$`', followed by the newline.

-n Turns off any encoding, except for the fact that backslashes are still doubled and hidden newline sequences inserted if **-f** or **-F** is selected. When combined with the **-f** flag, **vis** becomes like an invertible version of the `fold`(1) utility. That is, the output can be unfolded by running the output through `unvis`(1).

-o Request a format which displays non-printable characters as an octal number, \ddd. (`VIS_OCTAL`)

-s Only characters considered unsafe to send to a terminal are encoded. This flag allows backspace, bell, and carriage return in addition to the default space, tab and newline. (`VIS_SAFE`)

-t Tabs are also encoded. (`VIS_TAB`)

-w White space (space-tab-newline) is also encoded. (`VIS_WHITE`)

SEE ALSO

`unvis`(1), `svis`(3), `vis`(3)

HISTORY

The **vis** command appears in 4.4BSD.

NAME

vmstat — report virtual memory statistics

SYNOPSIS

vmstat [**-CefHiLlmstUvW**] [**-c** *count*] [**-h** *hashname*] [**-M** *core*] [**-N** *system*] [**-u** *histname*] [**-w** *wait*] [*disks*]

DESCRIPTION

vmstat reports certain kernel statistics kept about process, virtual memory, disk, trap, and CPU activity.

The options are as follows:

-C Report on kernel memory caches. Combine with the **-m** option to see information about memory pools that back the caches.

-c *count* Repeat the display *count* times. The first display is for the time since a reboot and each subsequent report is for the time period since the last display. If no *wait* interval is specified, the default is 1 second.

-e Report the values of system event counters.

-f Report fork statistics.

-H Report all hash table statistics.

-h *hashname*
Report hash table statistics for *hashname*.

-i Report the values of system interrupt counters.

-L List all the hashes supported for **-h** and **-H**.

-l List the UVM histories being maintained by the kernel.

-M *core* Extract values associated with the name list from the specified core instead of the default `/dev/mem`.

-m Report on the usage of kernel dynamic memory listed first by size of allocation and then by type of usage, followed by a list of the kernel memory pools and their usage.

-N *system* Extract the name list from the specified system instead of the default `/netbsd`.

-s Display the contents of the `uvmexp` structure. This contains various paging event and memory status counters.

-t Display the contents of the `vmtotal` structure. This includes information about processes and virtual memory.

The process part shows the number of processes in the following states:

ru	on the run queue
dw	in disk I/O wait
pw	waiting for paging
sl	sleeping while resident
sw	swapped out, but either runnable or in short-term blocking state

The virtual memory section shows:

total-v Total virtual memory

active-v	Active virtual memory in use
active-r	Active real memory in use
vm-sh	Shared virtual memory
avm-sh	Active shared virtual memory
rm-sh	Shared real memory
arm-sh	Active shared real memory
free	Free memory

All memory values are shown in number of pages.

-U Dump all UVM histories.

-u *histname*
Dump the specified UVM history.

-v Print more verbose information. When used with the **-i**, **-e**, or **-m** options prints out all counters, not just those with non-zero values.

-W Print more verbose information about kernel memory pools.

-w *wait* Pause *wait* seconds between each display. If no repeat *count* is specified, the default is infinity.

By default, **vmstat** displays the following information:

procs Information about the numbers of processes in various states.

r	in run queue
b	blocked for resources (i/o, paging, etc.)
w	runnable or short sleeper (< 20 secs) but swapped

memory Information about the usage of virtual and real memory. Virtual pages (reported in units of 1024 bytes) are considered active if they belong to processes which are running or have run in the last 20 seconds.

avm	active virtual pages
fre	size of the free list

page Information about page faults and paging activity. These are averaged every five seconds, and given in units per second.

flt	total page faults
re	page reclaims (simulating reference bits)
pi	pages paged in
po	pages paged out
fr	pages freed per second
sr	pages scanned by clock algorithm, per-second

disks Disk transfers per second. Typically paging will be split across the available drives. The header of the field is the first character of the disk name and the unit number. If more than four disk drives are configured in the system, **vmstat** displays only the first four drives. To force **vmstat** to display specific drives, their names may be supplied on the command line.

faults Trap/interrupt rate averages per second over last 5 seconds.

in	device interrupts per interval (including clock interrupts)
sy	system calls per interval

cs CPU context switch rate (switches/interval)

cpu Breakdown of percentage usage of CPU time.

us user time for normal and low priority processes
sy system time
id CPU idle

FILES

`/netbsd` default kernel namelist
`/dev/mem` default memory file

EXAMPLES

The command "`vmstat -w 5`" will print what the system is doing every five seconds; this is a good choice of printing interval since this is how often some of the statistics are sampled in the system. Others vary every second and running the output for a while will make it apparent which are recomputed every second.

SEE ALSO

`fstat`(1), `netstat`(1), `nfsstat`(1), `ps`(1), `systat`(1), `iostat`(8), `pstat`(8)

The sections starting with "Interpreting system activity" in *Installing and Operating 4.3BSD*.

BUGS

The **`-c`** and **`-w`** options are only available with the default output.

The **`-l`**, **`-U`**, and **`-u`** options are useful only if the system was compiled with support for UVM history.

NAME

vndcompress, **vnduncompress** — compress/uncompress file system images to/from cloop2 format

SYNOPSIS

vndcompress [**-cd**] *disk/fs-image compressed-image* [*blocksize*]
vnduncompress [**-cd**] *compressed-image disk/fs-image*

DESCRIPTION

The **vndcompress** program compresses an existing file system image into a cloop2 compatible compressed file system image. An optional blocksize can be given. If omitted, the default of 64kB is used.

The **vnduncompress** command decompress a cloop2-compressed file system image back into a regular image.

The file system images that can be handled are not limited to any specific file system, i.e. it is possible to handle images e.g. in ISO 9660 or UFS/FFS format. File system images in the cloop2 format are intended to be used with the vnd(4) driver in compressed mode as configured by the **-z** option of the vnconfig(8) program, and later mounted with the appropriate **-t** option to mount(8).

OPTIONS

The following options are available:

-c Always compress, even if the program was started as **vnduncompress**.

-d Always uncompress (decompress), even if the program was started as **vndcompress**.

EXIT STATUS

The **vndcompress** and **vnduncompress** utilities exit with one of the following values:

0 The operation was performed successfully.

1 An error occurred.

EXAMPLES

To compress an existing CD-ROM file system image, run the following commands:

```
# vndcompress netbsd.iso netbsd.izo
```

Note that the resulting compressed image cannot be mounted directly via NetBSD's vnd(4) and mount_cd9660(8) commands any longer. Instead, you will have to use the **-z** option of vnconfig(8).

The following example decompresses an existing CD-ROM file system image that was compressed in the cloop2 format into a regular file that can then be mounted:

```
# vnconfig vnd0 KNOPPIX.iso
# mount -t cd9660 -o ro /dev/vnd0d /mnt
# vnduncompress /mnt/KNOPPIX/KNOPPIX /var/tmp/knoppix.iso
# umount /mnt
# vnconfig -u vnd0
#
# vnconfig vnd1 /var/tmp/knoppix.iso
# mount -t cd9660 -o ro /dev/vnd1d /mnt
# ls /mnt
.rr_moved cdrom     floppy    lib       opt       sbin      usr
bin       dev       home      mnt       proc      sys       var
boot      etc       initrd    none      root      tmp       vmlinuz
# umount /mnt
```

```
# vnconfig -u vnd1
```

As an alternative, if your vnd(4) was compiled with VND_COMPRESSION, you can use vnconfig(8) to access the cloop-compressed image directly, e.g.,

```
# vnconfig vnd0 KNOPPIX.iso
# mount -t cd9660 -o ro /dev/vnd0d /mnt
# vnconfig -z vnd1 /mnt/KNOPPIX/KNOPPIX
# mount -t cd9660 -o ro /dev/vnd1d /mnt2
# ls /mnt2
.rr_moved cdrom     floppy    lib       opt       sbin      usr
bin       dev       home      mnt       proc      sys       var
boot      etc       initrd    none      root      tmp       vmlinuz
# df /mnt /mnt2
Filesystem    Size      Used     Avail Capacity  Mounted on
/dev/vnd0a    692M     692M        0B   100%    /mnt
/dev/vnd1a    1.9G     1.9G        0B   100%    /mnt2
# umount /mnt2
# vnconfig -u vnd1
# umount /mnt
# vnconfig -u vnd0
```

Note how the 1.9GB big filesystem on /mnt2 is mounted from the compressed file stored on the 692MB CD mounted on /mnt. To create a compressed file system image of an existing directory and mount it, run:

```
# makefs -t ffs include.fs /usr/include
# vndcompress include.fs include.fs.cloop2
# vnconfig -z vnd0 include.fs.cloop2
# mount -o ro /dev/vnd0a /mnt
# ls /mnt
```

To undo the steps, run:

```
# umount /mnt
# vnconfig -u vnd0
# rm /tmp/include.fs.cloop2
# rm /tmp/include.fs
```

SEE ALSO

gzip(1), vnd(4), mount(8), mount_cd9660(8), vnconfig(8)

AUTHORS

The **vndcompress** utility was written by Florian Stoehr ⟨netbsd@wolfnode.de⟩. The **vndcompress** manual page was written by Florian Stoehr ⟨netbsd@wolfnode.de⟩ and Hubert Feyrer ⟨hubertf@NetBSD.org⟩.

NAME

w — who present users are and what they are doing

SYNOPSIS

w [**-hinw**] [**-M** *core*] [**-N** *system*] [*user*]

DESCRIPTION

The **w** utility prints a summary of the current activity on the system, including what each user is doing. The first line displays the current time of day, how long the system has been running, the number of users logged into the system, and the load averages. The load average numbers give the number of jobs in the run queue averaged over 1, 5, and 15 minutes.

The fields output are the user's login name, the name of the terminal the user is on, the host from which the user is logged in, the time the user logged on, the time since the user last typed anything, and the name and arguments of the current process.

