NAME

intro — introduction to system maintenance procedures and commands

DESCRIPTION

This section contains information related to system operation and maintenance.

It describes commands used to create new file systems (newfs(8)), verify the integrity of the file systems (fsck(8)), control disk usage (edquota(8)), maintain system backups (dump(8)), and recover files when disks die an untimely death (restore(8)). The format(8) manual for the specific architecture the system is running on should be consulted when formatting disks and tapes. Network related services like inetd(8) and ftpd(8) are also described.

A number of pages in this section describe general system management topics. For example, the diskless(8) page describes how to boot a system over a network, and the compat_linux(8) page describes how to run Linux binaries on NetBSD architectures that support it.

HISTORY

The **intro** section manual page appeared in 4.2BSD.

NAME

MAKEDEV — create system and device special files

SYNOPSIS

MAKEDEV [**-fMs**] [**-m** *mknod*] [**-p** *pax*] [**-t** *mtree*] {*special* | *device*} [*...*]

DESCRIPTION

MAKEDEV is used to create system and device special files. As arguments it takes the names of known devices, like *sd0*, or of special targets, like `all` or `std`, which create a collection of device special files, or `local`, which invokes `MAKEDEV.local`(8) with the `all` argument.

The script is in `/dev/MAKEDEV`. Devices are created in the current working directory; in normal use, **MAKEDEV** should be invoked with `/dev` as the current working directory.

Supported options are:

-f Force permissions to be updated on existing devices. This works only if **MAKEDEV** invokes `mknod`(8); it is not compatible with the **-p**, **-s**, or **-t** options.

-M Create a memory file system, union mounted over the current directory, to contain the device special files. The memory file system is created using `mount_tmpfs`(8) or `mount_mfs`(8), in that order of preference.

If the **-M** flag is specified more than once, then **MAKEDEV** assumes that it is being invoked from `init`(1) to populate a memory file system for `/dev`. In this case, **MAKEDEV** will also redirect its output to the system console.

-m *mknod* Force the use of `mknod`(8), and specify the name or path to the `mknod`(8) program. [Usually, $TOOL_MKNOD or mknod.]

-p *pax* Force the use of `pax`(1), and specify the name or path to the `pax`(1) program. [Usually, $TOOL_PAX or pax.]

-s Generate an `mtree`(8) specfile instead of creating devices.

-t *mtree* Force the use of `mtree`(8), and specify the name or path to the `mtree`(8) program. [Usually, $TOOL_MTREE or mtree.]

MAKEDEV has several possible methods of creating device nodes:

- By invoking the `mknod`(8) command once for each device node. This is the traditional method, but it is slow because each device node is created using a new process.

  The **-m** option forces **MAKEDEV** to use the `mknod`(8) method.

- By internally creating a specfile in a format usable by `mtree`(8), and providing the specfile on standard input to a `pax`(1) or `mtree`(8) command, invoked with options that request it to create the device nodes as well as any necessary subdirectories. This is much faster than creating device nodes with `mknod`(8), because it requires much fewer processes; however, it's not compatible with the **-f** option.

  The **-p** or **-t** options force **MAKEDEV** to use the `pax`(1) or `mtree`(8) methods.

- If the **-s** option is specified, then **MAKEDEV** will not create device nodes at all, but will output a specfile in a format usable by `mtree`(8).

The **-m**, **-p**, **-s**, and **-t** flags are mutually exclusive. If none of these flags is specified, then **MAKEDEV** will use `mtree`(8), `pax`(1), or `mknod`(8), in that order of preference, depending on which commands appear to be available and usable. In normal use, it's expected that `mtree`(8) will be available, so it will be chosen. If **MAKEDEV** is invoked by `init`(8), it's expected that `mtree`(8) will not be available, but `pax`(1) may be available.

The special targets supported on NetBSD are:

all	Makes all known devices, including local devices. Tries to make the 'standard' number of each type.
init	A set of devices that is used for MFS /dev by init. May be equal to "all".
floppy	Devices to be put on install floppies
ramdisk	Devices to be put into INSTALL kernel ramdisks.
std	Standard devices
local	Configuration specific devices
wscons	Make wscons devices
usbs	Make USB devices
isdns	Make ISDN devices

Please note that any hash marks ("#") in the following list of supported device targets must be replaced by digits when calling **MAKEDEV**:

Tapes:

st#	SCSI tapes, see st(4)
wt#	QIC-interfaced (e.g. not SCSI) 3M cartridge tape, see wt(4)
ht#	MASSBUS TM03 and TU??, see vax/ht(4)
mt#	MSCP tapes (e.g. TU81, TK50), see vax/mt(4)
tm#	UNIBUS TM11 and TE10 emulations (e.g. Emulex TC-11), see vax/tm(4)
ts#	UNIBUS TS11, see vax/ts(4)
ut#	UNIBUS TU45 emulations (e.g. si 9700), see vax/ut(4)
uu#	TU58 cassettes on DL11 controller, see vax/uu(4)

Disks:

ccd#	Concatenated disk devices, see ccd(4)
cd#	SCSI or ATAPI CD-ROM, see cd(4)
cgd#	Cryptographic disk devices, see cgd(4)
raid#	RAIDframe disk devices, see raid(4)
sd#	SCSI disks, see sd(4)
wd#	"winchester" disk drives (ST506,IDE,ESDI,RLL,...), see wd(4)
bmd#	Nereid bank memory disks, see x68k/bmd(4)
ed#	IBM PS/2 ESDI disk devices, see edc(4)
fd#	"floppy" disk drives (3 1/2", 5 1/4"), see amiga/fdc(4), i386/fdc(4), sparc64/fdc(4)
fss#	Files system snapshot devices, see fss(4)
gdrom#	Dreamcast "gigadisc" CD-ROM drive, see dreamcast/gdrom(4)
hk#	UNIBUS RK06 and RK07, see vax/hk(4)
hp#	MASSBUS RM??, see vax/hp(4)
ld#	Logical disk devices (e.g., hardware RAID), see ld(4)
mcd#	Mitsumi CD-ROM, see mcd(4)
md#	Memory pseudo-disk devices, see md(4)
ofdisk#	OpenFirmware disk devices
ra#	MSCP disks (RA??, RD??)
rb#	730 IDC w/ RB80 and/or RB02
rd#	HDC9224 RD disks on VS2000, see hp300/rd(4)
rl#	UNIBUS RL02, see vax/rl(4)
rx#	MSCP floppy disk (RX33/50/...)
up#	Other UNIBUS devices (e.g. on Emulex SC-21V controller), see vax/up(4)

vnd#	``file'' pseudo-disks, see `vnd`(4)
xbd#	Xen virtual disks
xd#	Xylogic 753/7053 disks, see `sparc/xd`(4)
xy#	Xylogic 450/451 disks, see `sparc/xy`(4)

Pointing devices:

wsmouse#	wscons mouse events, see `wsmouse`(4)
lms#	Logitech bus mouse, see `i386/lms`(4)
mms#	Microsoft bus mouse, see `dreamcast/mms`(4), `i386/mms`(4)
qms#	``quadrature mouse'', see `acorn32/qms`(4)
pms#	PS/2 mouse
mouse	Mouse (provides events, for X11)

Keyboard devices:

wskbd#	wscons keyboard events, see `wskbd`(4)
kbd	Raw keyboard (provides events, for X11), see `sparc/kbd`(4), `sun2/kbd`(4), `sun3/kbd`(4)
kbdctl	Keyboard control

Terminals/Console ports:

tty[01]#	Standard serial ports, see `tty`(4)
tty0#	SB1250 (``sbscn'') serial ports (sbmips), see `tty`(4)
ttyE#	wscons - Workstation console (``wscons'') glass-tty emulators
ttyCZ?	Cyclades-Z multiport serial boards. Each ``unit'' makes 64 ports., see `cz`(4)
ttyCY?	Cyclom-Y multiport serial boards. Each ``unit'' makes 32 ports., see `cy`(4)
ttye#	ITE bitmapped consoles, see `amiga/ite`(4), `hp300/ite`(4)
ttyv0	pccons
ttyC?	NS16550 (``com'') serial ports
ttyS#	SA1110 serial port (hpcarm)
ttyTX?	TX39 internal serial ports (hpcmips)
ttyB?	DEC 3000 ZS8530 (``scc'') serial ports (alpha), see `scc`(4)
ttyA#	Mfc serial ports (amiga)
ttyB#	Msc serial ports (amiga)
ttyC#	Com style serial ports (DraCo, HyperCom) (amiga) On the DraCo, units 0 and 1 are the built-in ``modem'' and ``mouse'' ports, if configured.
ttyA0	8530 Channel A (formerly ser02) (atari)
ttyA1	8530 Channel B (formerly mdm02) (atari)
ttyB0	UART on first 68901 (formerly mdm01) (atari)
ixpcom	IXP12x0 COM ports
epcom	EP93xx COM ports
ttyM?	HP200/300 4 port serial mux interface (hp300)
ttya	``ttya'' system console (luna68k)
ttyb	Second system serial port (luna68k)
tty#	Onboard serial ports (mvme68k) On the mvme147 these are: ttyZ1, ttyZ2 and ttyZ3. On the mvme167, and '177: ttyC1, ttyC2 and ttyC3. Note that tty[CZ]0 is grabbed by the console device so is not created by default, see `tty`(4)
dc#	PMAX 4 channel serial interface (kbd, mouse, modem, printer)
scc#	82530 serial interface (pmax), see `scc`(4)
ttyZ#	Zilog 8530 (``zstty'') serial ports, see `zstty`(4)
tty[abcd]	Built-in serial ports (sparc)
tty#	Z88530 serial controllers (sparc64), see `tty`(4)

ttyh#	SAB82532 serial controllers (sparc64), see `sparc64/sab`(4)
tty[a-j]	Built-in serial ports (sun2, sun3)
ttyC?	pccons (arc)
dz#	UNIBUS DZ11 and DZ32 (vax), see `vax/dz`(4)
dh#	UNIBUS DH11 and emulations (e.g. Able DMAX, Emulex CS-11) (vax), see `vax/dh`(4)
dmf#	UNIBUS DMF32 (vax), see `vax/dmf`(4)
dhu#	UNIBUS DHU11 (vax), see `vax/dhu`(4)
dmz#	UNIBUS DMZ32 (vax), see `vax/dmz`(4)
dl#	UNIBUS DL11 (vax), see `vax/dl`(4)
xencons	Xen virtual console

Terminal multiplexors:

dc#	4 channel serial interface (keyboard, mouse, modem, printer)
dh#	UNIBUS DH11 and emulations (e.g. Able DMAX, Emulex CS-11), see `vax/dh`(4)
dhu#	UNIBUS DHU11, see `vax/dhu`(4)
dl#	UNIBUS DL11, see `vax/dl`(4)
dmf#	UNIBUS DMF32, see `vax/dmf`(4)
dmz#	UNIBUS DMZ32, see `vax/dmz`(4)
dz#	UNIBUS DZ11 and DZ32, see `vax/dz`(4)
scc#	82530 serial interface, see `scc`(4)

Call units:

dn#	UNIBUS DN11 and emulations (e.g. Able Quadracall), see `vax/dn`(4)

Pseudo terminals:

ptm	Pty multiplexor device., see `ptm`(4)
pty#	Set of 16 master and slave pseudo terminals, see `pty`(4)
opty	First 16 ptys, to save inodes on install media
ipty	First 2 ptys, for install media use only

Printers:

arcpp#	Archimedes parallel port
lpt#	Stock lp, see `lpt`(4), `acorn32/lpt`(4), `i386/lpt`(4), `mvme68k/lpt`(4)
lpa#	Interruptless lp
par#	Amiga motherboard parallel port
cpi#	Macintosh Nubus CSI parallel printer card

USB devices:

usb#	USB control devices, see `usb`(4)
uhid#	USB generic HID devices, see `uhid`(4)
ulpt#	USB printer devices, see `ulpt`(4)
ugen#	USB generic devices, see `ugen`(4)
urio#	USB Diamond Rio 500 devices, see `urio`(4)
uscanner#	USB scanners, see `uscanner`(4)
ttyU#	USB modems, see `ucom`(4)
ttyY#	USB serial adapters

ISDN devices:

isdn	Communication between userland isdnd and kernel, see `isdn`(4)
isdnctl	Control device, see `isdnctl`(4)
isdnbchan#	Raw b-channel access, see `isdnbchan`(4)

isdntel#	Telephony device, see isdntel(4)
isdnteld#	Telephony dialout device
isdntrc#	Trace device, see isdntrc(4)

Video devices:

bwtwo#	Monochromatic frame buffer, see sparc/bwtwo(4), sun2/bwtwo(4), sun3/bwtwo(4)
cgtwo#	8-bit color frame buffer, see sparc/cgtwo(4), sun3/cgtwo(4)
cgthree#	8-bit color frame buffer, see sparc/cgthree(4)
cgfour#	8-bit color frame buffer, see sparc/cgfour(4), sun3/cgfour(4)
cgsix#	Accelerated 8-bit color frame buffer, see sparc/cgsix(4)
cgeight#	24-bit color frame buffer, see sparc/cgeight(4)
etvme	Tseng et-compatible cards on VME (atari)
ik#	UNIBUS interface to Ikonas frame buffer, see vax/ik(4)
leo	Circad Leonardo VME-bus true color (atari)
ps#	UNIBUS interface to Picture System 2, see vax/ps(4)
qv#	QVSS (MicroVAX) display
tcx#	Accelerated 8/24-bit color frame buffer, see sparc/tcx(4)

Maple bus devices:

maple	Maple bus control devices, see dreamcast/maple(4)
mlcd#	Maple bus LCD devices, see dreamcast/mlcd(4)
mmem#	Maple bus storage devices, see dreamcast/mmem(4)

IEEE1394 bus devices:

fw#	IEEE1394 bus generic node access devices
fwmem#	IEEE1394 bus physical memory of the remote node access devices

Special purpose devices:

ad#	UNIBUS interface to Data Translation A/D converter, see vax/ad(4)
agp#	AGP GART devices, see agp(4)
altq	ALTQ control interface
amr#	AMI MegaRaid control device, see amr(4)
apm	Power management device, see i386/apm(4)
audio#	Audio devices, see audio(4)
bell#	OPM bell device (x68k)
bktr	Brooktree 848/849/878/879 based TV cards, see bktr(4)
bpf	Packet filter, see bpf(4)
bthub	Bluetooth Device Hub control interface, see bthub(4)
cfs#	Coda file system device
ch#	SCSI media changer, see ch(4)
cir#	Consumer IR, see cir(4)
clockctl	Clock control for non root users, see clockctl(4)
cpuctl	CPU control
crypto	Hardware crypto access driver, see crypto(4)
dmoverio	Hardware-assisted data movers, see dmoverio(4)
dpt#	DPT/Adaptec EATA RAID management interface, see dpt(4)
dpti#	DPT/Adaptec I2O RAID management interface, see dpti(4)
fb#	PMAX generic framebuffer pseudo-device
fd	File descriptors
grf#	Graphics frame buffer device, see amiga/grf(4), hp300/grf(4)

hil	HP300 HIL input devices, see `hp300/hil`(4)
icp	ICP-Vortex/Intel RAID control interface, see `icp`(4)
iic#	IIC bus device
io	X86 IOPL access for COMPAT_10, COMPAT_FREEBSD, see `hp700/io`(4), `i386/io`(4)
iop#	I2O IOP control interface, see `iop`(4)
ipl	IP Filter
irframe#	IrDA physical frame, see `irframe`(4)
ite#	Terminal emulator interface to HP300 graphics devices, see `amiga/ite`(4), `hp300/ite`(4)
joy#	Joystick device, see `joy`(4)
kttcp	Kernel ttcp helper device, see `kttcp`(4)
lkm	Loadable kernel modules interface, see `lkm`(4)
lockstat	Kernel locking statistics
magma#	Magma multiport serial/parallel cards, see `sparc/magma`(4)
midi#	MIDI, see `midi`(4)
mlx#	Mylex DAC960 control interface, see `mlx`(4)
mly#	Mylex AcceleRAID/eXtremeRAID control interface, see `mly`(4)
np#	UNIBUS Ethernet co-processor interface, for downloading., see `vax/np`(4)
nsmb#	SMB requester, see `nsmb`(4)
openfirm	OpenFirmware accessor
pad#	Pseudo-audio device driver, see `pad`(4)
pci#	PCI bus access devices, see `pci`(4)
pf	PF packet filter
pow#	Power management device (x68k), see `x68k/pow`(4)
putter	Pass-to-Userspace Transporter
px#	PixelStamp Xserver access, see `px`(4)
radio#	Radio devices, see `radio`(4)
random	Random number generator, see `rnd`(4)
rtc#	RealTimeClock, see `atari/rtc`(4), `evbppc/rtc`(4), `hp300/rtc`(4)
satlink#	PlanetConnect satellite receiver driver
scsibus#	SCSI busses, see `scsi`(4)
se#	SCSI Ethernet, see `se`(4)
ses#	SES/SAF-TE SCSI Devices, see `ses`(4)
speaker	PC speaker, see `speaker`(4)
sram	Battery backuped memory (x68k)
ss#	SCSI scanner, see `ss`(4)
stic#	PixelStamp interface chip
sysmon	System Monitoring hardware, see `envsys`(4)
tap#	Virtual Ethernet device, see `tap`(4)
tun#	Network tunnel driver, see `tun`(4)
twa	3ware Apache control interface, see `twa`(4)
twe	3ware Escalade control interface, see `twe`(4)
uk#	Unknown SCSI device, see `uk`(4)
veriexec	Verified executable fingerprint loader, see `veriexec`(4)
vmegen#	Generic VME access
view#	Generic interface to graphic displays (Amiga)
wsfont#	Console font control, see `wsfont`(4)
wsmux#	wscons event multiplexor, see `wsmux`(4)

xenevt	Xen event interface

ENVIRONMENT

The following environment variables affect the execution of **MAKEDEV**:

`MAKEDEV_AS_LIBRARY`
: If this is set, then **MAKEDEV** will define several shell functions and then return, ignoring all its command line options and arguments. This is used to enable `MAKEDEV.local`(8) to use the shell functions defined in **MAKEDEV**.