The options are as follows:

-h Suppress the heading.

-i Output is sorted by idle time.

-M Extract values associated with the name list from the specified core instead of the default "/dev/kmem".

-N Extract the name list from the specified system instead of the default "/netbsd".

-n Show network addresses as numbers (normally **w** interprets addresses and attempts to display them symbolically).

-w Show wide output without truncating any fields.

If a *user* name is specified, the output is restricted to that user.

FILES

`/var/run/utmp` list of users on the system

SEE ALSO

`finger`(1), `ps`(1), `uptime`(1), `who`(1)

HISTORY

The **w** command appeared in 3.0 BSD.

BUGS

The notion of the "current process" is muddy. The current algorithm is ''the highest numbered process on the terminal that is not ignoring interrupts, or, if there is none, the highest numbered process on the terminal''. This fails, for example, in critical sections of programs like the shell and editor, or when faulty programs running in the background fork and fail to ignore interrupts. (In cases where no process can be found, **w** prints "−".)

Background processes are not shown, even though they account for much of the load on the system.

Sometimes processes, typically those in the background, are printed with null or garbaged arguments. In these cases, the name of the command is printed in parentheses.

The **w** utility does not know about the new conventions for detection of background jobs. It will sometimes find a background job instead of the right one.

NAME

wait — await process completion

SYNOPSIS

wait [*pid*]

DESCRIPTION

If invoked with no arguments, the **wait** utility waits until all existing child processes in the background have terminated.

Available operands:

pid If a *pid* operand is specified, and it is the process ID of a background child process that still exists, the **wait** utility waits until that process has completed and consumes its status information, without consuming the status information of any other process.

If a pid operand is specified that is not the process ID of a child background process that still exists, **wait** exits without waiting for any processes to complete.

The **wait** utility exits with one of the following values:

0 The **wait** utility was invoked with no operands and all of the existing background child processes have terminated, or the process specified by the pid operand exited normally with 0 as its exit status.

>0 The specified process did not exist and its exit status information was not available, or the specified process existed or its exit status information was available, and it terminated with a non-zero exit status.

If the specified process terminated abnormally due to the receipt of a signal, the exit status information of **wait** contains that termination status as well.

STANDARDS

The **wait** command is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

wall — write a message to users

SYNOPSIS

wall [**-g** *group*] [*file*]

DESCRIPTION

wall displays the contents of *file* or, by default, its standard input, on the terminals of all currently logged in users.

Only the super-user can write on the terminals of users who have chosen to deny messages or are using a program which automatically denies messages.

-g Send messages to users in this group. This option may be specified multiple times, and any user in any of the specified groups will receive the message.

SEE ALSO

mesg(1), talk(1), write(1), shutdown(8)

HISTORY

A **wall** command appeared in Version 7 AT&T UNIX. Support for **-g** was added in NetBSD 2.0 (from OpenBSD 2.0).

NAME

wc — word, line, and byte count

SYNOPSIS

wc [**-c** | **-m**] [**-lw**] [*file* *. . .*]

DESCRIPTION

The **wc** utility displays the number of lines, words, bytes and characters contained in each input *file* (or standard input, by default) to the standard output. A line is defined as a string of characters delimited by a <newline> character, and a word is defined as a string of characters delimited by white space characters. White space characters are the set of characters for which the `iswspace`(3) function returns true. If more than one input file is specified, a line of cumulative counts for all the files is displayed on a separate line after the output for the last file.

The following options are available:

-c The number of bytes in each input file is written to the standard output.

-l The number of lines in each input file is written to the standard output.

-m The number of characters in each input file is written to the standard output.

-w The number of words in each input file is written to the standard output.

When an option is specified, **wc** only reports the information requested by that option. The default action is equivalent to all the flags **-clw** having been specified.

The following operands are available:

file A pathname of an input file.

If no file names are specified, the standard input is used and no file name is displayed.

By default, the standard output contains a line for each input file of the form:

```
lines   words  bytes   file_name
```

The **wc** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

`iswspace`(3)

COMPATIBILITY

Historically, the **wc** utility was documented to define a word as a ‘‘maximal string of characters delimited by <space>, <tab> or <newline> characters’’. The implementation, however, didn’t handle non-printing characters correctly so that ‘‘ ^D^E ’’ counted as 6 spaces, while ‘‘foo^D^Ebar’’ counted as 8 characters. 4BSD systems after 4.3BSD modified the implementation to be consistent with the documentation. This implementation defines a ‘‘word’’ in terms of the `iswspace`(3) function, as required by IEEE Std 1003.2 (“POSIX.2”).

STANDARDS

The **wc** utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

NAME

what — search files for SCCS identifiers

SYNOPSIS

what [**-s**] *file ...*

DESCRIPTION

The **what** utility reads each *file* operand and searches for sequences of the form "@(#)" as inserted by the source code control system. It prints the remainder of the string following this marker, up to a null character, newline, double quote, backslash, or ">" character.

If the **-s** option is specified, only the first occurrence of an identification string in each file is printed.

EXIT STATUS

The **what** utility exits 0 if any matches were found, and 1 otherwise.

STANDARDS

The **what** utility conforms to X/Open Portability Guide Issue 4, Version 2 ("XPG4.2").

HISTORY

The **what** command appeared in 4.0BSD.

BUGS

As BSD is not licensed to distribute SCCS this is a rewrite of the **what** command which is part of SCCS, and may not behave exactly the same as that command does.

NAME

whatis — describe what a command is

SYNOPSIS

whatis [**-C** *path*] [**-M** *path*] [**-m** *path*] *command ...*

DESCRIPTION

whatis looks up a given command and gives the header line from the manual page. You can then use the man(1) command to get more information.

The options are as follows:

-C *path* Use different man(1) configuration file than the default, /etc/man.conf.

-M *path* Override the list of standard directories **whatis** searches for its database named "whatis.db". The supplied *path* must be a colon ":" separated list of directories. This search path may also be set using the environment variable MANPATH.

-m *path* Augment the list of standard directories **whatis** searches for its database named "whatis.db". The supplied *path* must be a colon ":" separated list of directories. These directories will be searched before the standard directories or the directories supplied with the **-M** option or the MANPATH environment variable are searched.

ENVIRONMENT

MANPATH The standard search path used by man(1) may be overridden by specifying a path in the MANPATH environment variable.

FILES

whatis.db	name of the whatis databases
/etc/man.conf	man(1) configuration file, used to get location of whatis databases if MANPATH is not set.

SEE ALSO

apropos(1), man(1), whereis(1), man.conf(5), makewhatis(8)

HISTORY

The **whatis** command appeared in 3.0BSD.

NAME

whereis — locate programs

SYNOPSIS

whereis [**-p**] *program* [*program ...*]

DESCRIPTION

The **whereis** utility checks the standard binary directories for the specified programs, printing out the paths of any it finds.

The default path searched is the string returned by the sysctl(8) utility for the "user.cs_path" string. If the [**-p**] option is specified, then the value of the environment variable PATH is used instead.

EXIT STATUS

The **whereis** utility exits 0 on success, 1 on general error, 2 if only some programs were located and 3 if none were.

SEE ALSO

whatis(1), which(1), sysctl(8)

COMPATIBILITY

The historic flags and arguments for the **whereis** utility are no longer available in this version.

HISTORY

The **whereis** command appeared in 3.0BSD.

NAME

which — locate a program file in the users $PATH environment variable

SYNOPSIS

which [**-a**] *name* [**. . .**]

DESCRIPTION

which takes a list of names and looks for the files which would be executed had these names been given as commands. Each argument is searched for along the user's PATH.

If the **-a** flag is given, **which** will continue to search the PATH until all instances of a program file are found.

HISTORY

The **which** command appeared in 3.0 BSD.

BUGS

This implementation does not expand csh(1) aliases, and is shell agnostic. This is really a feature.

NAME

who — display who is logged in

SYNOPSIS

who [**-abdHlmqrstTuv**] [*file*]
who *am i*

DESCRIPTION

The **who** utility displays a list of all users currently logged on, showing for each user the login name, tty name, the date and time of login, and hostname if not local.

Available options:

-a Same as **-bdlprTtuv**.

-b Time of last system boot.

-d Print dead processes.

-H Write column headings above the regular output.

-l Print system login processes.

-m Only print information about the current terminal. This is the POSIX way of saying **who** *am i*.

-p Print active processes spawned by `init`(8).

-q "Quick mode": List only the names and the number of users currently logged on. When this option is used, all other options are ignored.

-r Print the current runlevel. Supported runlevels are:

`d (DEATH)`	The system has halted.
`s (SINGLE_USER)`	The system is running in single user mode.
`r (RUNCOM)`	The system is executing `/etc/rc`.
`t (READ_TTYS)`	The system is processing `/etc/ttys`.
`m (MULTI_USER)`	The system is running in multi-user mode.
`T (CLEAN_TTYS)`	The system is in the process of stopping processes associated with terminal devices.
`c (CATATONIA)`	The system is in the process of shutting down and will not create new processes.

-s List only the name, line and time fields. This is the default.

-T Print a character after the user name indicating the state of the terminal line: '+' if the terminal is writable; '-' if it is not; and '?' if a bad line is encountered.

-t Print last system clock change.

-u Print the idle time for each user, and the associated process ID.

-v When printing of more information is requested with **-u**, this switch can be used to also printed process termination signals, process exit status, session id for windowing and the type of the entry, see documentation of ut_type in `getutxent`(3).

am I Returns the invoker's real user name.

file By default, **who** gathers information from the file /var/run/utmpx. An alternative *file* may be specified which is usually /var/log/wtmpx (or /var/log/wtmp, or /var/log/wtmpx.[0-6] or /var/log/wtmp.[0-6] depending on site policy as wtmpx can grow quite large and daily versions may or may not be kept around after compression by ac(8)). The wtmpx and wtmp file contains a record of every login, logout, crash, shutdown and date change since wtmpx and wtmp were last truncated or created.

If /var/log/wtmpx or /var/log/wtmp are being used as the file, the user name may be empty or one of the special characters '|', '}' and '~'. Logouts produce an output line without any user name. For more information on the special characters, see utmp(5).

FILES

/var/run/utmp
/var/run/utmpx
/var/log/wtmp
/var/log/wtmp.[0-6]
/var/log/wtmpx
/var/log/wtmpx.[0-6]

SEE ALSO

last(1), mesg(1), users(1), getuid(2), utmp(5), utmpx(5)

STANDARDS

The **who** utility is expected to conform to IEEE Std 1003.2-1992 ("POSIX.2").

HISTORY

A **who** utility appeared in Version 6 AT&T UNIX.

NAME

whoami — display effective user id

SYNOPSIS

whoami

DESCRIPTION

The **whoami** utility has been obsoleted by the id(1) utility, and is equivalent to "**id -un**". The command "**id -p**" is suggested for normal interactive use.

The **whoami** utility displays your effective user ID as a name.

The **whoami** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

id(1)

NAME

whois — Internet domain name and network number directory service

SYNOPSIS

whois [**-6AadfgilmQRr**][**-c** *country-code* | **-h** *host*][**-p** *port*]*name* [**. . .**]

DESCRIPTION

The **whois** utility looks up records in the databases maintained by several Network Information Centers (NICs).

The options are as follows:

-6 Use the IPv6 Resource Center (6bone) database. It contains network names and addresses for the IPv6 network.

-A Use the Asia/Pacific Network Information Center (APNIC) database. It contains network numbers used in East Asia, Australia, New Zealand, and the Pacific islands.

-a Use the American Registry for Internet Numbers (ARIN) database. It contains network numbers used in those parts of the world not covered by AfriNIC, APNIC, LACNIC or by RIPE.

(Hint: All point of contact handles in the ARIN whois database end with "`-ARIN`".)

-c *country-code*
This is the equivalent of using the **-h** option with an argument of "*country-code*`.whois-servers.net`".

-d Use the US Department of Defense database. It contains points of contact for subdomains of .MIL.

-f Use the African Network Information Center (AfriNIC) database. It contains network numbers used in Africa

-g Use the US non-military federal government database, which contains points of contact for subdomains of .GOV.

-h *host*
Use the specified host instead of the default NIC (whois.crsnic.net). Either a host name or an IP address may be specified.

By default **whois** constructs the name of a whois server to use from the top-level domain (TLD) of the supplied (single) argument, and appending "`.whois-servers.net`". This effectively allows a suitable whois server to be selected automatically for a large number of TLDs.

In the event that an IP address is specified, the whois server will default to the American Registry for Internet Numbers (ARIN). If a query to ARIN references AfriNIC, APNIC, LACNIC, or RIPE, that server will be queried also, provided that the **-Q** option is not specified.

If the query is not a domain name or IP address, **whois** will fall back to `whois.crsnic.net`.

-i Use the Network Solutions Registry for Internet Numbers (whois.networksolutions.com) database. Historically, it contained network numbers and domain contact information for most of .COM, .NET, .ORG and .EDU domains. However, the registration of these domains is now done by a number of independent and competing registrars and this database holds no information on the domains registered by organizations other than Network Solutions, Inc. Also, note that the InterNIC database (`whois.internic.net`) is no longer handled by Network Solutions, Inc. For details, see `http://www.internic.net/`.

(Hint: Contact information, identified by the term *handle*, can be looked up by prefixing "`!`" or "`handle` " to the NIC handle in the query.)

-l Use the Latin American and Caribbean IP address Regional Registry (LACNIC) database. It contains network numbers used in much of Latin America and the Caribbean.

-m Use the Route Arbiter Database (RADB) database. It contains route policy specifications for a large number of operators' networks.

-p *port*
Connect to the whois server on *port*. If this option is not specified, **whois** defaults to the "whois" port listed in `/etc/services` (port 43).

-Q Do a quick lookup. This means that **whois** will not attempt to lookup the name in the authoritative whois server (if one is listed) nor will it contact InterNic if a lookup fails. This flag has no effect when combined with any other flag.

-R Use the Russia Network Information Center (RIPN) database. It contains network numbers and domain contact information for subdomains of .RU. This option is deprecated; use the **-c** option with an argument of "RU" instead.

-r Use the R´eseaux IP Europ´eens (RIPE) database. It contains network numbers and domain contact information for Europe.

The default action, unless directed otherwise with a special *name*, is to do a very broad search, looking for matches to *name* in all types of records and most fields (name, nicknames, hostname, net address, etc.) in the database. For more information as to what *name* operands have special meaning, and how to guide the search, use the special name "help".

Special cases

Queries beginning with an exclamation point '!' are assumed to be NSI contact handles. Unless a host or domain is specified on the command line, (whois.networksolutions.com) will be used as the **whois** database.

Similarly, queries beginning with "COCO-" are assumed to be CORE contact handles. Unless a host or domain is specified on the command line, (whois.corenic.net) will be used as the **whois** database.

EXAMPLES

Most types of data, such as domain names and IP addresses, can be used as arguments to **whois** without any options, and **whois** will choose the correct whois server to query. Some exceptions, where **whois** will not be able to handle data correctly, are detailed below.

To obtain contact information about an administrator located in the Russian TLD domain "RU", use the **-c** option as shown in the following example, where *CONTACT-ID* is substituted with the actual contact identifier.

```
whois -c RU CONTACT-ID
```

(Note: This example is specific to the TLD "RU", but other TLDs can be queried by using a similar syntax.)

The following example demonstrates how to obtain information about an IPv6 address or hostname using the **-6** option, which directs the query to 6bone.

```
whois -6 IPv6-IP-Address
```

The following example demonstrates how to query a whois server using a non-standard port, where "`query-data`" is the query to be sent to "`whois.example.com`" on port "`rwhois`" (written numerically as 4321).

```
whois -h whois.example.com -p rwhois query-data
```

SEE ALSO

Ken Harrenstien and Vic White, *NICNAME/WHOIS*, 1 March 1982, RFC 812.

HISTORY

The **whois** command appeared in 4.3BSD.

NAME

window — window environment

SYNOPSIS

window [**-t**] [**-f**] [**-d**] [**-e** *escape-char*] [**-c** *command*]

DESCRIPTION

window implements a window environment on ASCII terminals.

A window is a rectangular portion of the physical terminal screen associated with a set of processes. Its size and position can be changed by the user at any time. Processes communicate with their window in the same way they normally interact with a terminal−through their standard input, output, and diagnostic file descriptors. The window program handles the details of redirecting input and output to and from the windows. At any one time, only one window can receive input from the keyboard, but all windows can simultaneously send output to the display.

When **window** starts up, the commands (see long commands below) contained in the file `.windowrc` in the user's home directory are executed. If it does not exist, two equal sized windows spanning the terminal screen are created by default.

The command line options are

-t Turn on terse mode (see **terse** command below).

-f Fast. Don't perform any startup action.

-d Ignore `.windowrc` and create the two default windows instead.

-e *escape-char*
Set the escape character to *escape-char*. *Escape-char* can be a single character, or in the form **^X** where *X* is any character, meaning control−*X*.

-c *command*
Execute the string *command* as a long command (see below) before doing anything else.

Windows can overlap and are framed as necessary. Each window is named by one of the digits "1" to "9". This one-character identifier, as well as a user definable label string, are displayed with the window on the top edge of its frame. A window can be designated to be in the *foreground*, in which case it will always be on top of all normal, non-foreground windows, and can be covered only by other foreground windows. A window need not be completely within the edges of the terminal screen. Thus a large window (possibly larger than the screen) may be positioned to show only a portion of its full size.

Each window has a cursor and a set of control functions. Most intelligent terminal operations such as line and character deletion and insertion are supported. Display modes such as underlining and reverse video are available if they are supported by the terminal. In addition, similar to terminals with multiple pages of memory, each window has a text buffer which can have more lines than the window itself.

Process Environment

With each newly created window, a shell program is spawned with its process environment tailored to that window. Its standard input, output, and diagnostic file descriptors are bound to one end of either a pseudo-terminal (see `pty`(4)) or a UNIX domain socket (see `socketpair`(2)). If a pseudo-terminal is used, then its special characters and modes (see `stty`(1)) are copied from the physical terminal. A `termcap`(5) entry tailored to this window is created and passed as environment (see `environ`(7)) variable `TERMCAP`. The termcap entry contains the window's size and characteristics as well as information from the physical terminal, such as the existence of underline, reverse video, and other display modes, and the codes produced by the terminal's function keys, if any. In addition, the window size attributes of the pseudo-terminal are set to reflect the size of this window, and updated whenever it is changed by the user. In particular, the editor

vi(1) uses this information to redraw its display.

Operation

During normal execution, **window** can be in one of two states: conversation mode and command mode. In conversation mode, the terminal's real cursor is placed at the cursor position of a particular window--called the current window--and input from the keyboard is sent to the process in that window. The current window is always on top of all other windows, except those in foreground. In addition, it is set apart by highlighting its identifier and label in reverse video.

Typing **window**'s escape character (normally **^P**) in conversation mode switches it into command mode. In command mode, the top line of the terminal screen becomes the command prompt window, and **window** interprets input from the keyboard as commands to manipulate windows.

There are two types of commands: short commands are usually one or two key strokes; long commands are strings either typed by the user in the command window (see the "**:**" command below), or read from a file (see **source** below).

Short Commands

Below, *#* represents one of the digits "1" to "9" corresponding to the windows 1 to 9. **^X** means control−*X*, where *X* is any character. In particular, ^^ is control−^. *Escape* is the escape key, or **^[**.

Select window *#* as the current window and return to conversation mode.

%*#* Select window *#* but stay in command mode.

^^ Select the previous window and return to conversation mode. This is useful for toggling between two windows.

escape
Return to conversation mode.