FILES

`/dev`	special device files directory
`/dev/MAKEDEV`	script described in this man page
`/dev/MAKEDEV.local`	script for site-specific devices

DIAGNOSTICS

If the script reports an error that is difficult to understand, you can get more debugging output by using

sh -x *MAKEDEV argument*.

SEE ALSO

`config`(1), `init`(1), `pax`(1), `intro`(4), `MAKEDEV.local`(8), `diskless`(8), `mknod`(8), `mount_mfs`(8), `mount_tmpfs`(8), `mtree`(8)

HISTORY

The **MAKEDEV** command appeared in 4.2BSD. The **-f**, **-m**, and **-s** options were added in NetBSD 2.0. The **-p**, **-t**, and **-M** options were added in NetBSD 5.0. The ability to be used as a function library was added in NetBSD 5.0.

BUGS

The **-f** option is not compatible with the use of `mtree`(8) or `pax`(1).

NOTES

Not all devices listed in this manpage are supported on all platforms.

This man page is generated automatically from the same sources as `/dev/MAKEDEV`, in which the device files are not always sorted, which may result in an unusual (non-alphabetical) order.

In order to allow a diskless NetBSD client to obtain its `/dev` directory from a file server running a foreign operating system, one of the following techniques may be useful to populate a directory of device nodes on the foreign server:

- If the foreign server is sufficiently similar to NetBSD, run **MAKEDEV** in an appropriate directory of the foreign server, using the **-m** flag to refer to a script that converts from command line arguments that would be usable with the NetBSD `mknod`(8) command to the equivalent commands for the foreign server.
- Run **MAKEDEV** with the **-s** flag to generate an `mtree`(8) specification file; this can be done on any host with a POSIX-compliant shell and a few widely-available utilities. Use the `pax`(1) command with the **-w -M** flags to convert the `mtree`(8) specification file into an archive in a format that supports device nodes (such as *ustar* format); this can be done on a NetBSD host, or can be done in a cross-build environment using **TOOLDIR**`/bin/nbpax`. Finally, use apropriate tools on the foreign server to unpack the archive and create the device nodes.

NAME

MAKEDEV.local — create site-specific device special files

SYNOPSIS

MAKEDEV.local [**-fMs**] [**-m** *mknod*] [**-p** *pax*] [**-t** *mtree*] {all |
    site-specific-argument} [*...*]

DESCRIPTION

MAKEDEV.local is used to create site-specific device special files. Each argument may be the word `all` or a site-specific argument. By default, there are no valid site-specific arguments, and the `all` argument has no effect; This may be changed by editing the script.

The script is in `/dev/MAKEDEV.local`. Devices are created in the current working directory; in normal use, **MAKEDEV.local** should be invoked with `/dev` as the current working directory.

Supported options for **MAKEDEV.local** are the same as for MAKEDEV(8).

FILES

`/dev`	special device files directory
`/dev/MAKEDEV`	script that invokes **MAKEDEV.local** with the `all` argument.
`/dev/MAKEDEV.local`	script described in this man page

SEE ALSO

config(1), intro(4), MAKEDEV(8), mknod(8)

HISTORY

The **MAKEDEV.local** command appeared in 4.2BSD. Handling of the same command line options as MAKEDEV(8), and the use of MAKEDEV(8) as a function library, was added in NetBSD 5.0.

NOTES

The relationship between **MAKEDEV.local** and MAKEDEV(8) is complex:

- If MAKEDEV(8) is invoked with the `all` or `local` argument, then it will invoke **MAKEDEV.local** as a child process, with options similar to those that were originally passed to MAKEDEV(8), and with the `all` argument.
- **MAKEDEV.local** uses shell functions defined in MAKEDEV(8). This is done by loading MAKEDEV(8) using the shell "." command, with the `MAKEDEV_AS_LIBRARY` variable set (to inform MAKEDEV(8) that it should behave as a function library, not as an independent program).

NAME

ac — display connect time accounting

SYNOPSIS

ac [**-d** | **-p**] [**-t** *tty*] [**-w** *file*] [*users ...*]

DESCRIPTION

If the file /var/log/wtmp exists, a record of individual login and logout times are written to it by login(1) and init(8), respectively. The program **ac** examines these records and writes the accumulated connect time for all logins to the standard output.

Options available:

-d	Display the connect times in 24 hour chunks.
-p	Display individual user totals.
-t *tty*	Only do accounting logins on certain ttys. The *tty* specification can start with '!' to indicate not this *tty* and end with '*' to indicate all similarly named ttys. Multiple **-t** flags may be specified.
-w *file*	Read raw connect time data from *file* instead of the default file /var/log/wtmp.
users ...	Display totals for the given individuals only.

If no arguments are given, **ac** displays the total amount of login time for all active accounts on the system.

The default wtmp file is an infinitely increasing file unless frequently truncated. This is normally done by the daily daemon scripts scheduled by cron(8), which rename and rotate the wtmp files before truncating them (and keep about a week's worth on hand). No login times are collected, however, if the file does not exist.

For example,

```
ac -p -t "ttyd*" > modems
ac -p -t "!ttyd*" > other
```

allows times recorded in modems to be charged out at a different rate than other.

The **ac** utility exits 0 on success, and >0 if a fatal error occurs.

FILES

/var/log/wtmp	connect time accounting file
/var/log/wtmp.[0-7]	rotated files

SEE ALSO

login(1), utmp(5), init(8), sa(8)

HISTORY

An **ac** command appeared in Version 6 AT&T UNIX. This version of **ac** was written for NetBSD 1.0 from the specification provided by various systems' manual pages.

NAME

accton — enable/disable system accounting

SYNOPSIS

accton [*file*]

DESCRIPTION

With an argument naming an existing *file*, **accton** causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

The default accounting file is `/var/account/acct`. Typically, accounting is enabled by rc scripts during the boot process. In NetBSD, one may enable accounting by setting the variable "accounting" to "YES" in `/etc/rc.conf`.

Note that, traditionally, the system accounting log file can not be rotated cleanly by `newsyslog`(8). Instead, a default installation of NetBSD rotates `/var/account/acct` using the `/etc/daily` script.

FILES

`/var/account/acct` Default accounting file.

SEE ALSO

`lastcomm`(1), `acct`(5), `sa`(8)

HISTORY

The **accton** command has existed nearly forever, but this man page is new.

NAME

acpidump — dump ACPI tables

SYNOPSIS

acpidump [**-r**]
acpidump [**-r**] [**-o** *dsdt_file_for_output*]
acpidump [**-r**] [**-f** *dsdt_file_for_input*]

DESCRIPTION

The **acpidump** utility analyzes ACPI tables in physical memory and dumps them to standard output. In addition, **acpidump** can disassemble AML (ACPI Machine Language) found in these tables and dump them as ASL (ACPI Source Language).

ACPI tables have an essential data block (the DSDT, Differentiated System Description Table), that includes information used on the kernel side such as detailed information about PnP hardware, procedures for controlling power management support and so on. The **acpidump** utility can extract the DSDT data block from physical memory and store it into a DSDT data file, and also can generate an output in ASL from a given DSDT data file.

When **acpidump** is invoked without the **-f** option, it will read ACPI tables from physical memory via a special file `/dev/mem` and dump them. First it searches for the RSDP (Root System Description Pointer), which has the signature "RSD PTR ", and then gets the RSDT (Root System Description Table), which includes a list of pointers to physical memory addresses for other tables. The RSDT itself and all other tables linked from RSDT are generically called SDTs (System Description Tables) and their header has a common format which consists of items such as Signature, Length, Revision, Checksum, OEMID, OEM Table ID, OEM Revision, Creator ID and Creator Revision. The **acpidump** utility dumps contents of these SDTs. For further information about formats of each table, see chapter 5: “ACPI Software Programming Model” from the ACPI specifications referenced below.

There is always a pointer to a physical memory address in RSDT for FACP (Fixed ACPI Description Table). The FACP defines static system information about power management support (ACPI Hardware Register Implementation) such as interrupt mode (INT_MODEL), SCI interrupt number, SMI command port (SMI_CMD) and location of ACPI registers. The FACP also has a pointer to a physical memory address for DSDT, which includes information used on the kernel side such as PnP, power management support and so on. While the other tables are described in fixed format, the DSDT consists of AML data which is compiled from sources written in free formated ASL, which is the description language for ACPI. When **acpidump** outputs DSDT, it disassembles the AML data and formats it as ASL.

OPTIONS

The following options are supported by **acpidump**:

-f *dsdt_file_for_input*
Interprets AML data in DSDT from a file specified in *dsdt_file_for_input* and dumps them in ASL to standard output.

-h Displays usage and exits.

-o *dsdt_file_for_output*
Stores DSDT data block from physical memory into a file specified in *dsdt_file_for_output* in addition to behavior with no option.

-r Additionally outputs commented **ResourceTemplate**() macros for Buffer objects that contain valid resource streams. These macros are defined in the ACPI 2.0 specification section 16.2.4.

FILES

`/dev/mem`

EXAMPLES

This is an example to get a dump of SDTs and a DSDT data file simultaneously on a machine that supports ACPI BIOS.

```
# acpidump -o foo.dsdt > foo.asl
```

SEE ALSO

`acpi`(4), `amldb`(8)

Advanced Configuration and Power Interface Specification, Intel Microsoft Toshiba, Revision 1.0b, 2.0,

HISTORY

The **`acpidump`** utility appeared in FreeBSD 5.0.

AUTHORS

Doug Rabson ⟨dfr@FreeBSD.org⟩
Mitsuru IWASAKI ⟨iwasaki@FreeBSD.org⟩
Yasuo YOKOYAMA ⟨yokoyama@jp.FreeBSD.org⟩

Some contributions made by Chitoshi Ohsawa ⟨ohsawa@catv1.ccn-net.ne.jp⟩, Takayasu IWANASHI ⟨takayasu@wendy.a.perfect-liberty.or.jp⟩, Yoshihiko SARUMARU ⟨mistral@imasy.or.jp⟩, Hiroki Sato ⟨hrs@FreeBSD.org⟩, Michael Lucas ⟨mwlucas@blackhelicopters.org⟩ and Michael Smith ⟨msmith@FreeBSD.org⟩.

BUGS

In the current implementation, **`acpidump`** doesn't dump any information of Firmware ACPI Control Structure (FACS) specified by a pointer in FACP.

NAME

afterboot — things to check after the first complete boot

DESCRIPTION

Starting Out

This document attempts to list items for the system administrator to check and set up after the installation and first complete boot of the system. The idea is to create a list of items that can be checked off so that you have a warm fuzzy feeling that something obvious has not been missed. A basic knowledge of UNIX is assumed.

Complete instructions for correcting and fixing items is not provided. There are manual pages and other methodologies available for doing that. For example, to view the man page for the ls(1) command, type:

```
man 1 ls
```

Administrators will rapidly become more familiar with NetBSD if they get used to using the manual pages.

Security alerts

By the time that you have installed your system, it is quite likely that bugs in the release have been found. All significant and easily fixed problems will be reported at http://www.NetBSD.org/support/security/. It is recommended that you check this page regularly.

Login

Login as "**root**". You can do so on the console, or over the network using ssh(1). If you have enabled the ssh daemon and wish to allow root logins over the network, edit the /etc/ssh/sshd_config file and set **PermitRootLogin** to "yes" (see sshd_config(5)). The default is to not permit root logins over the network after fresh install in NetBSD.

Upon successful login on the console, you may see the message "We recommend creating a non-root account...". For security reasons, it is bad practice to login as root during regular use and maintenance of the system. In fact, the system will only let you login as root on a secure terminal. By default, only the console is considered to be a secure terminal. Instead, administrators are encouraged to add a "regular" user, add said user to the "wheel" group, then use the su(1) command when root privileges are required. This process is described in more detail later.

Root password

Change the password for the root user. (Note that throughout the documentation, the term "superuser" is a synonym for the root user.) Choose a password that has numbers, digits, and special characters (not space) as well as from the upper and lower case alphabet. Do not choose any word in any language. It is common for an intruder to use dictionary attacks. Type the command **/usr/bin/passwd** to change it.

It is a good idea to always specify the full path name for both the passwd(1) and su(1) commands as this inhibits the possibility of files placed in your execution PATH for most shells. Furthermore, the superuser's PATH should never contain the current directory ("**.**").

System date

Check the system date with the date(1) command. If needed, change the date, and/or change the symbolic link of /etc/localtime to the correct time zone in the /usr/share/zoneinfo directory.

Examples:

date 200205101820
Set the current date to May 10th, 2002 6:20pm.

ln -fs /usr/share/zoneinfo/Europe/Helsinki /etc/localtime
Set the time zone to Eastern Europe Summer Time.

Console settings

One of the first things you will likely need to do is to set up your keyboard map (and maybe some other aspects about the system console). To change your keyboard encoding, edit the "*encoding*" variable found in `/etc/wscons.conf`.

`wscons.conf`(5) contains more information about this file.

Check hostname

Use the **hostname** command to verify that the name of your machine is correct. See the man page for `hostname`(1) if it needs to be changed. You will also need to change the contents of the "*hostname*" variable in `/etc/rc.conf` or edit the `/etc/myname` file to have it stick around for the next reboot. Note that hostname is supposed include a domainname, and that this should not be confused with YP (NIS) `domainname`(1).

Verify network interface configuration

The first thing to do is an **ifconfig -a** to see if the network interfaces are properly configured. Correct by editing `/etc/ifconfig.`*interface* or the corresponding "*ifconfig_interface*" variable in `rc.conf`(5) (where *interface* is the interface name, e.g., "le0") and then using `ifconfig`(8) to manually configure it if you do not wish to reboot.

You can add new "virtual interfaces" by adding the required entries to `/etc/ifconfig.`*interface*. Read the `ifconfig.if`(5) man page for more information on the format of `/etc/ifconfig.`*interface* files. The loopback interface will look something like:

```
lo0: flags=8009<UP,LOOPBACK,MULTICAST> mtu 32972
        inet 127.0.0.1 netmask 0xff000000
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
        inet6 ::1 prefixlen 128
```

an Ethernet interface something like:

```
le0: flags=9863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
        inet 192.168.4.52 netmask 0xffffff00 broadcast 192.168.4.255
        inet6 fe80::5ef0:f0f0%le0 prefixlen 64 scopeid 0x1
```

and a PPP interface something like:

```
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
        inet 203.3.131.108 --> 198.181.0.253 netmask 0xffff0000
```

See `mrouted`(8) for instructions on configuring multicast routing.

See `dhcpd`(8) for instructions on configuring interfaces with DHCP.