^P Return to conversation mode and write **^P** to the current window. Thus, typing two **^P**'s in conversation mode sends one to the current window. If the **window** escape is changed to some other character, that character takes the place of **^P** here.

? List a short summary of commands.

^L Refresh the screen.

q Exit **window**. Confirmation is requested.

^Z Suspend **window**.

w Create a new window. The user is prompted for the positions of the upper left and lower right corners of the window. The cursor is placed on the screen and the keys "h", "j", "k", and "l" move the cursor left, down, up, and right, respectively. The keys "H", "J", "K", and "L" move the cursor to the respective limits of the screen. Typing a number before the movement keys repeats the movement that number of times. Return enters the cursor position as the upper left corner of the window. The lower right corner is entered in the same manner. During this process, the placement of the new window is indicated by a rectangular box drawn on the screen, corresponding to where the new window will be framed. Typing escape at any point cancels this command.

This window becomes the current window, and is given the first available ID. The default buffer size is used (see *default_nline* command below).

Only fully visible windows can be created this way.

c# Close window #. The process in the window is sent the hangup signal (see `kill`(1)). `csh`(1) should handle this signal correctly and cause no problems.

m# Move window # to another location. A box in the shape of the window is drawn on the screen to indicate the new position of the window, and the same keys as those for the **w** command are used to position the box. The window can be moved partially off-screen.

M# Move window # to its previous position.

s# Change the size of window #. The user is prompted to enter the new lower right corner of the window. A box is drawn to indicate the new window size. The same keys used in **w** and **m** are used to enter the position.

S# Change window # to its previous size.

^Y Scroll the current window up by one line.

^E Scroll the current window down by one line.

^U Scroll the current window up by half the window size.

^D Scroll the current window down by half the window size.

^B Scroll the current window up by the full window size.

^F Scroll the current window down by the full window size.

h Move the cursor of the current window left by one column.

j Move the cursor of the current window down by one line.

k Move the cursor of the current window up by one line.

l Move the cursor of the current window right by one column.

y Yank. The user is prompted to enter two points within the current window. Then the content of the current window between those two points is saved in the yank buffer.

p Put. The content of the yank buffer is written to the current window as input.

^S Stop output in the current window.

^Q Start output in the current window.

: Enter a line to be executed as long commands. Normal line editing characters (erase character, erase word, erase line) are supported.

Long Commands

Long commands are a sequence of statements parsed much like a programming language, with a syntax similar to that of C. Numeric and string expressions and variables are supported, as well as conditional statements.

There are two data types: string and number. A string is a sequence of letters or digits beginning with a letter. “_” and “.” are considered letters. Alternatively, non-alphanumeric characters can be included in strings by quoting them in double (“"”) quotes or escaping them with backslash (“\”). In addition, the “\” sequences of C are supported, both inside and outside quotes (e.g., “\n” is a new line, “\r” a carriage return). For example, these are legal strings: abcde01234, "&#$^*&#", ab"$#"cd, ab\$\#cd, "/usr/ucb/window".

A number is an integer value in one of three forms: a decimal number, an octal number preceded by “0”, or a hexadecimal number preceded by “0x” or “0X”. The natural machine integer size is used (i.e., the signed integer type of the C compiler). As in C, a non-zero number represents a boolean true.

The character "#" begins a comment which terminates at the end of the line.

A statement is either a conditional or an expression. Expression statements are terminated with a new line or ";". To continue an expression on the next line, terminate the first line with "\".

Conditional Statement

window has a single control structure: the fully bracketed if statement in the form

```
if <expr> then
<statement>
...
elsif <expr> then
<statement>
...
else
<statement>
...
endif
```

The **else** and **elsif** parts are optional, and the latter can be repeated any number of times. <Expr> must be numeric.

Expressions

Expressions in **window** are similar to those in the C language, with most C operators supported on numeric operands. In addition, some are overloaded to operate on strings.

When an expression is used as a statement, its value is discarded after evaluation. Therefore, only expressions with side effects (assignments and function calls) are useful as statements.

Single valued (no arrays) variables are supported, of both numeric and string values. Some variables are predefined. They are listed below.

The operators in order of increasing precedence:

⟨*expr1*⟩ **=** ⟨*expr2*⟩
: Assignment. The variable of name ⟨*expr1*⟩, which must be string valued, is assigned the result of ⟨*expr2*⟩. Returns the value of ⟨*expr2*⟩.

⟨*expr1*⟩ **?** ⟨*expr2*⟩ : ⟨*expr3*⟩
: Returns the value of ⟨*expr2*⟩ if ⟨*expr1*⟩ evaluates true (non-zero numeric value); returns the value of ⟨*expr3*⟩ otherwise. Only one of ⟨*expr2*⟩ and ⟨*expr3*⟩ is evaluated. ⟨*Expr1*⟩ must be numeric.

⟨*expr1*⟩ **||** ⟨*expr2*⟩
: Logical or. Numeric values only. Short circuit evaluation is supported (i.e., if ⟨*expr1*⟩ evaluates true, then ⟨*expr2*⟩ is not evaluated).

⟨*expr1*⟩ **&&** ⟨*expr2*⟩
: Logical and with short circuit evaluation. Numeric values only.

⟨*expr1*⟩ **|** ⟨*expr2*⟩
: Bitwise or. Numeric values only.

⟨*expr1*⟩ **^** ⟨*expr2*⟩
: Bitwise exclusive or. Numeric values only.

⟨*expr1*⟩ **&** ⟨*expr2*⟩
Bitwise and. Numeric values only.

⟨*expr1*⟩ **==** ⟨*expr2*⟩, ⟨*expr1*⟩ **!=** ⟨expr2⟩
Comparison (equal and not equal, respectively). The boolean result (either 1 or 0) of the comparison is returned. The operands can be numeric or string valued. One string operand forces the other to be converted to a string in necessary.

⟨*expr1*⟩ **<** ⟨*expr2*⟩, ⟨*expr1*⟩ **>** ⟨*expr2*⟩, ⟨*expr1*⟩ ≤ ⟨*expr2*⟩,
Less than, greater than, less than or equal to, greater than or equal to. Both numeric and string values, with automatic conversion as above.

⟨*expr1*⟩ **<<** ⟨*expr2*⟩, ⟨*expr1*⟩ **>>** ⟨*expr2*⟩
If both operands are numbers, ⟨*expr1*⟩ is bit shifted left (or right) by ⟨*expr2*⟩ bits. If ⟨*expr1*⟩ is a string, then its first (or last) ⟨*expr2*⟩ characters are returns (if ⟨*expr2*⟩ is also a string, then its length is used in place of its value).

⟨*expr1*⟩ **+** ⟨*expr2*⟩, ⟨*expr1*⟩ **-** ⟨*expr2*⟩
Addition and subtraction on numbers. For "+", if one argument is a string, then the other is converted to a string, and the result is the concatenation of the two strings.

⟨*expr1*⟩ ∗ ⟨*expr2*⟩, ⟨*expr1*⟩ **/** ⟨*expr2*⟩, ⟨*expr1*⟩ **%** ⟨*expr2*⟩
Multiplication, division, modulo. Numbers only.

−⟨*expr*⟩, **~**⟨*expr*⟩, **!**⟨*expr*⟩, **$**⟨*expr*⟩, **$?**⟨*expr*⟩
The first three are unary minus, bitwise complement and logical complement on numbers only. The operator, "$", takes ⟨*expr*⟩ and returns the value of the variable of that name. If ⟨*expr*⟩ is numeric with value *n* and it appears within an alias macro (see below), then it refers to the nth argument of the alias invocation. "$?" tests for the existence of the variable ⟨*expr*⟩, and returns 1 if it exists or 0 otherwise.

⟨*expr*⟩**(**⟨*arglist*⟩**)**
Function call. ⟨*Expr*⟩ must be a string that is the unique prefix of the name of a builtin **window** function or the full name of a user defined alias macro. In the case of a builtin function, ⟨*arglist*⟩ can be in one of two forms:

```
<expr1>, <expr2>, ...
argname1 = <expr1>, argname2 = <expr2>, ...
```

The two forms can in fact be intermixed, but the result is unpredictable. Most arguments can be omitted; default values will be supplied for them. The *argnames* can be unique prefixes of the argument names. The commas separating arguments are used only to disambiguate, and can usually be omitted.

Only the first argument form is valid for user defined aliases. Aliases are defined using the **alias** builtin function (see below). Arguments are accessed via a variant of the variable mechanism (see "$" operator above).

Most functions return value, but some are used for side effect only and so must be used as statements. When a function or an alias is used as a statement, the parentheses surrounding the argument list may be omitted. Aliases return no value.

Builtin Functions

The arguments are listed by name in their natural order. Optional arguments are in square brackets '[ ]'. Arguments that have no names are in angle brackets '<>'. An argument meant to be a boolean flag (often named *flag*) can be one of *on*, *off*, *yes*, *no*, *true*, or *false*, with obvious meanings, or it can be a numeric expression, in which case a non-zero value is true.

alias([⟨*string*⟩], [⟨*string−list*⟩)⟩]
If no argument is given, all currently defined alias macros are listed. Otherwise, ⟨*string*⟩ is defined as an alias, with expansion ⟨*string−list* >⟩. The previous definition of ⟨*string*⟩, if any, is returned. Default for ⟨*string−list*⟩ is no change.

close(⟨*window−list*⟩)
Close the windows specified in ⟨*window−list*⟩. If ⟨*window−list*⟩ is the word *all*, than all windows are closed. No value is returned.

cursormodes([*modes*])
Set the window cursor to *modes*. *Modes* is the bitwise or of the mode bits defined as the variables *m_ul* (underline), *m_rev* (reverse video), *m_blk* (blinking), and *m_grp* (graphics, terminal dependent). Return value is the previous modes. Default is no change. For example, `cursor($m_rev$m_blk)` sets the window cursors to blinking reverse video.

default_nline([*nline*])
Set the default buffer size to *nline*. Initially, it is 48 lines. Returns the old default buffer size. Default is no change. Using a very large buffer can slow the program down considerably.

default_shell([⟨*string−list*⟩])
Set the default window shell program to ⟨*string−list*⟩. Returns the first string in the old shell setting. Default is no change. Initially, the default shell is taken from the environment variable SHELL.

default_smooth([*flag*])
Set the default value of the *smooth* argument to the command **window** (see below). The argument is a boolean flag (one of *on*, *off*, *yes*, *no*, *true*, *false*, or a number, as described above). Default is no change. The old value (as a number) is returned. The initial value is 1 (true).

echo([*window*], [⟨*string−list*⟩])
Write the list of strings, ⟨*string-list*⟩, to **window**, separated by spaces and terminated with a new line. The strings are only displayed in the window, the processes in the window are not involved (see **write** below). No value is returned. Default is the current window.

escape([*escapec*])
Set the escape character to *escape-char*. Returns the old escape character as a one-character string. Default is no change. *Escapec* can be a string of a single character, or in the form **−^X**, meaning control−*X*.

foreground([*window*], [*flag*])
Move **window** in or out of foreground. *Flag* is a boolean value. The old foreground flag is returned. Default for **window** is the current window, default for *flag* is no change.

label([*window*], [*label*])
Set the label of **window** to *label*. Returns the old label as a string. Default for **window** is the current window, default for *label* is no change. To turn off a label, set it to an empty string ("").

list( ) No arguments. List the identifiers and labels of all windows. No value is returned.

select([*window*])
Make **window** the current window. The previous current window is returned. Default is no change.

source(*filename*)
Read and execute the long commands in *filename*. Returns −1 if the file cannot be read, 0 otherwise.

terse([flag])
Set terse mode to *flag*. In terse mode, the command window stays hidden even in command mode, and errors are reported by sounding the terminal's bell. *Flag* can take on the same values as in *foreground* above. Returns the old terse flag. Default is no change.

unalias(*alias*)
Undefine *alias*. Returns -1 if *alias* does not exist, 0 otherwise.

unset(*variable*)
Undefine *variable*. Returns -1 if *variable* does not exist, 0 otherwise.

variables()
No arguments. List all variables. No value is returned.

window([*row*], [*column*], [*nrow*], [*ncol*], [*nline*], [*label*], [*pty*], [*frame*], [*mapnl*], [*keepopen*], [*smooth*], [*shell*]).
Open a window with upper left corner at *row*, *column* and size *nrow*, *ncol*. If *nline* is specified, then that many lines are allocated for the text buffer. Otherwise, the default buffer size is used. Default values for *row*, *column*, *nrow*, and *ncol* are, respectively, the upper, left-most, lower, or right-most extremes of the screen. *Label* is the label string. *Frame*, *pty*, and *mapnl* are flag values interpreted in the same way as the argument to *foreground* (see above); they mean, respectively, put a frame around this window (default true), allocate pseudo-terminal for this window rather than socketpair (default true), and map new line characters in this window to carriage return and line feed (default true if socketpair is used, false otherwise). Normally, a window is automatically closed when its process exits. Setting *keepopen* to true (default false) prevents this action. When *smooth* is true, the screen is updated more frequently (for this window) to produce a more terminal-like behavior. The default value of *smooth* is set by the *default_smooth* command (see above). *Shell* is a list of strings that will be used as the shell program to place in the window (default is the program specified by *default_shell*, see above). The created window's identifier is returned as a number.

write([*window*], [⟨*string-list*⟩])
Send the list of strings, ⟨*string-list*⟩, to **window**, separated by spaces but not terminated with a new line. The strings are actually given to the window as input. No value is returned. Default is the current window.

Predefined Variables

These variables are for information only. Redefining them does not affect the internal operation of **window**.

baud The baud rate as a number between 50 and 38400.

modes The display modes (reverse video, underline, blinking, graphics) supported by the physical terminal. The value of *modes* is the bitwise or of some of the one bit values, *m_blk*, *m_grp*, *m_rev*, and *m_ul* (see below). These values are useful in setting the window cursors' modes (see *cursormodes* above).

m_blk The blinking mode bit.

m_grp The graphics mode bit (not very useful).

m_rev The reverse video mode bit.

m_ul The underline mode bit.

ncol The number of columns on the physical screen.

nrow The number of rows on the physical screen.

term The terminal type. The standard name, found in the second name field of the terminal's TERMCAP entry, is used.

ENVIRONMENT

window uses these environment variables: HOME, SHELL, TERM, TERMCAP, WINDOW_ID.

FILES

~/.windowrc	startup command file.
/dev/[pt]ty[pq]?	pseudo-terminal devices.

DIAGNOSTICS

Should be self explanatory.

HISTORY

The **window** command appeared in 4.3BSD.

NAME

windres − manipulate Windows resources.

SYNOPSIS

windres [options] [input−file] [output−file]

DESCRIPTION

windres reads resources from an input file and copies them into an output file. Either file may be in one of three formats:

`rc` A text format read by the Resource Compiler.

`res`
A binary format generated by the Resource Compiler.

`coff`
A COFF object or executable.

The exact description of these different formats is available in documentation from Microsoft.

When **windres** converts from the `rc` format to the `res` format, it is acting like the Windows Resource Compiler. When **windres** converts from the `res` format to the `coff` format, it is acting like the Windows `CVTRES` program.

When **windres** generates an `rc` file, the output is similar but not identical to the format expected for the input. When an input `rc` file refers to an external filename, an output `rc` file will instead include the file contents.

If the input or output format is not specified, **windres** will guess based on the file name, or, for the input file, the file contents. A file with an extension of *.rc* will be treated as an `rc` file, a file with an extension of *.res* will be treated as a `res` file, and a file with an extension of *.o* or *.exe* will be treated as a `coff` file.

If no output file is specified, **windres** will print the resources in `rc` format to standard output.

The normal use is for you to write an `rc` file, use **windres** to convert it to a COFF object file, and then link the COFF file into your application. This will make the resources described in the `rc` file available to Windows.

OPTIONS

−i *filename*
−−input *filename*
The name of the input file. If this option is not used, then **windres** will use the first non-option argument as the input file name. If there are no non-option arguments, then **windres** will read from standard input. **windres** can not read a COFF file from standard input.

−o *filename*
−−output *filename*
The name of the output file. If this option is not used, then **windres** will use the first non-option argument, after any used for the input file name, as the output file name. If there is no non-option argument, then **windres** will write to standard output. **windres** can not write a COFF file to standard output. Note, for compatability with **rc** the option **−fo** is also accepted, but its use is not recommended.

−J *format*
−−input−format *format*
The input format to read. *format* may be **res**, **rc**, or **coff**. If no input format is specified, **windres** will guess, as described above.

−O *format*
−−output−format *format*
The output format to generate. *format* may be **res**, **rc**, or **coff**. If no output format is specified, **windres** will guess, as described above.

−F *target*
−−target *target*
Specify the BFD format to use for a COFF file as input or output. This is a BFD target name; you can use the **−−help** option to see a list of supported targets. Normally **windres** will use the default format, which is the first one listed by the **−−help** option.

−−preprocessor *program*
When **windres** reads an `rc` file, it runs it through the C preprocessor first. This option may be used to specify the preprocessor to use, including any leading arguments. The default preprocessor argument is `gcc −E −xc−header −DRC_INVOKED`.

−I *directory*
−−include−dir *directory*
Specify an include directory to use when reading an `rc` file. **windres** will pass this to the preprocessor as an **−I** option. **windres** will also search this directory when looking for files named in the `rc` file. If the argument passed to this command matches any of the supported *formats* (as descrived in the **−J** option), it will issue a deprecation warning, and behave just like the **−J** option. New programs should not use this behaviour. If a directory happens to match a *format*, simple prefix it with **./** to disable the backward compatibility.

−D *target*
−−define *sym***[=***val***]**
Specify a **−D** option to pass to the preprocessor when reading an `rc` file.

−U *target*
−−undefine *sym*
Specify a **−U** option to pass to the preprocessor when reading an `rc` file.

−r Ignored for compatibility with rc.

−v Enable verbose mode. This tells you what the preprocessor is if you didn't specify one.

−l *val*
−−language *val*
Specify the default language to use when reading an `rc` file. *val* should be a hexadecimal language code. The low eight bits are the language, and the high eight bits are the sublanguage.

−−use−temp−file
Use a temporary file to instead of using popen to read the output of the preprocessor. Use this option if the popen implementation is buggy on the host (eg., certain non-English language versions of Windows 95 and Windows 98 are known to have buggy popen where the output will instead go the console).

−−no−use−temp−file
Use popen, not a temporary file, to read the output of the preprocessor. This is the default behaviour.

−h
−−help
Prints a usage summary.

−V
−−version
Prints the version number for **windres**.

−−yydebug
If **windres** is compiled with `YYDEBUG` defined as `1`, this will turn on parser debugging.

SEE ALSO

the Info entries for *binutils*.

COPYRIGHT

NAME

write — send a message to another user

SYNOPSIS

write *user* [*ttyname*]

DESCRIPTION

write allows you to communicate with other users, by copying lines from your terminal to theirs.

When you run the **write** command, the user you are writing to gets a message of the form:

```
Message from yourname@yourhost on yourtty at hh:mm ...
```

Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, they must run **write** as well.

When you are done, type an end-of-file or interrupt character. The other user will see the message `EOF` indicating that the conversation is over.

You can prevent people (other than the super-user) from writing to you with the `mesg`(1) command. Some commands, for example `nroff`(1) and `pr`(1), disallow writing automatically, so that your output isn't overwritten.

If the user you want to write to is logged in on more than one terminal, you can specify which terminal to write to by specifying the terminal name as the second operand to the **write** command. Alternatively, you can let **write** select one of the terminals − it will pick the one with the shortest idle time. This is so that if the user is logged in at work and also dialed up from home, the message will go to the right place.