Check routing tables

Issue a **netstat -rn** command. The output will look something like:

```
Routing tables

Internet:
Destination    Gateway          Flags  Refs      Use  Mtu  Interface
default        192.168.4.254    UGS       0 11098028    -  le0
127            127.0.0.1        UGRS      0        0    -  lo0
127.0.0.1      127.0.0.1        UH        3       24    -  lo0
```

```
192.168.4       link#1             UC       0        0    -  le0
192.168.4.52    8:0:20:73:b8:4a    UHL      1     6707    -  le0
192.168.4.254   0:60:3e:99:67:ea   UHL      1        0    -  le0

Internet6:
Destination         Gateway        Flags  Refs  Use     Mtu  Interface
::/96               ::1            UGRS     0     0   32972  lo0 =>
::1                 ::1            UH       4     0   32972  lo0
::ffff:0.0.0.0/96   ::1            UGRS     0     0   32972  lo0
fc80::/10           ::1            UGRS     0     0   32972  lo0
fe80::/10           ::1            UGRS     0     0   32972  lo0
fe80::%le0/64       link#1         UC       0     0    1500  le0
fe80::%lo0/64       fe80::1%lo0    U        0     0   32972  lo0
ff01::/32           ::1            U        0     0   32972  lo0
ff02::%le0/32       link#1         UC       0     0    1500  le0
ff02::%lo0/32       fe80::1%lo0    UC       0     0   32972  lo0
```

The default gateway address is stored in the "*defaultroute*" variable in `/etc/rc.conf`, or in the file `/etc/mygate`. If you need to edit this file, a painless way to reconfigure the network afterwards is to issue

```
/etc/rc.d/network restart
```

Or, you may prefer to manually configure using a series of **route add** and **route delete** commands (see `route`(8)). If you run `dhclient`(8) you will have to kill it by running

```
/etc/rc.d/dhclient stop
```

after you flush the routes.

If you wish to route packets between interfaces, add one or both of the following directives (depending on whether IPv4 or IPv6 routing is required) to `/etc/sysctl.conf`:

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
```

As an alternative, compile a new kernel with the **GATEWAY** option. Packets are not forwarded by default, due to RFC requirements.

Secure Shell (ssh)

By default, all services are disabled in a fresh NetBSD installation, and ssh is no exception. You may wish to enable it so you can remotely control your system. Set "*sshd=yes*" in `/etc/rc.conf` and then starting the server with the command

```
/etc/rc.d/sshd start
```

The first time the server is started, it will generate a new keypair, which will be stored inside the directory `/etc/ssh`.

BIND Name Server (DNS)

If you are using the BIND Name Server, check the `/etc/resolv.conf` file. It may look something like:

```
domain some.thing.dom
nameserver 192.168.0.1
nameserver 192.168.4.55
search some.thing.dom. thing.dom.
```

For further details, see `resolv.conf`(5). Note the name service lookup order is set via `nsswitch.conf`(5) mechanism.

If using a caching name server add the line "nameserver 127.0.0.1" first. To get a local caching name server to run you will need to set "named=yes" in `/etc/rc.conf` and create the `named.conf` file in the appropriate place for `named`(8), usually in `/etc/namedb`. The same holds true if the machine is going to be a name server for your domain. In both these cases, make sure that `named`(8) is running (otherwise there are long waits for resolver timeouts).

RPC-based network services

Several services depend on the RPC portmapper `rpcbind`(8) - formerly known as **`portmap`** - being running for proper operation. This includes YP (NIS) and NFS exports, among other services. To get the RPC portmapper to start automatically on boot, you will need to have this line in `/etc/rc.conf`:

```
rpcbind=YES
```

YP (NIS) Setup

Check the YP domain name with the `domainname`(1) command. If necessary, correct it by editing the `/etc/defaultdomain` file or by setting the "*domainname*" variable in `/etc/rc.conf`. The `/etc/rc.d/network` script reads this file on bootup to determine and set the domain name. You may also set the running system's domain name with the `domainname`(1) command. To start YP client services, simply run **`ypbind`**, then perform the remaining YP activation as described in `passwd`(5) and `group`(5).

In particular, to enable YP passwd support, you'll need to update `/etc/nsswitch.conf` to include "nis" for the "passwd" and "group" entries. A traditional way to accomplish the same thing is to add following entry to local passwd database via `vipw`(8):

```
+:*::::::::
```

Note this entry has to be the very last one. This traditional way works with the default `nsswitch.conf`(5) setting of "passwd", which is "compat".

There are many more YP man pages available to help you. You can find more information by starting with `yp`(8).

Check disk mounts

Check that the disks are mounted correctly by comparing the `/etc/fstab` file against the output of the `mount`(8) and `df`(1) commands. Example:

```
# cat /etc/fstab
/dev/sd0a / ffs     rw              1 1
/dev/sd0b none swap sw
/dev/sd0e /usr ffs  rw              1 2
/dev/sd0f /var ffs  rw              1 3
/dev/sd0g /tmp ffs  rw              1 4
/dev/sd0h /home ffs rw              1 5

# mount
/dev/sd0a on / type ffs (local)
/dev/sd0e on /usr type ffs (local)
/dev/sd0f on /var type ffs (local)
/dev/sd0g on /tmp type ffs (local)
/dev/sd0h on /home type ffs (local)

# df
```

```
Filesystem  1024-blocks     Used    Avail Capacity  Mounted on
/dev/sd0a         22311    14589     6606    69%    /
/dev/sd0e        203399   150221    43008    78%    /usr
/dev/sd0f         10447      682     9242     7%    /var
/dev/sd0g         18823        2    17879     0%    /tmp
/dev/sd0h          7519     5255     1888    74%    /home

# pstat -s
Device      512-blocks     Used    Avail Capacity  Priority
/dev/sd0b       131072    84656    46416    65%    0
```

Edit `/etc/fstab` and use the `mount`(8) and `umount`(8) commands as appropriate. Refer to the above example and `fstab`(5) for information on the format of this file.

You may wish to do NFS mounts now too, or you can do them later.

Concatenated disks (ccd)

If you are using `ccd`(4) concatenated disks, edit `/etc/ccd.conf`. You may wish to take a look to `ccdconfig`(8) for more information about this file. Use the **`ccdconfig -U`** command to unload and the **`ccdconfig -C`** command to create tables internal to the kernel for the concatenated disks. You then `mount`(8), `umount`(8), and edit `/etc/fstab` as needed.

Automounter daemon (AMD)

To use the `amd`(8) automounter, create the `/etc/amd` directory, copy example config files from `/usr/share/examples/amd` to `/etc/amd` and customize them as needed. Alternatively, you can get your maps with YP.

Clock synchronization

In order to make sure the system clock is synchronized to that of a publicly accessible NTP server, make sure that `/etc/rc.conf` contains the following:

```
ntpdate=yes
ntpd=yes
```

See `date`(1), `ntpdate`(8), `ntpd`(8), `rdate`(8), and `timed`(8) for more information on setting the system's date.

CHANGING /etc FILES

The system should be usable now, but you may wish to do more customizing, such as adding users, etc. Many of the following sections may be skipped if you are not using that package (for example, skip the **Kerberos** section if you won't be using Kerberos). We suggest that you **`cd /etc`** and edit most of the files in that directory.

Note that the `/etc/motd` file is modified by `/etc/rc.d/motd` whenever the system is booted. To keep any custom message intact, ensure that you leave two blank lines at the top, or your message will be overwritten.

Add new users

To add new users and groups, there are `useradd`(8) and `groupadd`(8), see also `user`(8) for further programs for user and group manipulation. You may use `vipw`(8) to add users to the `/etc/passwd` file and edit `/etc/group` by hand to add new groups. The manual page for `su`(1), tells you to make sure to put people in the 'wheel' group if they need root access (non-Kerberos). For example:

```
wheel:*:0:root,myself
```

Follow instructions for `kerberos`(8) if using Kerberos for authentication.

System boot scripts and /etc/rc.local

`/etc/rc` and the `/etc/rc.d/*` scripts are invoked at boot time after single user mode has exited, and at shutdown. The whole process is controlled by the master script `/etc/rc`. This script should not be changed by administrators.

The directory `/etc/rc.d` contains a serie of scripts used at startup/shutdown, called by `/etc/rc`. `/etc/rc` is in turn influenced by the configuration variables present in `/etc/rc.conf`.

The script `/etc/rc.local` is run as the last thing during multiuser boot, and is provided to allow any other local hooks necessary for the system.

rc.conf

To enable or disable various services on system startup, corresponding entries can be made in `/etc/rc.conf`. You can take a look at `/etc/defaults/rc.conf` to see a list of default system variables, which you can override in `/etc/rc.conf`. Note you are *not* supposed to change `/etc/defaults/rc.conf` directly, edit only `/etc/rc.conf`. See `rc.conf`(5) for further information.

If you've installed X, you may want to turn on `xdm`(1), the X Display Manager. To do this, set the variable "xdm" to yes in `/etc/rc.conf`, i.e.: "xdm=yes"

Printers

Edit `/etc/printcap` and `/etc/hosts.lpd` to get any printers set up. Consult `lpd`(8) and `printcap`(5) if needed.

Tighten up security

In `/etc/inetd.conf` comment out any extra entries you do not need, and only add things that are really needed. Note that by default all services are disabled for security reasons.

Kerberos

If you are going to use Kerberos for authentication, see `kerberos`(8) and "info heimdal" for more information. If you already have a Kerberos master, change directory to `/etc/kerberosV` and configure. Remember to get a `srvtab` from the master so that the remote commands work.

Mail Aliases

Check `/etc/mail/aliases` and update appropriately if you want e-mail to be routed to non-local address or to different users.

Run `newaliases`(1) after changes.

Postfix

NetBSD comes also with Postfix in the base system. You may wish to set it up in favor of sendmail. Take a look to `/etc/postfix/main.cf` and enable the daemon in `/etc/rc.conf` using "postfix=yes". It is very important to configure `/etc/mailer.conf` to point to Postfix binaries.

DHCP server

If this is a DHCP server, edit `/etc/dhcpd.conf` and `/etc/dhcpd.interfaces` as needed. You will have to make sure `/etc/rc.conf` has "dhcpd=yes" or run `dhcpd`(8) manually.

Bootparam server

If this is a Bootparam server, edit `/etc/bootparams` as needed. You will have to turn it on in `/etc/rc.conf` by adding "bootparamd=yes".

NFS server

If this is an NFS server, make sure `/etc/rc.conf` has:

```
nfs_server=yes
mountd=yes
rpcbind=yes
```

Edit `/etc/exports` and get it correct. After this, you can start the server by issuing:

```
/etc/rc.d/rpcbind start
/etc/rc.d/mountd start
/etc/rc.d/nfsd start
```

which will also start dependencies.

HP remote boot server

Edit `/etc/rbootd.conf` if needed for remote booting. If you do not have HP computers doing remote booting, do not enable this.

Daily, weekly, monthly scripts

Look at and possibly edit the `/etc/daily.conf`, `/etc/weekly.conf`, and `/etc/monthly.conf` configuration files. You can check which values you can set by looking to their matching files in `/etc/defaults`. Your site specific things should go into `/etc/daily.local`, `/etc/weekly.local`, and `/etc/monthly.local`.

These scripts have been limited so as to keep the system running without filling up disk space from normal running processes and database updates. (You probably do not need to understand them.)

Other files in /etc

Look at the other files in `/etc` and edit them as needed. (Do not edit files ending in `.db` — like `pwd.db`, `spwd.db`, nor `localtime`, nor `rmt`, nor any directories.)

Crontab (background running processes)

Check what is running by typing **`crontab -l`** as root and see if anything unexpected is present. Do you need anything else? Do you wish to change things? For example, if you do not like root getting standard output of the daily scripts, and want only the security scripts that are mailed internally, you can type **`crontab -e`** and change some of the lines to read:

```
30  1  *  *  *   /bin/sh /etc/daily 2>&1 > /var/log/daily.out
30  3  *  *  6   /bin/sh /etc/weekly 2>&1 > /var/log/weekly.out
30  5  1  *  *   /bin/sh /etc/monthly 2>&1 > /var/log/monthly.out
```

See `crontab`(5).

Next day cleanup

After the first night's security run, change ownerships and permissions on files, directories, and devices; root should have received mail with subject: "<hostname> daily insecurity output.". This mail contains a set of security recommendations, presented as a list looking like this:

```
var/mail:
        permissions (0755, 0775)
etc/daily:
```

```
            user (0, 3)
```

The best bet is to follow the advice in that list. The recommended setting is the first item in parentheses, while the current setting is the second one. This list is generated by `mtree`(8) using `/etc/mtree/special`. Use `chmod`(1), `chgrp`(1), and `chown`(8) as needed.

Packages

Install your own packages. The NetBSD packages collection, pkgsrc, includes a large set of third-party software. A lot of it is available as binary packages that you can download from `ftp://ftp.NetBSD.org/pub/NetBSD/packages/` or a mirror, and install using `pkg_add`(1). See `http://www.NetBSD.org/docs/pkgsrc/` and `pkgsrc/doc/pkgsrc.txt` for more details.

Copy vendor binaries and install them. You will need to install any shared libraries, etc. (Hint: **`man -k compat`** to find out how to install and use compatibility mode.)

There is also other third-party software that is available in source form only, either because it has not been ported to NetBSD yet, because licensing restrictions make binary redistribution impossible, or simply because you want to build your own binaries. Sometimes checking the mailing lists for past problems that people have encountered will result in a fix posted.

Check the running system

You can use `ps`(1), `netstat`(1), and `fstat`(1) to check on running processes, network connections, and opened files, respectively. Other tools you may find useful are `systat`(1) and `top`(1).

COMPILING A KERNEL

Note: The standard NetBSD kernel configuration (GENERIC) is suitable for most purposes.

First, review the system message buffer in `/var/run/dmesg.boot` and by using the `dmesg`(8) command to find out information on your system's devices as probed by the kernel at boot. In particular, note which devices were not configured. This information will prove useful when editing kernel configuration files.

To compile a kernel inside a writable source tree, do the following:

```
$ cd /usr/src/sys/arch/SOMEARCH/conf
$ cp GENERIC SOMEFILE (only the first time)
$ vi SOMEFILE (adapt to your needs)
$ config SOMEFILE
$ cd ../compile/SOMEFILE
$ make depend
$ make
```

where *`SOMEARCH`* is the architecture (e.g., i386), and *`SOMEFILE`* should be a name indicative of a particular configuration (often that of the hostname).

If you are building your kernel again, before you do a **`make`** you should do a **`make clean`** after making changes to your kernel options.

After either of these two methods, you can place the new kernel (called `netbsd`) in `/` (i.e., `/netbsd`) by issuing **`make install`** and the system will boot it next time. The old kernel is stored as `/onetbsd` so you can boot it in case of failure.

If you are using toolchain to build your kernel, you will also need to build a new set of toolchain binaries. You can do it by changing into `/usr/src` and issuing:

```
$ cd /usr/src
$ K=sys/arch/‘uname -m‘/conf
$ cp $K/GENERIC $K/SOMEFILE
$ vi $K/SOMEFILE (adapt to your needs)
$ ./build.sh tools
$ ./build.sh kernel=SOMEFILE
```

SYSTEM TESTING

At this point, the system should be fully configured to your liking. It is now a good time to ensure that the system behaves according to its specifications and that it is stable on your hardware. You can easily do so by running the test suites available at `/usr/tests/`, assuming that you installed the `tests.tgz` set. If not, you can install it now by running:

```
# cd /
# tar xzpf /path/to/tests.tgz
```

Once done, edit the `/etc/atf/NetBSD.conf` file to tune the configuration of the test suite, go to `/usr/tests/` hierarchy and use the `atf-run`(1) and `atf-report`(1) utilities to run all the tests in an automated way:

```
# cd /usr/tests/
# atf-run | atf-report
```

Should any problems appear when running the test suite, please let the NetBSD developers know by sending a message to the appropriate mailing list or by sending a problem report. For more details see:

- `http://www.netbsd.org/mailinglists/`
- `http://www.netbsd.org/support/send-pr.html`

SEE ALSO

`atf-report`(1), `atf-run`(1), `chgrp`(1), `chmod`(1), `config`(1), `crontab`(1), `date`(1), `df`(1), `domainname`(1), `hostname`(1), `make`(1), `man`(1), `netstat`(1), `newaliases`(1), `passwd`(1), `su`(1), `ccd`(4), `aliases`(5), `crontab`(5), `exports`(5), `fstab`(5), `group`(5), `krb.conf`(5), `krb.realms`(5), `mailer.conf`(5), `passwd`(5), `rc.conf`(5), `resolv.conf`(5), `hier`(7), `hostname`(7), `pkgsrc`(7), `adduser`(8), `amd`(8), `bootparamd`(8), `ccdconfig`(8), `chown`(8), `dhcpd`(8), `ifconfig`(8), `inetd`(8), `kerberos`(8), `mount`(8), `mrouted`(8), `mtree`(8), `named`(8), `rbootd`(8), `rc`(8), `rmt`(8), `route`(8), `umount`(8), `vipw`(8), `ypbind`(8)

HISTORY

This document first appeared in OpenBSD 2.2. It has been adapted to NetBSD and first appeared in NetBSD 2.0.

NAME

ahdilabel — modify AHDI partitions

SYNOPSIS

ahdilabel *disk*

DESCRIPTION

ahdilabel allows you to modify the AHDI partition table on a disk partitioned with AHDI or an AHDI compatible formatter. The 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.

ahdilabel supports the following options:

disk The name of the disk you want to edit. **ahdilabel** will first try to open a disk of this name. If this cannot be opened, it will attempt to open *r*<disk>*c*. Finally, if this also cannot be opened, it will attempt to open */dev/r*<disk>*c*.

ahdilabel will display information about the number of sectors, tracks and sectors on the disk, as well as the current AHDI partition information. It will then prompt for input. The input choices are:

a-p Modify a partition. You will be prompted for a partition id, root, start and size. NetBSD recognises the following partition id's:

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 NetBSD partition letter *a*. The following NBD partitions will be mapped from letter *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.

The root, start and size parameters can be entered using sector, cylinder/track/sector or megabyte notations. Whole numbers of cylinders can be entered using the shorthand <cylinder>/. Likewise, whole numbers of tracks can be entered using the shorthand <cylinder>/<track>/. Megabytes are entered using the suffix *M*.

The following can also be used to enter partition parameters:

−N (root)	Position the root sector for this partition immediately after partition N.
−N (start)	Make this partition start after partition N (leaving a gap of 1 sector for a root sector, if necessary).
−N (size)	Make this partition end immediately before partition N.
-1 (size)	Make this partition extend to the end of the disk.

The sector holding the primary AHDI partition table only has space for four partitions. Thus, if a disk has more than four partitions, the extra partition information is held in auxiliary root sectors. There is one auxiliary root for each additional partition (and also for the fourth partition, if the disk has more than four partitions).

r Recalculate the root sectors. This will automatically assign auxiliary root sectors if the disk has more than 4 partitions. The auxiliary root sectors will be positioned in a default location preceding the relevant partition.

s Show the current partition information.

u Toggle the unit display between sector and cylinder/track/sector notation.

w Write the AHDI partition table to the disk.

z Options for zero'ing the boot sector and bad sector lists. The default is to preserve them both.

q Quit

EXAMPLES

```
      ahdilabel sd0
```

Edit the AHDI label for disk sd0.

SEE ALSO

`bootpref`(8), `disklabel`(8), `installboot`(8)

HISTORY

The **`ahdilabel`** command first appeared in NetBSD 1.5.

BUGS

The changes made to the AHDI partitions will become active on the next *first open* of the disk. You are advised to use **`ahdilabel`** only on a disk without any mounted or otherwise active partitions. This is not enforced by **`ahdilabel`**.

Because of way NetBSD interprets AHDI partition tables to create the NetBSD disklabel, the NetBSD partition ordering may change if partitions labelled NBD are created or removed.

Creating an AHDI partition table on a disk that previously did not have one will almost certainly overwrite any existing partition information and/or data on that disk. This is especially the case if auxiliary root sectors are needed for the AHDI partition table.

As soon as a disk contains at least one NBD partition, you are allowed to write NetBSD disklabels and install bootstraps.

NAME

altqd — ALTQ daemon

SYNOPSIS

altqd [**-dv**] [**-f** *conf_file*]

DESCRIPTION

altqd is a daemon program that reads a configuration file and then sets up the ALTQ state of network interfaces. After configuring the ALTQ state, **altqd** will detach and become a daemon.

The signals `SIGINT` or `SIGTERM` will shutdown **altqd**, and the signal `SIGHUP` will restart **altqd**.

The following options are available:

-d Debug mode. **altqd** does not detach and goes into the command mode.

-f *conf_file*
Specify a configuration file to read instead of the default. The default file is `/etc/altq.conf`.

-v Print debugging information. This option implies **-d**.

COMMANDS

When **-d** option is provided, **altqd** goes into the command mode after reading the configuration file and setting up the ALTQ state. Each command is a single line, starting with the command verb.

The basic commands are as follows:

help | **?**
Display a complete list of commands and their syntax.

quit Exit.

altq *reload*
Reload the configuration file and reinitialize ALTQ.

altq *interface* [enable|disable]
Enables or disables ALTQ on the interface named *interface*. When **altqd** enters the command mode, ALTQ is enabled on all the interfaces listed in the configuration file.

FILES

`/etc/altq.conf`	configuration file
`/var/run/altqd.pid`	pid of the running **altqd**
`/var/run/altq_quip`	Unix domain socket for communicating with `altqstat`(1)

SEE ALSO

`altqstat`(1), `altq.conf`(5), `altq`(9)

NAME

amd − automatically mount file systems

SYNOPSIS

amd −H
amd [**−F** *conf_file*]
amd [**−nprvHS**] [**−a** *mount_point*] [**−c** *duration*] [**−d** *domain*] [**−k** *kernel-arch*] [**−l** *logfile*] [**−o** *op_sys_ver*] [**−t** *interval.interval*] [**−w** *interval*] [**−x** *log-option*] [**−y** *YP-domain*] [**−A** *arch*] [**−C** *cluster-name*] [**−D** *option*] [**−F** *conf_file*] [**−O** *op_sys_name*] [**−T** *tag*] [*directory mapname* [*−map-options*]] . . .

DESCRIPTION

Amd is a daemon that automatically mounts filesystems whenever a file or directory within that filesystem is accessed. Filesystems are automatically unmounted when they appear to have become quiescent.

Amd operates by attaching itself as an NFS server to each of the specified *directories***.** Lookups within the specified directories are handled by **amd**, which uses the map defined by *mapname* to determine how to resolve the lookup. Generally, this will be a host name, some filesystem information and some mount options for the given filesystem.

In the first form depicted above, **amd** will print a short help string. In the second form, if no options are specified, or the **-F** is used, **amd** will read configuration parameters from the file *conf_file* which defaults to **/etc/amd.conf**. The last form is described below.

OPTIONS

−a *temporary-directory*
Specify an alternative location for the real mount points. The default is **/a**.

−c *duration*
Specify a *duration*, in seconds, that a looked up name remains cached when not in use. The default is 5 minutes.

−d *domain*
Specify the local domain name. If this option is not given the domain name is determined from the hostname.

−k *kernel-arch*
Specifies the kernel architecture. This is used solely to set the ${karch} selector.

−l *logfile*
Specify a logfile in which to record mount and unmount events. If *logfile* is the string **syslog** then the log messages will be sent to the system log daemon by *syslog*(3). The default syslog facility used is LOG_DAEMON. If you wish to change it, append its name to the log file name, delimited by a single colon. For example, if *logfile* is the string **syslog:local7** then **Amd** will log messages via *syslog*(3) using the LOG_LOCAL7 facility (if it exists on the system).

−n Normalize hostnames. The name refereed to by ${rhost} is normalized relative to the host database before being used. The effect is to translate aliases into ‘‘official’’ names.

−o *op_sys_ver*
Override the compiled-in version number of the operating system. Useful when the built in version is not desired for backward compatibility reasons. For example, if the build in version is ‘‘2.5.1’’, you can override it to ‘‘5.5.1’’, and use older maps that were written with the latter in mind.

−p Print PID. Outputs the process-id of **amd** to standard output where it can be saved into a file.

−r Restart existing mounts. **Amd** will scan the mount file table to determine which filesystems are currently mounted. Whenever one of these would have been auto-mounted, **amd** *inherits* it.

−t *timeout.retransmit*
Specify the NFS timeout *interval*, in tenths of a second, between NFS/RPC retries (for UDP only). The default is 0.8 seconds. The second value alters the retransmit counter, which defaults to 11 retransmissions. Both of these values are used by the kernel to communicate with amd. Useful defaults are supplied if either or both values are missing.

Amd relies on the kernel RPC retransmit mechanism to trigger mount retries. The values of these parameters change the overall retry interval. Too long an interval gives poor interactive response; too short an interval causes excessive retries.

−v Version. Displays version and configuration information on standard error.

−w *interval*
Specify an *interval*, in seconds, between attempts to dismount filesystems that have exceeded their cached times. The default is 2 minutes.

−x *options*
Specify run-time logging options. The options are a comma separated list chosen from: fatal, error, user, warn, info, map, stats, all.

−y *domain*
Specify an alternative NIS domain from which to fetch the NIS maps. The default is the system domain name. This option is ignored if NIS support is not available.

−A *arch*
Specifies the OS architecture. This is used solely to set the ${arch} selector.

−C *cluster-name*
Specify an alternative HP-UX cluster name to use.

−D *option*
Select from a variety of debug options. Prefixing an option with the strings **no** reverses the effect of that option. Options are cumulative. The most useful option is **all**. Since *−D* is only used for debugging other options are not documented here: the current supported set of options is listed by the −v option and a fuller description is available in the program source.

−F *conf_file*
Specify an amd configuration file to use. See **amd.conf**(5) for description of this file’s format. This configuration file is used to specify any options in lieu of typing many of them on the command line. The *amd.conf* file includes directives for every command line option amd has, and many more that are only available via the configuration file facility. The configuration file specified by this option is processed after all other options had been processed, regardless of the actual location of this option on the command line.

−H Print help and usage string.

−O *op_sys_name*
Override the compiled-in name of the operating system. Useful when the built in name is not desired for backward compatibility reasons. For example, if the build in name is ‘‘sunos5’’, you can override it to ‘‘sos5’’, and use older maps which were written with the latter in mind.

−S Do not lock the running executable pages of amd into memory. To improve amd’s performance, systems that support the **plock**(3) call, could lock the amd process into memory. This way there is less chance the operating system will schedule, page out, and swap the amd process as needed. This tends improves amd’s performance, at the cost of reserving the memory used by the amd process (making it unavailable for other processes). If this behavior is not desired, use the **−S** option.

−T *tag* Specify a tag to use with **amd.conf**(5). All map entries tagged with *tag* will be processed. Map entries that are not tagged are always processed. Map entries that are tagged with a tag other than *tag* will not be processed.

FILES

/a directory under which filesystems are dynamically mounted

/etc/amd.conf
default configuration file

CAVEATS

Some care may be required when creating a mount map.

Symbolic links on an NFS filesystem can be incredibly inefficient. In most implementations of NFS, their interpolations are not cached by the kernel and each time a symlink is encountered during a *lookuppn* translation it costs an RPC call to the NFS server. It would appear that a large improvement in real-time performance could be gained by adding a cache somewhere. Replacing symlinks with a suitable incarnation of the auto-mounter results in a large real-time speedup, but also causes a large number of process context switches.

A weird imagination is most useful to gain full advantage of all the features.

SEE ALSO

domainname(1), **hostname**(1), **syslog**(3), **amd.conf**(5), **amq**(8), **mount**(8), **umount**(8)
‘‘am-utils’’ **info**(1) entry.

‘‘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

Jan-Simon Pendry <jsp@doc.ic.ac.uk>, Department of Computing, Imperial College, London, UK.

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

amldb — executing and debugging AML interpreter (with DSDT files)

SYNOPSIS

amldb [**-dhst**] *dsdt_file ...*

DESCRIPTION

The **amldb** utility parses the DSDT (Differentiated System Description Table) files, which usually are acquired from ACPI BIOS, and executes the sequence of ACPI Control Methods described in AML (ACPI Machine Language) with its AML interpreter. The **amldb** utility also has a simple ACPI virtual machine. During execution of the Control Methods each access to the region, such as SystemMemory, SystemIO, PCI_Config, does not affect the real hardware but only the virtual machine. Because the sequence of virtual accesses is maintained in user space, AML interpreter developers need not worry about any effect on hardware when they analyze DSDT data files. They can develop and debug the interpreter, even if the machine has no ACPI BIOS.

The developer will need to acquire a DSDT data file from any machine with ACPI BIOS through acpidump(8). The DSDT is a table, a part of the whole ACPI memory table located in somewhere in the BIOS area (0xa0000 - 0x100000). It includes such information as the detailed hardware information for PnP, and the set of procedures which perform power management from the OS. The information is stored in AML format.

The AML interpreter can execute any of the Control Methods specified by users. When executed, it interprets the byte sequence in the Control Method of DSDT, and disassembles the opcodes that it recognizes into ASL (ACPI Source Language) format to be displayed.

If it encounters one of more accesses to the region such as SystemMemory in executing the Control Methods, its ACPI Virtual Machine simulates the input/output operations to the resources in the region. In writing to a certain region, the ACPI Virtual Machine prepares a piece of memory corresponding to its address, if necessary, and holds the specified value in the memory as the *region contents*. In reading from a certain region, it fetches the value in the memory (*region contents*), prompts with it as the following:

```
DEBUG[read(0, 0x100b6813)&mask:0x1](default: 0x1 / 1) >>
```

for users to have the opportunity to modify it, and hands it to the AML interpreter. In case that there is no corresponding region in the AML Virtual Machine, the value zero is handed.

The interpreter continues to maintain all of the *region contents* until **amldb** terminates. You can specify their initial values with the file region.ini in the current directory. If it is executed with **-d** option, it dumps the final status of all of its *region contents* to the file region.dmp when it terminates. Each line of there files consists of the following fields, separated by tabs; region type, address, and value. Region types are specified as follows;

Value	**Region type**
0	SystemMemory
1	SystemIO
2	PCI_Config
3	EmbeddedControl
4	SMBus

Interactive commands are described below:

s *Single step*: Performs single-step execution of the current Control Method. If the next instruction is an invocation of another Control Method, the step execution will continue in the following Control Method.

n *Step program*: Performs single-step execution of the current Control Method. Even if the next instruction is an invocation of another Control Method, the step execution will not continue.

c *Continue program being debugged*: Resumes execution of the AML interpreter. Because the current **amldb** has no way of breakpoint, this command might not so much useful.

q *Quit method execution*: Terminates execution of the current Control Method. If **amldb** is not in execution, this command causes to input the next DSDT data file. If there are no next DSDT data files, it terminates **amldb** itself.

t *Show local name space tree and variables*: Displays the structure of the ACPI namespace tree. If **amldb** is in execution, this command displays the structure that relates to the objects, arguments, and local variables below the scope of the current Control Method.

i *Toggle region input prompt*: Switches whether the prompt for modifying the value read from the *region contents* be showed or not. Default is On.

o *Toggle region output prompt*: Switches whether the prompt for modifying the value to be written to the region contents will be shown or not. The default is Off.

m *Show memory management statistics*: Displays the current statistics of the memory management system on the AML interpreter.

r *method*

Run specified method: Executes the specified Control Method. If it requires one or more arguments, a prompt such as the following appears;

```
Method: Arg 1 From 0x280626ce To 0x28062775
  Enter argument values (ex. number 1 / string foo). ’q’ to quit.
  Arg0 ?
```

For each argument, a pair of type string and value delimited by one or more spaces can be entered. Now only **number** and **string** can be specified as the type string. In the current implementation, only the first character of the type string, such as **n** or **s**, is identified. For example, we can enter as follows:

```
  Arg0 ? n 1
```

f *string*

Find named objects from namespace: Lists the named objects that includes the specified string as the terminate elements searching from the ACPI namespace. For the namespace is expressed as the sequence of four-character elements, appropriate number of additional underscore (‘_’) characters are necessary for specifying objects which have less than four character string. Unless additional underscores specified, matching occurs as the beginning of word with the specified number of characters.

h *Show help messsage*: Displays the command summary of **amldb**.

OPTIONS

Exactly one of the following options must be specified. Otherwise, **amldb** shows its usage and terminates.

-d Dump the final status of all of the *region contents* in the ACPI Virtual Machine to the file `region.dmp`.

-h Terminate with the usage of this command.

-s Display the statistics of the memory management system on the AML interpreter when **amldb** terminates.

-t Display the tree structure of ACPI namespace after the DSDT data file is read.

FILES

`region.ini`
`region.dmp`

EXAMPLES

The following is an example including, invoking the **amldb**, searching `_PRS` (Possible Resource Settings) objects, and executing the `_PTS` (Prepare To Sleep) Control Method by the AML interpreter.

```
% amldb p2b.dsdt.dat
Loading p2b.dsdt.dat...done
AML>f _PRS
\_SB_.PCI0.ISA_.PS2M._PRS.
\_SB_.PCI0.ISA_.IRDA._PRS.
\_SB_.PCI0.ISA_.UAR2._PRS.
\_SB_.PCI0.ISA_.UAR1._PRS.
\_SB_.PCI0.ISA_.ECP_._PRS.
\_SB_.PCI0.ISA_.LPT_._PRS.
\_SB_.PCI0.ISA_.FDC0._PRS.
\_SB_.LNKD._PRS.
\_SB_.LNKC._PRS.
\_SB_.LNKB._PRS.
\_SB_.LNKA._PRS.
AML>r _PTS
Method: Arg 1 From 0x2805f0a3 To 0x2805f0db
  Enter argument values (ex. number 1 / string foo). 'q' to quit.
  Arg0 ? n 5
==== Running _PTS. ====
AML>s
[_PTS. START]
If(LNot(LEqual(Arg0, 0x5)))
AML>
If(LEqual(Arg0, 0x1))
AML>
If(LEqual(Arg0, 0x2))
AML>
Store(One, TO12)
[aml_region_write(1, 1, 0x1, 0xe42c, 0x18, 0x1)]
amldb: region.ini: No such file or directory
        [1:0x00@0xe42f]->[1:0x01@0xe42f]
[write(1, 0x1, 0xe42f)]
[aml_region_read(1, 1, 0xe42c, 0x18, 0x1)]
        [1:0x01@0xe42f]
DEBUG[read(1, 0xe42f)&mask:0x1](default: 0x1 / 1) >>
[read(1, 0xe42f)->0x1]
AML>
Or(Arg0, 0xf0, Local2)[Copy number 0xf5]
AML>t
_PTS  Method: Arg 1 From 0x2805f0a3 To 0x2805f0db
  Arg0    Num:0x5
  Local2  Num:0xf5
```

```
AML>s
Store(Local2, DBG1)
[aml_region_write(1, 1, 0xf5, 0x80, 0x0, 0x8)]
        [1:0x00@0x80]->[1:0xf5@0x80]
[write(1, 0xf5, 0x80)]
[aml_region_read(1, 1, 0x80, 0x0, 0x8)]
        [1:0xf5@0x80]
DEBUG[read(1, 0x80)&mask:0xf5](default: 0xf5 / 245) >>
[read(1, 0x80)->0xf5]
AML>
[_PTS. END]
_PTS  Method: Arg 1 From 0x2805f0a3 To 0x2805f0db
NO object
==== _PTS finished. ====
AML>q
%
```

SEE ALSO

acpi(4), acpidump(8)

HISTORY

The **amldb** utility appeared in FreeBSD 5.0.

AUTHORS

Takanori Watanabe ⟨takawata@FreeBSD.org⟩
Mitsuru IWASAKI ⟨iwasaki@FreeBSD.org⟩
Yasuo YOKOYAMA ⟨yokoyama@jp.FreeBSD.org⟩

Some contributions made by
Chitoshi Ohsawa ⟨ohsawa@catv1.ccn-net.ne.jp⟩,
Takayasu IWANASHI ⟨takayasu@wendy.a.perfect-liberty.or.jp⟩,
Norihiro KUMAGAI ⟨kumagai@home.com⟩,
Kenneth Ingham ⟨ingham@I-pi.com⟩, and
Michael Lucas ⟨mwlucas@blackhelicopters.org⟩.

BUGS

The ACPI virtual machine does not completely simulate the behavior of a machine with an ACPI BIOS. In the current implementation, the ACPI virtual machine only reads or writes the stored values by emulating access to regions such as SystemMemory.

Because the AML interpreter interprets and disassembles simultaneously, it is impossible to implement such features as setting breakpoints with the specified line number in ASL. Setting breakpoints at certain Control Methods, which is not very difficult, has not yet implemented because nobody has ever needed it.

NAME

amq − automounter query tool

SYNOPSIS

amq [**−fmpsvwHTU**] [**−h** *hostname*] [**−l** *log_file*] [**−x** *log_options*] [**−D** *debug_options*] [**−P** *program_number*] [[**−u**] *directory* . . .]

DESCRIPTION

Amq provides a simple way of determining the current state of **amd** program. Communication is by RPC. Three modes of operation are supported by the current protocol. By default a list of mount points and automounted filesystems is output. An alternative host can be specified using the *−h* option.

If *directory* names are given, as output by default, then per-filesystem information is displayed.

OPTIONS

−f Ask the automounter to flush the internal caches and reload all the maps.

−h *hostname*
Specify an alternate host to query. By default the local host is used. In an HP-UX cluster, the root server is queried by default, since that is the system on which the automounter is normally run.

−l *log_file*
Tell amd to use *log_file* as the log file name. For security reasons, this must be the same log file which amd used when started. This option is therefore only useful to refresh amd's open file handle on the log file, so that it can be rotated and compressed via daily cron jobs.

−m Ask the automounter to provide a list of mounted filesystems, including the number of references to each filesystem and any error which occurred while mounting.

−p Return the process ID of the remote or locally running amd. Useful when you need to send a signal to the local amd process, and would rather not have to search through the process table. This option is used in the *ctl-amd* script.

−s Ask the automounter to provide system-wide mount statistics.

−u Ask the automounter to unmount the filesystems named in *directory* instead of providing information about them. Unmounts are requested, not forced. They merely cause the mounted filesystem to timeout, which will be picked up by **amd**'s main scheduler thus causing the normal timeout action to be taken.

−v Ask the automounter for its version information. This is a subset of the information output by **amd**'s *-v* option.

−w Translate a full pathname as returned by *getcwd*(3) into a short **Amd** pathname that goes through its mount points. This option requires that **Amd** is running.

−x *log_options*
Ask the automounter to use the logging options specified in *log_options* from now on.

−D *log_options*
Ask the automounter to use the debugging options specified in *debug_options* from now on.

−H Display short usage message.

−P *program_number*
: Contact an alternate running amd that had registered itself on a different RPC *program_number* and apply all other operations to that instance of the automounter. This is useful when you run multiple copies of amd, and need to manage each one separately. If not specified, amq will use the default program number for amd, 300019. For security reasons, the only alternate program numbers amd can use range from 300019 to 300029, inclusive.

−T Contact **amd** using the TCP transport only. Normally **amq** will try TCP, and if that failed, will try UDP.

−U Contact **amd** using UDP (connectionless) transport only. Normally **amq** will try TCP, and if that failed, will try UDP.

FILES

amq.x	RPC protocol description.

CAVEATS

Amq uses a Sun registered RPC program number (300019 decimal) which may not be in the /etc/rpc database.

If the TCP wrappers library is available, and the **use_tcpwrappers** global **amd.conf** option is set to ‘‘yes’’, then **amd** will verify that the host running **amq** is authorized to connect. The *amd* service name must used in the **/etc/hosts.allow** and **/etc/hosts.deny** files. For example, to allow only localhost to connect to **amd,** add this line to **/etc/hosts.allow:**

```
amd: localhost
```

and this line to **/etc/hosts.deny:**

```
amd: ALL
```

SEE ALSO

amd(8), **amd.conf**(5), **hosts_access**(5).

‘‘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

Jan-Simon Pendry <jsp@doc.ic.ac.uk>, Department of Computing, Imperial College, London, UK.

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

amrctl — Control utility for AMI MegaRaid controllers

SYNOPSIS

amrctl *stat* [**-bgv**] [**-a** *attempts*] [**-f** *device*] [**-l** *volno*] [**-p** *driveno*] [**-s** *busno*] [**-s** *busno:driveno*] [**-t** *microseconds*]

DESCRIPTION

The **amrctl** queries or controls AMI MegaRaid controllers supported by the amr(4) driver.

Only the *stat* subcommand is currently implemented, and reports status of the controller. The options for the *stat* subcommand are as follows:

-a *attempts*	Number of retries for a command before giving up. Default 5.
-b	Report battery status.
-f *device*	Device to use. Default to `/dev/amr0`.
-g	Report global paramters for the controller.
-l *volno*	Report status of a logical drive.
-p *driveno*	Report status of a physical drive.
-s *busno*	Report status of all physical drives on the specified bus.
-s *busno:driveno*	Report status of the specified physical drive on the specified bus.
-t *microseconds*	Delay between retries for a command.
-v	Increase verbosity level by one.

SEE ALSO

amr(4)

HISTORY

The **amrctl** command first appeared in NetBSD 4.0.

AUTHORS

The **amrctl** command was written by Pierre David ⟨Pierre.David@crc.u-strasbg.fr⟩ and Jung-uk Kim ⟨jkim@FreeBSD.org⟩ for FreeBSD.

NAME

anvil − Postfix session count and request rate control

SYNOPSIS

anvil [generic Postfix daemon options]

DESCRIPTION

The Postfix **anvil**(8) server maintains statistics about client connection counts or client request rates. This information can be used to defend against clients that hammer a server with either too many simultaneous sessions, or with too many successive requests within a configurable time interval. This server is designed to run under control by the Postfix **master**(8) server.

In the following text, **ident** specifies a (service, client) combination. The exact syntax of that information is application-dependent; the **anvil**(8) server does not care.

CONNECTION COUNT/RATE CONTROL

To register a new connection send the following request to the **anvil**(8) server:

request=connect
ident=*string*

The **anvil**(8) server answers with the number of simultaneous connections and the number of connections per unit time for the (service, client) combination specified with **ident**:

status=0
count=*number*
rate=*number*

To register a disconnect event send the following request to the **anvil**(8) server:

request=disconnect
ident=*string*

The **anvil**(8) server replies with:

status=0

MESSAGE RATE CONTROL

To register a message delivery request send the following request to the **anvil**(8) server:

request=message
ident=*string*

The **anvil**(8) server answers with the number of message delivery requests per unit time for the (service, client) combination specified with **ident**:

status=0
rate=*number*

RECIPIENT RATE CONTROL

To register a recipient request send the following request to the **anvil**(8) server:

request=recipient
ident=*string*

The **anvil**(8) server answers with the number of recipient addresses per unit time for the (service, client) combination specified with **ident**:

status=0
rate=*number*

TLS SESSION NEGOTIATION RATE CONTROL

The features described in this section are available with Postfix 2.3 and later.

To register a request for a new (i.e. not cached) TLS session send the following request to the **anvil**(8) server:

request=newtls
ident=*string*

The **anvil**(8) server answers with the number of new TLS session requests per unit time for the (service, client) combination specified with **ident**:

status=0
rate=*number*

To retrieve new TLS session request rate information without updating the counter information, send:

request=newtls_report
ident=*string*

The **anvil**(8) server answers with the number of new TLS session requests per unit time for the (service, client) combination specified with **ident**:

status=0
rate=*number*

SECURITY

The **anvil**(8) server does not talk to the network or to local users, and can run chrooted at fixed low privilege.

The **anvil**(8) server maintains an in-memory table with information about recent clients requests. No persistent state is kept because standard system library routines are not sufficiently robust for update-intensive applications.

Although the in-memory state is kept only temporarily, this may require a lot of memory on systems that handle connections from many remote clients. To reduce memory usage, reduce the time unit over which state is kept.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

Upon exit, and every **anvil_status_update_time** seconds, the server logs the maximal count and rate values measured, together with (service, client) information and the time of day associated with those events. In order to avoid unnecessary overhead, no measurements are done for activity that isn't concurrency limited or rate limited.

BUGS

Systems behind network address translating routers or proxies appear to have the same client address and can run into connection count and/or rate limits falsely.

In this preliminary implementation, a count (or rate) limited server can have only one remote client at a time. If a server reports multiple simultaneous clients, state is kept only for the last reported client.

The **anvil**(8) server automatically discards client request information after it expires. To prevent the

anvil(8) server from discarding client request rate information too early or too late, a rate limited service should always register connect/disconnect events even when it does not explicitly limit them.

CONFIGURATION PARAMETERS

On low-traffic mail systems, changes to **main.cf** are picked up automatically as **anvil**(8) processes run for only a limited amount of time. On other mail systems, use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

anvil_rate_time_unit (60s)
The time unit over which client connection rates and other rates are calculated.

anvil_status_update_time (600s)
How frequently the **anvil**(8) connection and rate limiting server logs peak usage information.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

smtpd(8), Postfix SMTP server
postconf(5), configuration parameters
master(5), generic daemon options

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
TUNING_README, performance tuning

LICENSE

The Secure Mailer license must be distributed with this software.

HISTORY

The anvil service is available in Postfix 2.2 and later.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research

P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

apm, **zzz** — Advanced Power Management control program

SYNOPSIS

apm [**-abdlmsSvz**] [**-f** *sockname*]
zzz [**-Sz**] [**-f** *sockname*]

DESCRIPTION

apm communicates with the Advanced Power Management daemon, apmd(8), making requests of it for current power status or to place the system into a suspend or stand-by state. With no flags, **apm** displays the current power management state in verbose form.

Available command-line flags are:

-z Put the system into suspend (deep sleep) mode.
-S Put the system into stand-by (light sleep) mode.
-l Display the estimated battery lifetime in percent.
-m Display the estimated battery lifetime in minutes.
-b Display the battery status. 0 means high, 1 means low, 2 means critical, 3 means charging, 4 means absent, and 255 means unknown.
-a Display the external charger (A/C status). 0 means disconnected, 1 means connected, 2 means backup power source, and 255 means unknown.
-s Display if power management is enabled.
-v Request more verbose description of the displayed states.
-f *sockname*
Set the name of the socket via which to contact apmd(8) to sockname.
-d Do not communicate with the APM daemon; attempt instead to manipulate the APM control device directly.

The **zzz** variant of this command is an alternative for suspending the system. With no arguments, **zzz** places the system into suspend mode. The command line flags serve the same purpose as for the **apm** variant of this command.

This command does not wait for positive confirmation that the requested mode has been entered; to do so would mean the command does not return until the system resumes from its sleep state.

FILES

/var/run/apmdev is the default UNIX-domain socket used for communication with apmd(8). The **-f** flag may be used to specify an alternate socket name. The protection modes on this socket govern which users may access the APM functions.

/dev/apmctl is the control device which is used when the **-d** flag is specified; it must be writable for the **-d** flag to work successfully. /dev/apm is the status device used when the socket is not accessible; it must be readable to provide current APM status.

SEE ALSO

apm(4), apmd(8)

REFERENCES

Advanced Power Management (APM) BIOS Interface Specification (revision 1.1), Intel Corporation and Microsoft Corporation

HISTORY

The **apm** command appeared in NetBSD 1.3.

NAME

apmd — Advanced Power Management monitor daemon

SYNOPSIS

apmd [**-adlqsv**] [**-t** *rate*] [**-S** *sockname*] [**-m** *sockmode*]
[**-o** *sockowner:sockgroup*] [**-f** *devname*]

DESCRIPTION

apmd monitors the advanced power management (APM) pseudo-device, acting on signaled events and upon user requests as sent by the apm(8) utility. For suspend and standby request events delivered by the BIOS, or via apm(8), **apmd** runs the appropriate configuration program (if one exists), syncs the buffer cache to disk and initiates the requested mode. When resuming after suspend or standby, **apmd** runs the appropriate configuration utility (if one exists). For power status change events, **apmd** fetches the current status and reports it via syslog(3) with logging facility LOG_DAEMON.

apmd announces the transition to standby mode with a single high tone on the speaker (using the /dev/speaker device). Suspends are announced with two high tones.

apmd periodically polls the APM driver for the current power state. If the battery charge level changes substantially or the external power status changes, the new status is logged. The polling rate defaults to once per 10 minutes, but may be specified using the **-t** command-line flag.

apmd supports the following options:

-a Any BIOS-initiated suspend or standby requests are ignored if the system is connected to line current and not running from batteries (user requests are still honored).

-d Enter debug mode, log to facility LOG_LOCAL1 and stay in the foreground on the controlling terminal.

-f *devname*
Specify an alternate device file name.

-l A low-battery event causes a suspend request to occur.

-m *sockmode*
Use *sockmode* instead of '0660' for the mode of /var/run/apmdev.

-o *sockowner:sockgroup*
Use *sockowner:sockgroup* instead of ''0:0'' for the owner/group of /var/run/apmdev.

-q Do not announce suspend and standby requests on the speaker.

-s The current battery statistics are reported via syslog(3) and exit without monitoring the APM status.

-S *sockname*
Specify an alternate socket name (used by apm(8) to communicate with **apmd**).

-t *rate* Change the polling rate from 600 seconds to *rate* seconds.

-v Periodically log the power status via syslog(3).

When a client requests a suspend or stand-by mode, **apmd** does not wait for positive confirmation that the requested mode has been entered before replying to the client; to do so would mean the client does not get a reply until the system resumes from its sleep state. Rather, **apmd** replies with the intended state to the client and then places the system in the requested mode after running the configuration script and flushing the buffer cache.

Actions can be configured for the five transitions: **suspend**, **standby**, **resume**, **line** or **battery**. The suspend and standby actions are run prior to **apmd** performing any other actions (such as disk syncs) and entering the new mode. The resume program is run after resuming from a stand-by or suspended state.

The line and battery actions are run after switching power sources to AC (line) or battery, respectively. The appropriate line or battery action is also run upon the startup of apmd based on the current power source.

FILES

/etc/apm/suspend	
/etc/apm/standby	
/etc/apm/resume	
/etc/apm/line	
/etc/apm/battery	Contain the host's customized actions. Each file must be an executable binary or shell script suitable for execution by the execve(2) function. If you wish to have the same program or script control all transitions, it may determine which transition is in progress by examining its *argv[0]* which is set to one of *suspend*, *standby*, *resume*, *line* or *battery*. See /usr/share/examples/apm/script for such an example script.
/var/run/apmdev	The default UNIX-domain socket used for communication with apm(8). The socket is protected by default to mode 0660, UID 0, GID 0.
/dev/apmctl	The default device used to control the APM kernel driver.

SEE ALSO

execve(2), syslog(3), apm(4), speaker(4), apm(8), syslogd(8)

REFERENCES

Advanced Power Management (APM) BIOS Interface Specification (revision 1.1), Intel Corporation and Microsoft Corporation.

HISTORY

The **apmd** command appeared in NetBSD 1.3.

NAME

apmlabel — update disk label from Apple Partition Map

SYNOPSIS

apmlabel [**-fqrw**] *device*

DESCRIPTION

apmlabel is used to update a NetBSD disk label from the Apple Partition Map found on disks that were previously used on Mac OS systems (or other APM using systems).

apmlabel scans the APM contained in the first blocks of the disk and generates additional partition entries for the disk from the entries found. Driver and patches partitions are ignored.

Each APM entry which does not have an equivalent partition in the disk label (equivalent in having the same size and offset) is added to the first free partition slot in the disk label. A free partition slot is defined as one with an `fstype` of '`unused`' and a `size` of zero ('0'). If there are not enough free slots in the disk label, a warning will be issued.

The raw partition (typically partition *c*, but *d* on i386 and some other platforms) is left alone during this process.

By default, the proposed changed disk label will be displayed and no disk label update will occur.

Available options:

-f
Force an update, even if there has been no change.

-q
Performs operations in a quiet fashion.

-r
In conjunction with **-w**, also update the on-disk label.

-w
Update the in-core label if it has been changed.

SEE ALSO

`disklabel`(8), `dkctl`(8), `pdisk`(8)

HISTORY

The **apmlabel** command appeared in NetBSD 5.0.

NAME

arp — address resolution display and control

SYNOPSIS

arp [**−n**] *hostname*
arp [**−nv**] **−a**
arp [**−v**] **−d −a**
arp [**−v**] **−d** *hostname* [*proxy*]
arp −s *hostname ether_addr* [*temp*] [*pub* [*proxy*]]
arp −f *filename*

DESCRIPTION

The **arp** program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (arp(4)). With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

Available options:

−a The program displays all of the current ARP entries.

−d A super-user may delete an entry for the host called *hostname* with the **−d** flag. If the **proxy** keyword is specified, only the published "proxy only" ARP entry for this host will be deleted. If used with **−a** instead of a *hostname*, it will delete all arp entries.

−f Causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

hostname ether_addr [*temp*] [*pub*]

with argument meanings as described below.

−n Show network addresses as numbers (normally **arp** attempts to display addresses symbolically).

−s *hostname ether_addr*
Create an ARP entry for the host called *hostname* with the Ethernet address *ether_addr*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word *temp* is given in the command. If the word *pub* is given, the entry will be "published"; i.e., this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. If the word *proxy* is also given, the published entry will be a "proxy only" entry.

−v Display verbose information when adding or deleting ARP entries.

SEE ALSO

inet(3), arp(4), ifconfig(8)

HISTORY

The **arp** command appeared in 4.3BSD.

NAME

atactl — a program to manipulate ATA (IDE) devices and busses

SYNOPSIS

atactl *device command* [*arg* [...]]

DESCRIPTION

atactl allows a user or system administrator to issue commands to and otherwise control devices which reside on standard IDE and ATA controllers, or the ATA bus itself. It is used by specifying a device or bus to manipulate, the command to perform, and any arguments the command may require.

DEVICE COMMANDS

The following commands may be used on IDE and ATA devices. Note that not all devices support all commands.

identify Identify the specified device, displaying the device's vendor, product, revision strings, and the device's capabilities.

idle Place the specified device into Idle mode. This mode may consume less power than Active mode.

standby Place the specified device into Standby mode. This mode will consume less power than Idle mode.

sleep Place the specified device into Sleep mode. This mode will consume less power than Standby mode, but requires a device reset to resume operation. Typically the wd(4) driver performs this reset automatically, but this should still be used with caution.

setidle *idle-timer*
Places the specified device into Idle mode, and sets the Idle timer to *idle-timer* seconds. A value of 0 will disable the Idle timer.

setstandby *standby-timer*
Places the specified device into Standby mode, and sets the Standby timer to *standby-timer* seconds. A value of 0 will disable the Standby timer.

checkpower
Will print out if the device is in Active, Idle, or Standby power management mode.

apm [*disable* | *set #*]
Controls the Advanced Power Management feature of the specified device. Advanced Power Management is an optional feature used to specify a power management level to balance between device performance and power consumption.

disable Disable the Advanced Power Management.

set # Enable the Advanced Power Management feature and set its level to the value #, where # is an integer within the scale 0-253; being 0 the mode with the lowest power consumption (and thus the worse performance) and 253 the mode which provides the better performance at a cost of more power consumption.

It should be noted that the effect of the value need not be continous. For example, a device might provide only two modes: one from 0 to 126 and other from 127 to 253. Per the specification, values of 127 and higher do not permit the device to spin down to save power.

smart [*enable* | *disable* | *status* | *offline #* | *error-log* | *selftest-log*]

Controls SMART feature set of the specified device. SMART stands for Self-Monitoring, Analysis, and Reporting Technology. It provides an early warning system by comparing subtle operation characteristics to those determined in vendor testing to precede device failures.

enable Enables access to SMART capabilities within the device. Prior to being enabled, a SMART capable device neither monitors nor saves SMART attribute values. The state of SMART, either enabled or disabled, will be preserved by the device across power cycles.

disable Disables access to SMART capabilities within the device. Attribute values will be saved, and will no longer be monitored.

status Reports whether SMART is supported by the device, and whether SMART is enabled on the device (can only be determined on ATA6 or better devices). If SMART is enabled, then a table of attribute information is printed. Attributes are the specific performance or calibration parameters that are used in analyzing the status of the device. The specific set of attributes being used and the identity of these attributes is vendor specific and proprietary.

Attribute values are used to represent the relative reliability of individual performance or calibration parameters. The valid range of attribute values is from 1 to 253 decimal. Lower values indicate that the analysis algorithms being used by the device are predicting a higher probability of a degrading or faulty condition.

Each attribute value has a corresponding threshold limit which is used for direct comparison to the attribute value to indicate the existence of a degrading or faulty condition. The numerical value of the attribute thresholds are determined by the device manufacturer through design and reliability testing and analysis. Each attribute threshold represents the lowest limit to which its corresponding attribute value can equal while still retaining a positive reliability status.

If the crit field is "yes" then negative reliability of this attribute predicts imminent data loss. Otherwise it merely indicates that the intended design life period of usage or age has been exceeded. The collect field indicates whether this attribute is updated while the device is online. The reliability field indicates whether the attribute value is within the acceptable threshold.

offline # Runs the numbered offline self-test on the drive.

error-log Prints the error log.

selftest-log Prints the self-test log.

security [*freeze* | *status*]

Controls "security" (password protection) features of modern ATA drives. The security commands are intended to be issued by low-level software (firmware / BIOS) only. Generally, the security status should be "frozen" before the operating system is started so that misbehaving or malicious software cannot set or change a password. Older and buggy BIOSes neglect to do so; in these cases it might make sense to issue the "freeze" command early in the boot process.

freeze	freezes the drive's security status
status	displays the drive's security status

BUS COMMANDS

The following commands may be used on IDE and ATA busses. Note that not all devices support all commands.

reset Reset the bus. This will reset all ATA devices present on the bus. Any ATAPI device with pending commands will also be reset.

SEE ALSO

`ioctl`(2), `wd`(4), `dkctl`(8), `scsictl`(8)

HISTORY

The **atactl** command first appeared in NetBSD 1.4.

AUTHORS

The **atactl** command was written by Ken Hornstein. It was based heavily on the `scsictl`(8) command written by Jason R. Thorpe.

BUGS

The output from the **identify** command is rather ugly.

NAME

atrun — run jobs queued for later execution

SYNOPSIS

atrun [**-l** *load_avg*] [**-d**]

DESCRIPTION

atrun runs jobs queued by at(1). Root's crontab(5) must contain the line:

```
*/10    *     *     *     *     root    /usr/libexec/atrun
```

so that atrun(8) gets called every ten minutes.

At every invocation, every job in lowercase queues whose starting time has passed is started. A maximum of one batch job (denoted by uppercase queues) is started each time **atrun** is invoked.

OPTIONS

-l *load_avg*
: Specifies a limiting load factor, over which batch jobs should not be run, instead of the compiled-in value of 1.5.

-d
: Debug; print error messages to standard error instead of using syslog(3).

WARNINGS

For **atrun** to work, you have to start up a cron(8) daemon.

FILES

/var/at/spool	Directory containing output spool files
/var/at/jobs	Directory containing job files

SEE ALSO

at(1), crontab(1), syslog(3), crontab(5), cron(8)

AUTHORS

Thomas Koenig ⟨ig25@rz.uni-karlsruhe.de⟩

BUGS

The functionality of **atrun** should be merged into cron(8).

NAME

authpf — authenticating gateway user shell

SYNOPSIS

authpf

DESCRIPTION

authpf is a user shell for authenticating gateways. It is used to change pf(4) rules when a user authenticates and starts a session with sshd(8) and to undo these changes when the user's session exits. It is designed for changing filter and translation rules for an individual source IP address as long as a user maintains an active ssh(1) session. Typical use would be for a gateway that authenticates users before allowing them Internet use, or a gateway that allows different users into different places. **authpf** logs the successful start and end of a session to syslogd(8). This, combined with properly set up filter rules and secure switches, can be used to ensure users are held accountable for their network traffic.

authpf can add filter and translation rules using the syntax described in pf.conf(5). **authpf** requires that the pf(4) system be enabled before use. **authpf** can also maintain the list of IP address of connected users in the "authpf_users" table.

authpf is meant to be used with users who can connect via ssh(1) only. On startup, **authpf** retrieves the client's connecting IP address via the SSH_CLIENT environment variable and, after performing additional access checks, reads a template file to determine what filter and translation rules (if any) to add. On session exit the same rules that were added at startup are removed.

Each **authpf** process stores its rules in a separate ruleset inside a pf(4) anchor shared by all **authpf** processes. By default, the anchor name "authpf" is used, and the ruleset names equal the username and PID of the **authpf** processes as "username(pid)". The following rules need to be added to the main ruleset /etc/pf.conf in order to cause evaluation of any **authpf** rules:

```
nat-anchor "authpf/*"
rdr-anchor "authpf/*"
binat-anchor "authpf/*"
anchor "authpf/*"
```

The "/*" at the end of the anchor name is required for pf(4) to process the rulesets attached to the anchor by **authpf**.

FILTER AND TRANSLATION RULES

Filter and translation rules for **authpf** use the same format described in pf.conf(5). The only difference is that these rules may (and probably should) use the macro *user_ip*, which is assigned the connecting IP address whenever **authpf** is run. Additionally, the macro *user_id* is assigned the user name.

Filter and translation rules are stored in a file called authpf.rules. This file will first be searched for in /etc/authpf/users/$USER/ and then in /etc/authpf/. Only one of these files will be used if both are present.

Per-user rules from the /etc/authpf/users/$USER/ directory are intended to be used when non-default rules are needed on an individual user basis. It is important to ensure that a user can not write or change these configuration files.

The authpf.rules file must exist in one of the above locations for **authpf** to run.

CONFIGURATION

Options are controlled by the /etc/authpf/authpf.conf file. If the file is empty, defaults are used for all configuration options. The file consists of pairs of the form name=value, one per line. Currently, the allowed values are as follows:

anchor=name
: Use the specified `anchor` name instead of "authpf".

table=name
: Use the specified `table` name instead of "authpf_users".

USER MESSAGES

On successful invocation, **authpf** displays a message telling the user he or she has been authenticated. It will additionally display the contents of the file `/etc/authpf/authpf.message` if the file exists and is readable.

There exist two methods for providing additional granularity to the control offered by **authpf** - it is possible to set the gateway to explicitly allow users who have authenticated to `ssh`(1) and deny access to only a few troublesome individuals. This is done by creating a file with the banned user's login name as the filename in `/etc/authpf/banned/`. The contents of this file will be displayed to a banned user, thus providing a method for informing the user that they have been banned, and where they can go and how to get there if they want to have their service restored. This is the default behaviour.

It is also possible to configure **authpf** to only allow specific users access. This is done by listing their login names, one per line, in `/etc/authpf/authpf.allow`. If "*" is found on a line, then all usernames match. If **authpf** is unable to verify the user's permission to use the gateway, it will print a brief message and die. It should be noted that a ban takes precedence over an allow.

On failure, messages will be logged to `syslogd`(8) for the system administrator. The user does not see these, but will be told the system is unavailable due to technical difficulties. The contents of the file `/etc/authpf/authpf.problem` will also be displayed if the file exists and is readable.

CONFIGURATION ISSUES

authpf maintains the changed filter rules as long as the user maintains an active session. It is important to remember however, that the existence of this session means the user is authenticated. Because of this, it is important to configure `sshd`(8) to ensure the security of the session, and to ensure that the network through which users connect is secure. `sshd`(8) should be configured to use the *ClientAliveInterval* and *ClientAliveCountMax* parameters to ensure that a ssh session is terminated quickly if it becomes unresponsive, or if arp or address spoofing is used to hijack the session. Note that TCP keepalives are not sufficient for this, since they are not secure. Also note that *AllowTcpForwarding* should be disabled for **authpf** users to prevent them from circumventing restrictions imposed by the packet filter ruleset.

authpf will remove state table entries that were created during a user's session. This ensures that there will be no unauthenticated traffic allowed to pass after the controlling `ssh`(1) session has been closed.

authpf is designed for gateway machines which typically do not have regular (non-administrative) users using the machine. An administrator must remember that **authpf** can be used to modify the filter rules through the environment in which it is run, and as such could be used to modify the filter rules (based on the contents of the configuration files) by regular users. In the case where a machine has regular users using it, as well as users with **authpf** as their shell, the regular users should be prevented from running **authpf** by using the `/etc/authpf/authpf.allow` or `/etc/authpf/banned/` facilities.

authpf modifies the packet filter and address translation rules, and because of this it needs to be configured carefully. **authpf** will not run and will exit silently if the `/etc/authpf/authpf.conf` file does not exist. After considering the effect **authpf** may have on the main packet filter rules, the system administrator may enable **authpf** by creating an appropriate `/etc/authpf/authpf.conf` file.

FILES

```
/etc/authpf/authpf.conf
/etc/authpf/authpf.allow
/etc/authpf/authpf.rules
/etc/authpf/authpf.message
/etc/authpf/authpf.problem
```

EXAMPLES

Control Files − To illustrate the user-specific access control mechanisms, let us consider a typical user named bob. Normally, as long as bob can authenticate himself, the **authpf** program will load the appropriate rules. Enter the `/etc/authpf/banned/` directory. If bob has somehow fallen from grace in the eyes of the powers-that-be, they can prohibit him from using the gateway by creating the file `/etc/authpf/banned/bob` containing a message about why he has been banned from using the network. Once bob has done suitable penance, his access may be restored by moving or removing the file `/etc/authpf/banned/bob`.

Now consider a workgroup containing alice, bob, carol and dave. They have a wireless network which they would like to protect from unauthorized use. To accomplish this, they create the file `/etc/authpf/authpf.allow` which lists their login ids, one per line. At this point, even if eve could authenticate to `sshd`(8), she would not be allowed to use the gateway. Adding and removing users from the work group is a simple matter of maintaining a list of allowed userids. If bob once again manages to annoy the powers-that-be, they can ban him from using the gateway by creating the familiar `/etc/authpf/banned/bob` file. Though bob is listed in the allow file, he is prevented from using this gateway due to the existence of a ban file.

Distributed Authentication − It is often desirable to interface with a distributed password system rather than forcing the sysadmins to keep a large number of local password files in sync. The `login.conf`(5) mechanism in OpenBSD can be used to fork the right shell. To make that happen, `login.conf`(5) should have entries that look something like this:

```
shell-default:shell=/bin/csh

default:\
        ...
        :shell=/usr/sbin/authpf

daemon:\
        ...
        :shell=/bin/csh:\
        :tc=default:

staff:\
        ...
        :shell=/bin/csh:\
        :tc=default:
```

Using a default password file, all users will get **authpf** as their shell except for root who will get `/bin/csh`.

SSH Configuration − As stated earlier, `sshd`(8) must be properly configured to detect and defeat network attacks. To that end, the following options should be added to `sshd_config`(5):

```
Protocol 2
ClientAliveInterval 15
ClientAliveCountMax 3
```

This ensures that unresponsive or spoofed sessions are terminated within a minute, since a hijacker should not be able to spoof ssh keepalive messages.

Banners − Once authenticated, the user is shown the contents of `/etc/authpf/authpf.message`. This message may be a screen-full of the appropriate use policy, the contents of `/etc/motd` or something as simple as the following:

```
     This means you will be held accountable by the powers that be
     for traffic originating from your machine, so please play nice.
```

To tell the user where to go when the system is broken, `/etc/authpf/authpf.problem` could contain something like this:

```
     Sorry, there appears to be some system problem. To report this
     problem so we can fix it, please phone 1-900-314-1597 or send
     an email to remove@bulkmailerz.net.
```

Packet Filter Rules − In areas where this gateway is used to protect a wireless network (a hub with several hundred ports), the default rule set as well as the per-user rules should probably allow very few things beyond encrypted protocols like `ssh`(1), `ssl`(8), or `ipsec`(4). On a securely switched network, with plug-in jacks for visitors who are given authentication accounts, you might want to allow out everything. In this context, a secure switch is one that tries to prevent address table overflow attacks.

Example `/etc/pf.conf`:

```
# by default we allow internal clients to talk to us using
# ssh and use us as a dns server.
internal_if="fxp1"
gateway_addr="10.0.1.1"
nat-anchor "authpf/*"
rdr-anchor "authpf/*"
binat-anchor "authpf/*"
block in on $internal_if from any to any
pass in quick on $internal_if proto tcp from any to $gateway_addr \
      port = ssh
pass in quick on $internal_if proto udp from any to $gateway_addr \
      port = domain
anchor "authpf/*"
```

For a switched, wired net − This example `/etc/authpf/authpf.rules` makes no real restrictions; it turns the IP address on and off, logging TCP connections.