The traditional protocol for writing to someone is that the string '`-o`', either at the end of a line or on a line by itself, means that it's the other person's turn to talk. The string '`oo`' means that the person believes the conversation to be over.

SEE ALSO

`mesg`(1), `talk`(1), `who`(1)

HISTORY

A **write** command appeared in Version 6 AT&T UNIX.

NAME

xargs — construct argument list(s) and execute utility

SYNOPSIS

xargs [**-0opt**] [**-E** *eofstr*] [**-I** *replstr* [**-R** *replacements*] [**-S** *replsize*]]
[**-J** *replstr*] [**-L** *number*] [**-n** *number* [**-x**]] [**-P** *maxprocs*] [**-s** *size*]
[*utility* [*argument ...*]]

DESCRIPTION

The **xargs** utility reads space, tab, newline and end-of-file delimited strings from the standard input and executes *utility* with the strings as arguments.

Any arguments specified on the command line are given to *utility* upon each invocation, followed by some number of the arguments read from the standard input of **xargs**. This is repeated until standard input is exhausted.

Spaces, tabs and newlines may be embedded in arguments using single (“ ’ ”) or double (“"”) quotes or backslashes (“\”). Single quotes escape all non-single quote characters, excluding newlines, up to the matching single quote. Double quotes escape all non-double quote characters, excluding newlines, up to the matching double quote. Any single character, including newlines, may be escaped by a backslash.

The options are as follows:

-0 Change **xargs** to expect NUL (“\0”) characters as separators, instead of spaces and newlines. This is expected to be used in concert with the **-print0** function in find(1).

-E *eofstr*
Use *eofstr* as a logical EOF marker.

-I *replstr*
Execute *utility* for each input line, replacing one or more occurrences of *replstr* in up to *replacements* (or 5 if no **-R** flag is specified) arguments to *utility* with the entire line of input. The resulting arguments, after replacement is done, will not be allowed to grow beyond *replsize* (or 255 if no **-S** flag is specified) bytes; this is implemented by concatenating as much of the argument containing *replstr* as possible, to the constructed arguments to *utility*, up to *replsize* bytes. The size limit does not apply to arguments to *utility* which do not contain *replstr*, and furthermore, no replacement will be done on *utility* itself. Implies **-x**.

-J *replstr*
If this option is specified, **xargs** will use the data read from standard input to replace the first occurrence of *replstr* instead of appending that data after all other arguments. This option will not affect how many arguments will be read from input (**-n**), or the size of the command(s) **xargs** will generate (**-s**). The option just moves where those arguments will be placed in the command(s) that are executed. The *replstr* must show up as a distinct *argument* to **xargs**. It will not be recognized if, for instance, it is in the middle of a quoted string. Furthermore, only the first occurrence of the *replstr* will be replaced. For example, the following command will copy the list of files and directories which start with an uppercase letter in the current directory to destdir:

```
/bin/ls -1d [A-Z]* | xargs -J % cp -rp % destdir
```

-L *number*
Call *utility* for every *number* lines read. If EOF is reached and fewer lines have been read than *number* then *utility* will be called with the available lines.

-n *number*
Set the maximum number of arguments taken from standard input for each invocation of *utility*. An invocation of *utility* will use less than *number* standard input arguments if the number of bytes accumulated (see the **-s** option) exceeds the specified *size* or there are fewer than *number* arguments remaining for the last invocation of *utility*. The current default value for *number* is 5000.

-o Reopen stdin as /dev/tty in the child process before executing the command. This is useful if you want **xargs** to run an interactive application.

-P *maxprocs*
Parallel mode: run at most *maxprocs* invocations of *utility* at once.

-p Echo each command to be executed and ask the user whether it should be executed. An affirmative response, 'y' in the POSIX locale, causes the command to be executed, any other response causes it to be skipped. No commands are executed if the process is not attached to a terminal.

-R *replacements*
Specify the maximum number of arguments that **-I** will do replacement in. If *replacements* is negative, the number of arguments in which to replace is unbounded.

-S *replsize*
Specify the amount of space (in bytes) that **-I** can use for replacements. The default for *replsize* is 255.

-s *size*
Set the maximum number of bytes for the command line length provided to *utility*. The sum of the length of the utility name, the arguments passed to *utility* (including NULL terminators) and the current environment will be less than or equal to this number. The current default value for *size* is ARG_MAX - 4096.

-t Echo the command to be executed to standard error immediately before it is executed.

-x Force **xargs** to terminate immediately if a command line containing *number* arguments will not fit in the specified (or default) command line length.

If *utility* is omitted, echo(1) is used.

Undefined behavior may occur if *utility* reads from the standard input.

The **xargs** utility exits immediately (without processing any further input) if a command line cannot be assembled, *utility* cannot be invoked, an invocation of *utility* is terminated by a signal, or an invocation of *utility* exits with a value of 255.

EXIT STATUS

xargs exits with one of the following values:

0 All invocations of *utility* returned a zero exit status.
123 One or more invocations of *utility* returned a nonzero exit status.
124 The *utility* exited with a 255 exit status.
125 The *utility* was killed or stopped by a signal.
126 The *utility* was found but could not be invoked.
127 The *utility* could not be found.
1 Some other error occurred.

FILES

`/dev/tty` used to read responses in prompt mode

SEE ALSO

`echo`(1), `find`(1), `execvp`(3)

STANDARDS

The **`xargs`** utility is expected to be IEEE Std 1003.2 ("POSIX.2") compliant. The **`-J`**, **`-o`**, **`-P`**, **`-R`**, and **`-S`** options are non-standard FreeBSD extensions which may not be available on other operating systems.

HISTORY

The **`xargs`** utility appeared in PWB UNIX 1.0. It made its first BSD appearance in the 4.3 Reno release.

The meaning of 123, 124, and 125 exit values and the **`-0`** option were taken from GNU xargs.

BUGS

If *`utility`* attempts to invoke another command such that the number of arguments or the size of the environment is increased, it risks `execvp`(3) failing with `E2BIG`.

The **`xargs`** utility does not take multibyte characters into account when performing string comparisons for the **`-I`** and **`-J`** options, which may lead to incorrect results in some locales.

NAME

xgettext − extract gettext strings from source

SYNOPSIS

xgettext [*OPTION*] [*INPUTFILE*]...

DESCRIPTION

Extract translatable strings from given input files.

Mandatory arguments to long options are mandatory for short options too. Similarly for optional arguments.

Input file location:

INPUTFILE ...
: input files

−f, **−−files−from**=*FILE*
: get list of input files from FILE

−D, **−−directory**=*DIRECTORY*
: add DIRECTORY to list for input files search

If input file is -, standard input is read.

Output file location:

−d, **−−default−domain**=*NAME*
: use NAME.po for output (instead of messages.po)

−o, **−−output**=*FILE*
: write output to specified file

−p, **−−output−dir**=*DIR*
: output files will be placed in directory DIR

If output file is -, output is written to standard output.

Choice of input file language:

−L, **−−language**=*NAME*
: recognise the specified language (C, C++, ObjectiveC, PO, Shell, Python, Lisp, EmacsLisp, librep, Scheme, Smalltalk, Java, JavaProperties, C#, awk, YCP, Tcl, Perl, PHP, GCC-source, NXStringTable, RST, Glade)

−C, **−−c**++
: shorthand for **−−language**=*C*++

By default the language is guessed depending on the input file name extension.

Input file interpretation:

−−from−code=*NAME*
: encoding of input files (except for Python, Tcl, Glade)

By default the input files are assumed to be in ASCII.

Operation mode:

−j, **−−join−existing**
: join messages with existing file

−x, **−−exclude−file**=*FILE*.po
: entries from FILE.po are not extracted

−c, **−−add−comments**[=*TAG*]
: place comment block with TAG (or those preceding keyword lines) in output file

Language specific options:

−a, **−−extract−all**
extract all strings (only languages C, C++, ObjectiveC, Shell, Python, Lisp, EmacsLisp, librep, Scheme, Java, C#, awk, Tcl, Perl, PHP, GCC-source, Glade)

−k, **−−keyword**[*=WORD*]
additional keyword to be looked for (without WORD means not to use default keywords)

(only languages C, C++, ObjectiveC, Shell, Python, Lisp, EmacsLisp, librep, Scheme, Java, C#, awk, Tcl, Perl, PHP, GCC-source, Glade)

−−flag=*WORD*:ARG:FLAG
additional flag for strings inside the argument number ARG of keyword WORD

(only languages C, C++, ObjectiveC, Shell, Python, Lisp, EmacsLisp, librep, Scheme, Java, C#, awk, YCP, Tcl, Perl, PHP, GCC-source)

−T, **−−trigraphs**
understand ANSI C trigraphs for input (only languages C, C++, ObjectiveC)

−−qt recognize Qt format strings (only language C++)

−−debug
more detailed formatstring recognition result

Output details:

−e, **−−no−escape**
do not use C escapes in output (default)

−E, **−−escape**
use C escapes in output, no extended chars

−−force−po
write PO file even if empty

−i, **−−indent**
write the .po file using indented style

−−no−location
do not write ’#: filename:line’ lines

−n, **−−add−location**
generate ’#: filename:line’ lines (default)

−−strict
write out strict Uniforum conforming .po file

−−properties−output
write out a Java .properties file

−−stringtable−output
write out a NeXTstep/GNUstep .strings file

−w, **−−width**=*NUMBER*
set output page width

−−no−wrap
do not break long message lines, longer than the output page width, into several lines

−s, **−−sort−output**
generate sorted output

−F, **−−sort−by−file**
sort output by file location

−−omit−header
don’t write header with ‘msgid ""’ entry

−−copyright−holder=*STRING*
set copyright holder in output

−−foreign−user
omit FSF copyright in output for foreign user

−−msgid−bugs−address=*EMAIL*@ADDRESS
set report address for msgid bugs

−m, **−−msgstr−prefix**[=*STRING*]
use STRING or "" as prefix for msgstr entries

−M, **−−msgstr−suffix**[=*STRING*]
use STRING or "" as suffix for msgstr entries

Informative output:

−h, **−−help**
display this help and exit

−V, **−−version**
output version information and exit

AUTHOR

Written by Ulrich Drepper.

REPORTING BUGS

Report bugs to <bug-gnu-gettext@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for **xgettext** is maintained as a Texinfo manual. If the **info** and **xgettext** programs are properly installed at your site, the command

info xgettext

should give you access to the complete manual.

NAME

xnlock − amusing lock screen program with message for passers-by

SYNOPSIS

xnlock [*options*] [*message*]

DESCRIPTION

xnlock is a program that acts as a screen saver for workstations running X11. It also "locks" the screen such that the workstation can be left unattended without worry that someone else will walk up to it and mess everything up. When *xnlock* is running, a little man with a big nose and a hat runs around spewing out messages to the screen. By default, the messages are "humorous", but that depends on your sense of humor.

If a key or mouse button is pressed, a prompt is printed requesting the user's password. If a RETURN is not typed within 30 seconds, the little man resumes running around.

Text on the command line is used as the message. For example:

% xnlock I´m out to lunch for a couple of hours.

Note the need to quote shell metacharacters.

In the absence of flags or text, *xnlock* displays random fortunes.

OPTIONS

Command line options override all resource specifications. All arguments that are not associated with a command line option is taken to be message text that the little man will "say" every once in a while. The resource **xnlock.text** may be set to a string.

−fn *fontname*

The default font is the first 18 point font in the *new century schoolbook* family. While larger fonts are recokmmended over smaller ones, any font in the server's font list will work. The resource to use for this option is **xnlock.font**.

−filename *filename*

Take the message to be displayed from the file *filename*. If *filename* is not specified, *$HOME/.msgfile* is used. If the contents of the file are changed during runtime, the most recent text of the file is used (allowing the displayed message to be altered remotely). Carriage returns within the text are allowed, but tabs or other control characters are not translated and should not be used. The resource available for this option is **xnlock.file**.

−ar Accept root's password to unlock screen. This option is true by default. The reason for this is so that someone's screen may be unlocked by autorized users in case of emergency and the person running the program is still out to lunch. The resource available for specifying this option is **xnlock.acceptRootPasswd**.

−noar Don't accept root's password. This option is for paranoids who fear their peers might breakin using root's password and remove their files anyway. Specifying this option on the command line overrides the **xnlock.acceptRootPasswd** if set to True.

−ip Ignore password prompt. The resource available for this option is **xnlock.ignorePasswd**.

−noip Don't ignore password prompt. This is available in order to override the resource **ignorePasswd** if set to True.

-fg *color*

Specifies the foreground color. The resource available for this is **xnlock.foreground**.

-bg *color*

Specifies the background color. The resource available for this is **xnlock.background**.

−rv Reverse the foreground and background colors. The resource for this is **xvnlock.reverseVideo**.

−norv Don't use reverse video. This is available to override the reverseVideo resource if set to True.

−prog *program*

Receive message text from the running program *program*. If there are arguments to *program*, encase them with the name of the program in quotes (e.g. xnlock -t "fortune -o"). The resource for this is **xnlock.program**.

RESOURCES

xnlock.font: fontname
xnlock.foreground: color
xnlock.background: color
xnlock.reverseVideo: True/False
xnlock.text: Some random text string
xnlock.program: program [args]
xnlock.ignorePasswd: True/False
xnlock.acceptRootPasswd: True/False

FILES

xnlock executable file
~/.msgfile default message file

AUTHOR

Dan Heller <argv@sun.com> Copyright (c) 1985, 1990.
The original version of this program was written using pixrects on a Sun 2 running SunOS 1.1.

NAME

xstr — extract strings from C programs to implement shared strings

SYNOPSIS

xstr [**-cv**] [**-l** *array*] [**-**] [*file ...*]

DESCRIPTION

xstr maintains a file `strings` into which strings in component parts of a large program are hashed. These strings are replaced with references to this common area. This serves to implement shared constant strings, most useful if they are also read-only.

Available options:

- **xstr** reads from the standard input.

-c **xstr** will extract the strings from the C source *file* or the standard input (**-**), replacing string references by expressions of the form (&xstr[number]) for some number. An appropriate declaration of **xstr** is prepended to the file. The resulting C text is placed in the file `x.c`, to then be compiled. The strings from this file are placed in the `strings` data base if they are not there already. Repeated strings and strings which are suffixes of existing strings do not cause changes to the data base.

-l *array* Specify the named array in program references to abstracted strings. The default array name is xstr.

-v Be verbose.

After all components of a large program have been compiled, a file `xs.c` declaring the common **xstr** space can be created by a command of the form:

```
$ xstr
```

The file `xs.c` should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared) saving space and swap overhead.

xstr can also be used on a single file. The following command creates files `x.c` and `xs.c` as before, without using or affecting any `strings` file in the same directory:

```
$ xstr name
```

It may be useful to run **xstr** after the C preprocessor if any macro definitions yield strings or if there is conditional code which contains strings which may not, in fact, be needed. An appropriate command sequence for running **xstr** after the C preprocessor is:

```
$ cc -E name.c | xstr -c -
$ cc -c x.c
$ mv x.o name.o
```

xstr does not touch the file `strings` unless new items are added, thus `make`(1) can avoid remaking `xs.o` unless truly necessary.

FILES

`strings`	Data base of strings
`x.c`	Massaged C source
`xs.c`	C source for definition of array ‘xstr’
`/tmp/xs*`	Temp file when ‘xstr name’ doesn’t touch `strings`

SEE ALSO

`mkstr`(1)

HISTORY

The **`xstr`** command appeared in 3.0 BSD.

BUGS

If a string is a suffix of another string in the data base, but the shorter string is seen first by **`xstr`** both strings will be placed in the data base, when just placing the longer one there will do.

`xstr` does not parse the file properly so it does not know not to process:

```
char var[] = "const";
```

into:

```
char var[] = (&xstr[N]);
```

These must be changed manually into an appropriate initialization for the string, or use the following ugly hack.

Also, **`xstr`** cannot initialize structures and unions that contain strings. Those can be fixed by changing from:

```
struct foo {
        int i;
        char buf[10];
} = {
        1, "foo"
};
```

to:

```
struct foo {
        int i;
        char buf[10];
} = {
        1, { 'f', 'o', 'o', '\0' }
};
```

The real problem in both cases above is that the compiler knows the size of the literal constant so that it can perform the initialization required, but when **`xstr`** changes the literal string to a pointer reference, the size information is lost. It would require a real parser to do this right, so the obvious solution is to fix the program manually to compile, or even better rely on the compiler and the linker to merge strings appropriately.

Finally, **`xstr`** is not very useful these days because most of the string merging is done automatically by the compiler and the linker, provided that the strings are identical and read-only.

NAME

yacc — an LALR(1) parser generator

SYNOPSIS

yacc [**-dlrtv**] [**-b** *prefix*] [**-o** *outputfile*] [**-p** *symbol_prefix*] *filename*

DESCRIPTION

yacc reads the grammar specification in the file *filename* and generates an LR(1) parser for it. The parsers consist of a set of LALR(1) parsing tables and a driver routine written in the C programming language. **yacc** normally writes the parse tables and the driver routine to the file `y.tab.c`.

The following options are available:

-b *prefix* The **-b** option changes the prefix prepended to the output file names to the string denoted by *prefix*. The default prefix is the character *y*.

-d The **-d** option causes the header file `y.tab.h` to be written.

-l If the **-l** option is not specified, **yacc** will insert #line directives in the generated code. The #line directives let the C compiler relate errors in the generated code to the user's original code. If the **-l** option is specified, **yacc** will not insert the #line directives. #line directives specified by the user will be retained.

-o *outputfile*
The **-o** option specifies an explicit output file name should be used instead of the default.

-p *symbol_prefix*
The **-p** option changes the prefix prepended to yacc-generated symbols to the string denoted by *symbol_prefix*. The default prefix is the string *yy*.

-r The **-r** option causes **yacc** to produce separate files for code and tables. The code file is named `y.code.c`, and the tables file is named `y.tab.c`.

-t The **-t** option changes the preprocessor directives generated by **yacc** so that debugging statements will be incorporated in the compiled code.

-v The **-v** option causes a human-readable description of the generated parser to be written to the file `y.output`.

ENVIRONMENT

The following environment variable is referenced by **yacc**:

`TMPDIR` If the environment variable `TMPDIR` is set, the string denoted by `TMPDIR` will be used as the name of the directory where the temporary files are created.

TABLES

The names of the tables generated by this version of **yacc** are "yylhs", "yylen", "yydefred", "yydgoto", "yysindex", "yyrindex", "yygindex", "yytable", and "yycheck". Two additional tables, "yyname" and "yyrule", are created if `YYDEBUG` is defined and non-zero.

FILES

`y.code.c`
`y.tab.c`
`y.tab.h`

```
y.output
/tmp/yacc.aXXXXXX
/tmp/yacc.tXXXXXX
/tmp/yacc.uXXXXXX
```

DIAGNOSTICS

If there are rules that are never reduced, the number of such rules is written to the standard error. If there are any LALR(1) conflicts, the number of conflicts is also written to the standard error.

STANDARDS

The **yacc** utility conforms to IEEE Std 1003.2 (“POSIX.2”).

NAME

yes — be repetitively affirmative

SYNOPSIS

yes [*expletive*]

DESCRIPTION

yes outputs *expletive*, or, by default, “y”, forever.

HISTORY

The **yes** command appeared in Version 7 AT&T UNIX.

NAME

ypcat — print the values of all keys in a NIS database

SYNOPSIS

ypcat [**-kt**] [**-d** *domainname*] *mapname*
ypcat -x

DESCRIPTION

ypcat prints out the values of all keys from the NIS database specified by *mapname*, which may be a map name or a map nickname.

The options are as follows:

-d *domainname*
: Specify a domain other than the default domain.

-k
: Display map keys. This option is useful with maps in which the values are null or the key is not part of the value.

-t
: Inhibit translation of map nicknames to their corresponding map names.

-x
: Display the map nickname table.

SEE ALSO

domainname(1), ypmatch(1), ypwhich(1), nis(8), ypbind(8), yppoll(8), ypset(8)

AUTHORS

Theo De Raadt

NAME

ypmatch — print the values of one or more keys in a NIS database

SYNOPSIS

ypmatch [**-ktz**] [**-d** *domainname*] *key ... mapname*
ypmatch -x

DESCRIPTION

ypmatch prints out the values of one or more keys from the NIS database specified by *mapname*, which may be a map name or a map nickname.

The options are as follows:

-d *domainname*
Specify a domain other than the default domain.

-k Print the map key followed by a colon, before printing key value. This option is useful with maps in which the values are null or the key is not part of the value.

-t Inhibit translation of map nicknames to their corresponding map names.

-x Display the map nickname table.

-z Append a NUL character at the end of the key when doing the lookup. This is useful for mail.aliases maps, which include a trailing NUL in the keys.

SEE ALSO

domainname(1), ypcat(1), ypwhich(1), nis(8), ypbind(8), yppoll(8), ypset(8)

AUTHORS

Theo De Raadt

NAME

yppasswd — modify a user's NIS password

SYNOPSIS

yppasswd [*user*]

DESCRIPTION

yppasswd changes the user's NIS password.

The **yppasswd** command is deprecated. See passwd(1) for more information.

NAME

ypwhich — return hostname of NIS server of map master

SYNOPSIS

ypwhich [**-d** *domain*] [[**-h**] *host*] [**-T**]
ypwhich [**-d** *domain*] [**-h** *host*] [**-f**] [**-t**] [**-T**] **-m** [*mname*]
ypwhich -x

DESCRIPTION

ypwhich tells which NIS server supplies NIS services to a client, or which is the master for a map. If invoked without arguments, it gives the NIS server for the local machine. If *host* is specified, that machine is queried to find out which NIS server it is using.

The options are as follows:

-d *domain*
: Specify a domain other than the default domain.

-h *host*
: Specify a host other than localhost to query for information.

-f
: When used in conjunction with **-m**, force **ypwhich** to query *host* directly, without using the local copy of ypbind(8).

-t
: Inhibit translation of map nicknames to their corresponding map names.

-T
: Use TCP protocol instead of UDP.

-m [*mname*]
: Find the master NIS server for the named map. *mname* can be a map name or nickname. If *mname* is omitted, **ypwhich** will produce a list of available maps.

-x
: Display the map nickname table.

ypwhich exits with a non-zero exit code if **-m** is used, and there was a problem in determining the map's master.

SEE ALSO

domainname(1), ypcat(1), ypmatch(1), nis(8), ypbind(8), yppoll(8), ypset(8)

AUTHORS

Charles D. Cranor

NAME

zcmp, **zdiff** — compare compressed files

SYNOPSIS

zcmp [*options*] *file* [*file2*]
zdiff [*options*] *file* [*file2*]

DESCRIPTION

zcmp and **zdiff** are filters that invoke cmp(1) or diff(1) respectively to compare compressed files. Such files generally have a "Z" or "gz" extension (both the compress(1) and gzip(1) formats are supported). Any *options* that are specified are passed to cmp(1) or diff(1).

If only *file1* is specified, it is compared against a file with the same name, but with the extension removed. When both *file1* or *file2* are specified, either file may be compressed.

ENVIRONMENT

TMPDIR Directory in which to place temporary files. If unset, /tmp is used.

FILES

/tmp/zcmp.XXXXXXXXXX Temporary file for **zcmp**.
/tmp/zdiff.XXXXXXXXXX Temporary file for **zdiff**.

SEE ALSO

cmp(1), compress(1), diff(1)

CAVEATS

zcmp and **zdiff** rely solely on the file extension to determine what is, or is not, a compressed file. Consequently, the following are not supported as arguments:

- directories
- device special files
- filenames indicating the standard input ("−")

NAME

zforce — force gzip files to have a .gz suffix

SYNOPSIS

zforce *file ...*

DESCRIPTION

The **zforce** utility renames gzip(1) files to have a '.gz' suffix, so that gzip(1) will not compress them twice. This can be useful if file names were truncated during a file transfer. Files that have an existing '.gz', '-gz', '_gz', '.tgz' or '.taz' suffix, or that have not been compressed by gzip(1), are ignored.

SEE ALSO

gzip(1)

CAVEATS

zforce overwrites existing files without warning.

NAME

zgrep, **zegrep**, **zfgrep** — print lines matching a pattern in gzip-compressed files

SYNOPSIS

zgrep [*grep-flags*] [**--**] *pattern* [*files ...*]

zegrep [*grep-flags*] [**--**] *pattern* [*file ...*]

zfgrep [*grep-flags*] [**--**] *pattern* [*file ...*]

DESCRIPTION

zgrep runs grep(1) on *files* or stdin, if no *files* argument is given, after decompressing them with zcat(1).

The *grep-flags* and *pattern* arguments are passed on to grep(1). If an **-e** flag is found in the *grep-flags*, **zgrep** will not look for a *pattern* argument.

zegrep calls egrep(1), while **zfgrep** calls fgrep(1).

EXIT STATUS

In case of missing arguments or missing pattern, 1 will be returned, otherwise 0.

SEE ALSO

egrep(1), fgrep(1), grep(1), gzip(1), zcat(1)

AUTHORS

Thomas Klausner ⟨wiz@NetBSD.org⟩

NAME

zmore — view compressed files on a CRT

SYNOPSIS

zmore [*flags*] [*file ...*]

DESCRIPTION

zmore is a filter that allows the viewing of files compressed with Lempel-Ziv encoding. Such files generally have a "Z" or "gz" extension (both the compress(1) and gzip(1) formats are supported). Any *flags* that are specified are passed to the user's preferred PAGER (which is /usr/bin/more by default).

When multiple files are specified, **zmore** will pause at the end of each file and present the following prompt to the user:

```
prev_file (END) - Next: next_file
```

Where **prev_file** is the file that was just displayed and **next_file** is the next file to be displayed. The following keys are recognized at the prompt:

e or **q** quit **zmore**.

s skip the next file (or exit if the next file is the last).

If no files are specified, **zmore** will read from the standard input. In this mode **zmore** will assume gzip(1) style compression since there is no suffix on which to make a decision.

ENVIRONMENT

PAGER Program used to display files. If unset, /usr/bin/more is used.

SEE ALSO

compress(1), less(1), more(1)

NAME

znew — convert compressed files to gzipped files

SYNOPSIS

znew [**-ftv9K**] *file ...*

DESCRIPTION

The **znew** utility uncompresses files compressed by `compress`(1) and recompresses them with `gzip`(1).

The options are as follows:

-f Overwrite existing '.gz' files. Unless this option is specified, **znew** refuses to overwrite existing files.

-t Test integrity of the gzipped file before deleting the original file. If the integrity check fails, the original '.Z' file is not removed.

-v Print a report specifying the achieved compression ratios.

-9 Use the -9 mode of `gzip`(1), achieving better compression at the cost of slower execution.

-K Keep the original '.Z' file if it uses less disk blocks than the gzipped one. A disk block is 1024 bytes.

SEE ALSO

`gzip`(1)

CAVEATS

The **znew** utility tries to maintain the file mode of the original file. If the original file is not writable, it is not able to do that and **znew** will print a warning.

NAME

zone2ldap /- Load BIND 9 Zone files into LDAP Directory

SYNOPSIS

zone2ldap [-D Bind DN] [-w Bind Password] [-b Base DN] [-z Zone] [-f Zone File] [-h Ldap Host] [-cd] [-v]

DESCRIPTION

zone2ldap will parse a complete BIND 9 format DNS zone file, and load the contents into an LDAP directory, for use with the LDAP sdb back-end.

If the zone already exists, zone2ldap will exit succesfully. If the zone does not exists, or partially exists, zone2ldap will attempt to add all/missing zone data.

Options

-b LDAP Base DN. LDAP systems require a "base dn", which is generally considered the LDAP Directory root. If the zone you are loading is different from the base, then you will need to tell zone2ldap what your LDAP base is.

-v Print version information, and immediatly exit.

-f Zone file. Bind 9.1 compatible zone file, from which zone information will be read.

-d Dump debug information to standard out.

-w LDAP Bind password, corresponding the the value of "-b".

-h LDAP Directory host. This is the hostname of the LDAP system you wish to store zone information on. An LDAP server should be listening on port 389 of the target system. This may be ommited, and will default to "localhost".

-c This will create the zone portion of the DN you are importing. For instance, if you are creating a domain.com zone, zone2ldap should first create "dc=domain,dc=com". This is useful if you are creating multiple domains.

-z This is the name of the zone specified in the SOA record.

EXAMPLES

Following are brief examples of how to import a zone file into your LDAP DIT.

Loading zone domain.com, with an LDAP Base DN of dc=domain,dc=com

zone2ldap -D dc=root -w secret -h localhost -z domain.com -f domain.com.zone

This will add Resource Records into an ALREADY EXISTING dc=domain,dc=com. The final SOA DN in this case, will be dc=@,dc=domain,dc=com

Loading customer.com, if your LDAP Base DN is dc=provider,dc=net.

zone2ldap -D dc=root -w secret -h localhost -z customer.com -b dc=provider,dc=net -f customer.com.zone -c

This will create dc=customer,dc=com under dc=provider,dc=net, and add all necessary Resource Records. The final root DN to the SOA will be dc=@,dc=customer,dc=com,dc=provider,dc=net.

SEE ALSO

named(8) ldap(3) http://www.venaas.no/ldap/bind-sdb/

BUGS

Send all bug reports to Jeff McNeil <jeff@snapcase.g-rock.net>

AUTHOR

Jeff McNeil <jeff@snapcase.g-rock.net>