```
external_if = "xl0"
internal_if = "fxp0"

pass in log quick on $internal_if proto tcp from $user_ip to any \
      keep state
pass in quick on $internal_if from $user_ip to any
```

For a wireless or shared net − This example `/etc/authpf/authpf.rules` could be used for an insecure network (such as a public wireless network) where we might need to be a bit more restrictive.

```
internal_if="fxp1"
ipsec_gw="10.2.3.4"

# rdr ftp for proxying by ftp-proxy(8)
rdr on $internal_if proto tcp from $user_ip to any port 21 \
      -> 127.0.0.1 port 8081

# allow out ftp, ssh, www and https only, and allow user to negotiate
# ipsec with the ipsec server.
pass in log quick on $internal_if proto tcp from $user_ip to any \
      port { 21, 22, 80, 443 } flags S/SA
pass in quick on $internal_if proto tcp from $user_ip to any \
      port { 21, 22, 80, 443 }
pass in quick proto udp from $user_ip to $ipsec_gw port = isakmp \
      keep state
pass in quick proto esp from $user_ip to $ipsec_gw
```

Dealing with NAT − The following `/etc/authpf/authpf.rules` shows how to deal with NAT, using tags:

```
ext_if = "fxp1"
ext_addr = 129.128.11.10
int_if = "fxp0"
# nat and tag connections...
nat on $ext_if from $user_ip to any tag $user_ip -> $ext_addr
pass in quick on $int_if from $user_ip to any
pass out log quick on $ext_if tagged $user_ip keep state
```

With the above rules added by **authpf**, outbound connections corresponding to each users NAT'ed connections will be logged as in the example below, where the user may be identified from the ruleset name.

```
# tcpdump -n -e -ttt -i pflog0
Oct 31 19:42:30.296553 rule 0.bbeck(20267).1/0(match): pass out on fxp1: \
129.128.11.10.60539 > 198.137.240.92.22: S 2131494121:2131494121(0) win \
16384 <mss 1460,nop,nop,sackOK> (DF)
```

Using the authpf_users table − Simple **authpf** settings can be implemented without an anchor by just using the "authpf_users" `table`. For example, the following `pf.conf`(5) lines will give SMTP and IMAP access to logged in users:

```
table <authpf_users> persist
pass in on $ext_if proto tcp from <authpf_users> \
        to port { smtp imap } keep state
```

It is also possible to use the "authpf_users" `table` in combination with anchors. For example, `pf`(4) processing can be sped up by looking up the anchor only for packets coming from logged in users:

```
table <authpf_users> persist
anchor "authpf/*" from <authpf_users>
rdr-anchor "authpf/*" from <authpf_users>
```

SEE ALSO

`pf`(4), `pf.conf`(5), `ftp-proxy`(8)

HISTORY

The **authpf** program first appeared in OpenBSD 3.1.

BUGS

Configuration issues are tricky. The authenticating `ssh`(1) connection may be secured, but if the network is not secured the user may expose insecure protocols to attackers on the same network, or enable other attackers on the network to pretend to be the user by spoofing their IP address.

authpf is not designed to prevent users from denying service to other users.

NAME

automount2amd − converts old Sun automount maps to Amd maps

SYNOPSIS

automount2amd *auto.map*

DESCRIPTION

automount2amd is used to convert an old Sun automount maps named *auto.map* to an Amd map.

This perl script will use the following /default entry

```
    type:=nfs;opts:=rw,grpid,nosuid,utimeout=600
```

If you wish to override that, define the **$DEFAULTS** environment variable, or modify the script.

If you wish to generate Amd maps using the *hostd* (host domain) Amd map syntax, then define the environment variable **$DOMAIN** or modify the script.

EXAMPLE

Say you have the Sun automount file auto.foo, with these two lines:

```
  home                earth:/home
  moon  -ro,intr        server:/proj/images
```

Running

```
    automount2amd auto.foo > amd.foo
```

will produce the Amd map *amd.foo* with this content:

```
# generated by automount2amd on Sat Aug 14 17:59:32 US/Eastern 1999

/defaults \
  type:=nfs;opts:=rw,grpid,nosuid,utimeout=600

home   host==earth;type:=link;fs:=/home \
  rhost:=earth;rfs:=/home

moon   -addopts:=ro,intr \
  host==server;type:=link;fs:=/proj/images \
  rhost:=server;rfs:=/proj/images
```

BUGS

automount2amd does not understand newer Sun Automount map syntax, those used by autofs.

SEE ALSO

amd(8).

‘‘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

Original author Mike Walker <mike@tab00.larc.nasa.gov>. Script modified by 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

bad144 — read/write DEC standard 144 bad sector information

SYNOPSIS

bad144 [**−c**] [**−f**] [**−v**] *disk* [*sno* [*bad ...*]]
bad144 **−a** [**−c**] [**−f**] [**−v**] *disk* [*bad ...*]

DESCRIPTION

bad144 can be used to inspect the information stored on a disk that is used by the disk drivers to implement bad sector forwarding.

Available options:

−a The argument list consists of new bad sectors to be added to an existing list. The new sectors are sorted into the list, which must have been in order. Replacement sectors are moved to accommodate the additions; the new replacement sectors are cleared.

−c Forces an attempt to copy the old sector to the replacement, and may be useful when replacing an unreliable sector.

−f (vax only) For a RP06, RM03, RM05, Fujitsu Eagle, or SMD disk on a MASSBUS, the **−f** option may be used to mark the new bad sectors as ‘‘bad’’ by reformatting them as unusable sectors. This option is *required unless* the sectors have already been marked bad, or the system will not be notified that it should use the replacement sector. This option may be used while running multiuser; it is no longer necessary to perform format operations while running single-user.

−v The entire process is described as it happens in gory detail if **−v** (verbose) is given.

The format of the information is specified by DEC standard 144, as follows. The bad sector information is located in the first 5 even numbered sectors of the last track of the disk pack. There are five identical copies of the information, described by the *dkbad* structure.

Replacement sectors are allocated starting with the first sector before the bad sector information and working backwards towards the beginning of the disk. A maximum of 126 bad sectors are supported. The position of the bad sector in the bad sector table determines the replacement sector to which it corresponds. The bad sectors must be listed in ascending order.

The bad sector information and replacement sectors are conventionally only accessible through the ‘‘c’’ file system partition of the disk. If that partition is used for a file system, the user is responsible for making sure that it does not overlap the bad sector information or any replacement sectors. Thus, one track plus 126 sectors must be reserved to allow use of all of the possible bad sector replacements.

The bad sector structure is as follows:

```
struct dkbad {
        int32_t   bt_csn;             /* cartridge serial number */
        u_int16_t bt_mbz;             /* unused; should be 0 */
        u_int16_t bt_flag;            /* -1 => alignment cartridge */
        struct bt_bad {
                u_int16_t bt_cyl;     /* cylinder number of bad sector */
                u_int16_t bt_trksec;  /* track and sector number */
        } bt_bad[126];
};
```

Unused slots in the *bt_bad* array are filled with all bits set, a putatively illegal value.

bad144 is invoked by giving a device name (e.g. wd0, hk0, hp1, etc.). With no optional arguments it reads the first sector of the last track of the corresponding disk and prints out the bad sector information. It issues a warning if the bad sectors are out of order. **bad144** may also be invoked with a serial number for the pack and a list of bad sectors. It will write the supplied information into all copies of the bad-sector file, replacing any previous information. Note, however, that **bad144** does not arrange for the specified sectors to be marked bad in this case. This procedure should only be used to restore known bad sector information which was destroyed.

It is no longer necessary to reboot to allow the kernel to reread the bad-sector table from the drive.

SEE ALSO

`badsect`(8)

HISTORY

The **bad144** command appeared in 4.1BSD.

BUGS

It should be possible to format disks on-line under 4BSD.

It should be possible to mark bad sectors on drives of all type.

On an 11/750, the standard bootstrap drivers used to boot the system do not understand bad sectors, handle ECC errors, or the special SSE (skip sector) errors of RM80-type disks. This means that none of these errors can occur when reading the file `/netbsd` to boot. Sectors 0-15 of the disk drive must also not have any of these errors.

The drivers which write a system core image on disk after a crash do not handle errors; thus the crash dump area must be free of errors and bad sectors.

NAME

badsect — create files to contain bad sectors

SYNOPSIS

badsect *bbdir sector ...*

DESCRIPTION

badsect makes a file to contain a bad sector. Normally, bad sectors are made inaccessible by the standard formatter, which provides a forwarding table for bad sectors to the driver; see bad144(8) for details. If a driver supports the bad blocking standard it is much preferable to use that method to isolate bad blocks, since the bad block forwarding makes the pack appear perfect, and such packs can then be copied with dd(1). The technique used by this program is also less general than bad block forwarding, as **badsect** can't make amends for bad blocks in the i-list of file systems or in swap areas.

On some disks, adding a sector which is suddenly bad to the bad sector table currently requires the running of the standard DEC formatter. Thus to deal with a newly bad block or on disks where the drivers do not support the bad-blocking standard **badsect** may be used to good effect.

badsect is used on a quiet file system in the following way: First mount the file system, and change to its root directory. Make a directory BAD there. Run **badsect** giving as argument the *BAD* directory followed by all the bad sectors you wish to add. The sector numbers must be relative to the beginning of the file system, but this is not hard as the system reports relative sector numbers in its console error messages. Then change back to the root directory, unmount the file system and run fsck(8) on the file system. The bad sectors should show up in two files or in the bad sector files and the free list. Have fsck(8) remove files containing the offending bad sectors, but *do not* have it remove the BAD/*nnnnn* files. This will leave the bad sectors in only the BAD files.

badsect works by giving the specified sector numbers in a mknod(2) system call, creating an illegal file whose first block address is the block containing bad sector and whose name is the bad sector number. When it is discovered by fsck(8) it will ask "HOLD BAD BLOCK ?" A positive response will cause fsck(8) to convert the inode to a regular file containing the bad block.

DIAGNOSTICS

badsect refuses to attach a block that resides in a critical area or is out of range of the file system. A warning is issued if the block is already in use.

SEE ALSO

bad144(8), fsck(8)

HISTORY

The **badsect** command appeared in 4.1BSD.

BUGS

If more than one of the sectors in a file system fragment are bad, you should specify only one of them to **badsect**, as the blocks in the bad sector files actually cover all the sectors in a file system fragment.

NAME

binpatch — examine and or modify initialized data in a binary file

SYNOPSIS

binpatch [**-b** | **-w** | **-l**] [**-o** *offset*] **-s** *symname* [**-r** *value*] *binfile*
binpatch [**-b** | **-w** | **-l**] [**-o** *offset*] **-a** *addr* [**-r** *value*] *binfile*

DESCRIPTION

binpatch is used to modify or examine the data associated with a symbol in a binary file *binfile*. The flags **-b**, **-w** and **-l** specify the size of the data to be modified or examined (byte, word and long respectively.) The *binfile* is scanned in search of the symbol *symname* (specified with the **-s** flag) If the symbol is found the current data and address are printed. Next if the **-r** flag has been given, the current data is replaced with that of *value*.

If the second form is used the address *addr* specified with the **-a** flag is used as a direct address into the data section of the binary and no symbol search is performed.

The **-o** flag specifies an offset in byte, word or long (**-b**, **-w**, or **-l**) units from the given locator (**-s** or **-a**) for **binpatch** to perform its described actions.

NAME

binpatch — examine and or modify initialized data in a binary file

SYNOPSIS

binpatch [**-b** | **-w** | **-l**] [**-o** *offset*] **-s** *symname* [**-r** *value*] *binfile*
binpatch [**-b** | **-w** | **-l**] [**-o** *offset*] **-a** *addr* [**-r** *value*] *binfile*

DESCRIPTION

binpatch is used to modify or examine the data associated with a symbol in a binary file *binfile*. The flags **-b**, **-w** and **-l** specify the size of the data to be modified or examined (byte, word and long respectively.) The *binfile* is scanned in search of the symbol *symname* (specified with the **-s** flag.) If the symbol is found the current data and address are printed. Next if the **-r** flag has been given, the current data is replaced with that of *value*.

If the second form is used the address *addr* specified with the **-a** flag is used as a direct address into the data section of the binary and no symbol search is performed.

The **-o** flag specifies an offset in byte, word or long (**-b**, **-w**, or **-l**) units from the given locator (**-s** or **-a**) for **binpatch** to perform its described actions.

NAME

/usr/mdec/binstall — install sparc and sparc64 boot blocks

SYNOPSIS

/usr/mdec/binstall [**-htUuv**] [**-b** *bootprog*] [**-f** *filesystem*] [**-m** *mdec*]
[**-i** *installbootprog*] ["net" | "ffs"] [directory]

DESCRIPTION

The **/usr/mdec/binstall** program prepares a sparc or sparc64 system for booting, either from local disk from a "ffs" partition or over the network. The default type of boot block installed is derived from the host system. If it is an UltraSPARC, the sparc64 boot blocks will be used, otherwise the SPARC boot blocks will be used. **/usr/mdec/binstall** can be forced to prepare a disk for either.

OPTIONS

The following options are available:

-b Set the second stage boot program to *bootprog*. This will typically be `boot.net` for sparc systems and `ofwboot.net` for sparc64 systems.

-f Set the path to the filesystem being installed for to *filesystem*. This is otherwise derived from the [directory].

-h Display help.

-i Set the path to the `installboot`(8) program to *installbootprog*. This is useful for using **/usr/mdec/binstall** on non-sparc or sparc64 systems.

-m Sets the path to the machine dependent directory to *mdec*. This is the directory that both the boot blocks and the `installboot`(8) program live.

-t Test mode; does not run any program. Implies the **-v** option.

-U Install sparc (SPARC) boot blocks.

-u Install sparc64 (UltraSPARC) boot blocks.

-v Be verbose.

SEE ALSO

`disklabel`(8), `installboot`(8)

NAME

bioctl — RAID management interface

SYNOPSIS

bioctl *device command* [*arg* [...]]

DESCRIPTION

RAID device drivers which support management functionality can register their services with the bio(4) driver. **bioctl** then can be used to manage the RAID controller's properties.

COMMANDS

The following commands are supported:

show [*disks* | *volumes*]
: Without any argument by default **bioctl** will show information about all volumes and the logical disks used on them. If *disks* is specified, only information about physical disks will be shown. If *volumes* is specified, only information about the volumes will be shown.

alarm [*disable* | *enable* | *silence* | *test*]
: Control the RAID card's alarm functionality, if supported. By default if no argument is specified, its current state will be shown. Optionally the *disable*, *enable*, *silence*, or *test* arguments may be specified to enable, disable, silence, or test the RAID card's alarm.

blink *start channel:target.lun* | *stop channel:target.lun*
: Instruct the device at *channel:target.lun* to start or cease blinking, if there's ses(4) support in the enclosure.

hotspare *add channel:target.lun* | *remove channel:target.lun*
: Create or remove a hot-spare drive at location *channel:target.lun*.

passthru *add DISKID channel:target.lun* | *remove channel:target.lun*
: Create or remove a *pass-through* device. The *DISKID* argument specifies the disk that will be used for the new device, and it will be created at the location *channel:target.lun*. *NOTE*: Removing a pass-through device that has a mounted filesystem will lead to undefined behaviour.

check *start VOLID* | *stop VOLID*
: Start or stop consistency volume check in the volume with index *VOLID*. *NOTE*: Not many RAID controllers support this feature.

create volume *VOLID DISKIDs* [*SIZE*] *STRIPE RAID_LEVEL channel:target.lun*
: Create a volume at index *VOLID*. The *DISKIDs* argument will specify the first and last disk, i.e.: *0-3* will use the disks 0, 1, 2, and 3. The *SIZE* argument is optional and may be specified if not all available disk space is wanted (also dependent of the *RAID_LEVEL*). The volume will have a stripe size defined in the *STRIPE* argument and it will be located at *channel:target.lun*.

remove volume *VOLID channel:target.lun*
: Remove a volume at index *VOLID* and located at *channel:target.lun*. *NOTE*: Removing a RAID volume that has a mounted filesystem will lead to undefined behaviour.

EXAMPLES

The following command, executed from the command line, shows the status of the volumes and its logical disks on the RAID controller:

```
$ bioctl arcmsr0 show
Volume Status       Size         Device/Label   RAID Level Stripe
=================================================================
     0 Building     468G  sd0 ARC-1210-VOL#00       RAID 6  128KB  0% done
   0:0 Online       234G         0:0.0 noencl <WDC WD2500YS-01SHB1 20.06C06>
   0:1 Online       234G         0:1.0 noencl <WDC WD2500YS-01SHB1 20.06C06>
   0:2 Online       234G         0:2.0 noencl <WDC WD2500YS-01SHB1 20.06C06>
   0:3 Online       234G         0:3.0 noencl <WDC WD2500YS-01SHB1 20.06C06>
```

To create a RAID 5 volume on the SCSI 0:15.0 location on the disks 0, 1, 2 and 3, with stripe size of 64Kb on the first volume ID, using all available free space on the disks:

```
$ bioctl arcmsr0 create volume 0 0-3 64 5 0:15.0
```

To remove the volume 0 previously created at the SCSI 0:15.0 location:

```
$ bioctl arcmsr0 remove volume 0 0:15.0
```

SEE ALSO

`arcmsr`(4), `bio`(4), `cac`(4), `ciss`(4), `mfi`(4)

HISTORY

The **bioctl** command first appeared in OpenBSD 3.8, it was rewritten for NetBSD 5.0.

AUTHORS

The **bioctl** interface was written by Marco Peereboom ⟨marco@openbsd.org⟩ and was rewritten with multiple features by
Juan Romero Pardines ⟨xtraeme@NetBSD.org⟩.

NAME

boot — system bootstrapping procedures

DESCRIPTION

This document provides information on using common features in the NetBSD boot loader. Additional information may be found in architecture-specific `boot`(8) manual pages.

Interactive mode

In interactive mode, the boot loader will present a prompt, allowing input of these commands:

boot [*device*:][*filename*] [**-acdqsvxz**]

The default *device* will be set to the disk that the boot loader was loaded from. To boot from an alternate disk, the full name of the device should be given at the prompt. *device* is of the form *xd* [*N*[*x*]] where *xd* is the device from which to boot, *N* is the unit number, and *x* is the partition letter.

The following list of supported devices may vary from installation to installation:

hd	Hard disks.
fd	Floppy drives.

The default *filename* is `netbsd`; if the boot loader fails to successfully open that image, it then tries `netbsd.gz` (expected to be a kernel image compressed by gzip), followed by `netbsd.old`, `netbsd.old.gz`, `onetbsd`, and finally `onetbsd.gz`. Alternate system images can be loaded by just specifying the name of the image.

Options are:

-a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

-c Bring the system up into the device configuration manager. From here the device locators can be tuned to the hardware; see `userconf`(4).

-d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

-q Boot the system in quiet mode.

-s Bring the system up in single-user mode.

-v Boot the system in verbose mode.

-x Boot the system with debug messages enabled.

-z Boot the system in silent mode.

consdev *dev*

Immediately switch the console to the specified device *dev* and reprint the banner. *dev* must be one of *`pc`*, *`com0`*, *`com1`*, *`com2`*, *`com3`*, *`com0kbd`*, *`com1kbd`*, *`com2kbd`*, *`com3kbd`*, or *`auto`*. See **Console Selection Policy** in `boot_console`(8).

dev [*device*]

Set the default drive and partition for subsequent filesystem operations. Without an argument, print the current setting. *device* is of the form specified in **boot**.

help

Print an overview about commands and arguments.

ls [path]
Print a directory listing of path, containing inode number, filename, and file type. path can contain a device specification.

quit
Reboot the system.

In an emergency, the bootstrap methods described in the NetBSD installation notes for the specific architecture can be used.

FILES

/boot	boot program code loaded by the primary bootstrap
/netbsd	system code
/netbsd.gz	gzip-compressed system code
/usr/mdec/boot	master copy of the boot program (copy to /boot)
/usr/mdec/bootxx_fstype	primary bootstrap for filesystem type fstype, copied to the start of the NetBSD partition by installboot(8).

SEE ALSO

Architecture-specific boot(8) manual pages, ddb(4), userconf(4), boot_console(8), halt(8), installboot(8), reboot(8), shutdown(8)

BUGS

Any *filename* specified after the boot options, e.g.:

```
boot -d netbsd.test
```

is ignored, and the default kernel is booted.

NAME

boot — Alpha system bootstrapping procedures

DESCRIPTION

DEC Alpha systems can have either of two different firmware systems: ARC (a.k.a. AlphaBIOS), and SRM. Some Alpha systems have both in their flash RAM and can switch between them on command. ARC is used to bootstrap Microsoft Windows NT for Alpha. SRM is used to bootstrap OpenVMS and Ultrix. NetBSD requires SRM.

SRM can bootstrap from supported local storage devices, e.g., IDE disks or CD-ROM drives, SCSI disks or CD-ROM drives, and floppy drives. SRM can also network bootstrap via supported Ethernet interfaces, using BOOTP or MOP. The particular capabilities of SRM will vary from system to system.

When SRM boots the system, it performs a Power On Self Test (POST), probes the system busses to identify devices, and initializes them. SRM includes an x86 instruction emulator in order to run the BIOS initialization routines found in the PROM of any video cards found. In this way, most generic PCI video cards can work in Alpha systems that have PCI bus slots.

SRM then examines the state of one of several variables: `auto_action`. If the value of `auto_action` is "halt" then SRM will stop, print its prompt: ">>>" and wait for commands to be entered on the console. If the value of `auto_action` is "boot" then SRM will automatically bootstrap the operating system specified by various non-volatile environment variables.

SRM device names are not the same as in NetBSD, e.g., **ewa0** is a DEC "tulip" Ethernet interface, **dka0** is a SCSI disk on a recognized controller, **dqa0** is an IDE disk on a recognized controller. The **show device** command will list all the devices that SRM can bootstrap from.

SRM Commands

SRM is somewhat UNIX-like in that it incorporates a simple pipe and I/O redirection, which allows command sequences like:

```
show config | more
show * | more
```

An essential but incomplete list of SRM commands follows:

boot [**-file** *filename*] [**-flags** *value*] [*device*]

Boot an operating system. The default arguments for this command are taken from the SRM environment variables:

`boot_file`	file name to bootstrap.
`boot_osflags`	flags to pass to the secondary bootstrap program.
`bootdef_dev`	default bootstrap device.

help [*command*]

Invoke the SRM help system.

init

Reset the SRM console, and take actions as specified by SRM variables.

set *variable value* [**-default**]

Set an SRM variable, e.g.,

```
set auto_action boot
set bootdef_dev dka0
set ewa0_mode auto
```

If the **-default** flag is used, the variable will be set to its default value.

show *variable or subsystem*

Show SRM variables and values, or show system state or configuration. If a wildcard is used, then all matching SRM variables are shown, e.g.,

show *	will display all SRM variables.
show b*	will display all variables whose names begin with *b*.
show config	will display the complete system configuration.
show device	will display all bootable devices.
show memory	will display the system's memory configuration.

SRM Variables

auto_action — What SRM will do at system startup or reset:

boot automatically bootstrap the operating system.

halt after POST, prompt the user on the console for commands to execute.

Some Alpha systems (e.g., AlphaServer 800) have a "halt" switch, which if set, will override the action of this variable, and cause SRM to stop after POST and prompt the user for commands to execute.

bootdef_dev — The default bootstrap device, e.g., **dka0**, **dqa0**, **ewa0**. The **show device** command will list the available and recognized bootable devices.

boot_file — The file to bootstrap from; this is a null string by default.

boot_osflags — The flag passed to the secondary bootstrap program, and the NetBSD kernel:

a	(automatic) multi-user mode bootstrap.
c	crash dump immediately after autoconf(4), if the NetBSD kernel is compiled with DEBUG; See options(4).
d	break into the debugger ASAP, if the NetBSD kernel is compiled with DDB or KGDB; See options(4).
h	on a reboot request from the NetBSD kernel, halt the system instead of rebooting.
i	the NetBSD secondary bootstrap program will stop and prompt for the NetBSD kernel file name to bootstrap.
n	the NetBSD kernel will ask for the root file system's device, the kernel core dump device, and the path to init(8).
q	bootstrap quietly.
s	single-user mode bootstrap.
v	bootstrap verbosely.

These may be used in combinations that are not mutually exclusive. These options are case-insensitive to be compatible with DEC operating systems.

console — What type of console device SRM and NetBSD will use:

graphics use a video card for output, and a PC keyboard for input.

	`serial` use the first serial port for console.
	Just as with Sun systems, Alpha systems will use the first serial port as a console if there is no keyboard plugged into the keyboard port, even if `console` is set to "graphics".
`ew*0_mode`	The media and speed for DEC "tulip" Ethernet interfaces (e.g., DECchip 21040, 21140, 21143); possible values are: **`auto`** (IEEE 802.3u "Nway" negotiation), **`BNC`**, **`AUI`**, **`Twisted-Pair`**, **`FastFD`** (Fast Full Duplex).
`ew*0_protocols`	The protocol to use when netbooting, i.e., MOP (Maintenance Operations Protocol), or BOOTP (Bootstrap Protocol).
	The Alpha SRM firmware is picky about BOOTP responses; the `dhcpd.conf`(5) on the server needs the
	`always-reply-rfc1048 on;`
	directive in the section for netbooting Alpha systems.
`os_type`	This determines which system firmware will be used after the next power-cycle, if both ARC and SRM are present in Flash RAM. This should be set to any of "UNIX", "osf", or "vms" to select the SRM console required for NetBSD. OSF refers to the Open Software Foundation.

After bootstrap

Once the NetBSD/alpha kernel is booted normally it initializes itself and proceeds to start the system. An automatic consistency check of the file systems takes place, and unless this fails, the system comes up to multi-user operation.

The proper way to shut the system down is with the `shutdown`(8) command.

If the system crashes, it will enter the kernel debugger, `ddb`(4), if it is configured in the kernel. If the crash occurred during initialization and the debugger is not present or is exited, the kernel will halt the system.

If the crash occurred during normal operation and the debugger is not present or is exited, the system will attempt a dump to the configured dump device (which will be automatically recovered with `savecore`(8) during the next bootstrap cycle), and after the dump is complete (successful or not) the kernel will attempt a reboot.

FILES

`/boot`	NetBSD secondary bootstrap program (installed)
`/netbsd`	default NetBSD system kernel
`/usr/mdec/bootxx_cd9660`	primary bootstrap for "cd9660" (ISO 9660) file system
`/usr/mdec/bootxx_ffs`	primary bootstrap for "ffs" file system (Berkeley Fast File System)
`/usr/mdec/boot`	secondary bootstrap
`/usr/mdec/netboot`	network bootstrap
`/usr/mdec/ustarboot`	"ustar" disk and tape bootstrap

SEE ALSO

`ddb`(4), `diskless`(8), `init`(8), `installboot`(8), `mkbootimage`(8), `rc`(8), `reboot`(8), `savecore`(8), `setnetbootinfo`(8), `shutdown`(8)

Alpha Architecture Reference Manual Third Edition, *Digital Press*, Alpha Architecture Committee, 1998.

BUGS

The device names used by NetBSD/alpha and the SRM Console often have no relation to each other.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

When the NetBSD kernel is booted normally (using one of the two methods discussed below), it initializes itself and proceeds to boot the system. An automatic consistency check of the file systems takes place, and unless this fails, the system comes up to multi-user operations. The proper way to shut the system down is with the shutdown(8) command.

If the system crashes, it will enter the kernel debugger, ddb(4), if it is configured in the kernel. If the debugger is not present, or the debugger is exited, the system will attempt a dump to the configured dump device (which will be automatically recovered with savecore(8) during the next boot cycle). After the dump is complete (successful or not), the system will attempt a reboot.

Booting NetBSD using the bootloader

When a bootable NetBSD partition is created by means of HDTOOLBOX or another RDB editing program and a bootblock has been copied there by installboot(8) and the boot priority of the NetBSD partition is either the highest or the NetBSD partition is selected by means of the boot menu, the Amiga ROM will automatically start the NetBSD bootloader. By default it will, after a short timeout, load the kernel image /netbsd and attempt to boot it into multi-user mode. This behaviour can be changed by typing in an alternate command sequence. The command line looks like:

kernel-path [**−abknpqstvADZ**] [**−c** *model*] [**−m** *memsize*] [**−n** *memsegments*] [**−I** *mask*] [**−S** *amount*] [**−T** *amount*]

kernel-path
This gives you the opportunity to boot another kernel, say: /netbsd.old. The default is /netbsd.

−a Autoboot into multi-user mode (default).

−b Prompt for the root file system device, the system crash dump device, and the path to init(8).

−c *model*
force machine *model*. Use 32000+(Qlogic chip revision) for the DraCo.

−k Reserve the first 4M of fastmem.

−m *memsize*
Force fastmem size to be *memsize* kBytes.

−n maximum number of *segments* of memory to use, encoded as follows: 0 (default): 1 segment, 1: 2 segments, 2: 3 or more segments.

−p Select kernel load segment by priority instead of size.

−q Boot in quiet mode.

−s Boot into single-user mode.

−v Boot in verbose mode.

−D Enter the kernel debugger (best used with -S)

−I *mask*
inhibit sync negotiation as follows: The *mask* is a bitmap expressed in C notation (e.g., 0xff) with 4∗8bits, each bit, if set to 1, disabling sync negotiation for the corresponding target. Note that this only applies to (some of the) real SCSI busses, but not, e.g., to internal IDE. The bytes are used up

from right to left by SCSI bus drivers using this convention.

−S Load the kernel symbols

Booting NetBSD using the loadbsd program

When you want (or have to) start NetBSD from AmigaOS, you have to use the **loadbsd** program that is supplied in the utils directory of the distribution. The loadbsd command line specification is:

loadbsd [**−abknpstADZ**] [**−c** *model*] [**−m** *memsize*] [**−n** *memsegments*] [**−I** *mask*] [**−S** *amount*] [**−T** *amount*] *kernel-path*

Description of options:

−a Autoboot into multi-user mode.

−b Prompt for the root file system device, the system crash dump device, and the path to init(8).

−c force machine model.

−k Reserve the first 4M of fastmem.

−m Force fastmem size to be *memsize* kBytes.

−n maximum number of *segments* of memory to use, encoded as follows: 0 (default): 1 segment, 1: 2 segments, 2: 3 or more segments.

−p Select kernel load segment by priority instead of size.

−s Boot into single-user mode.

−t Test loading of the kernel but don't start NetBSD.

−A enable AGA modes.

−D Enter the kernel debugger after booting. Best with -S.

−I *mask*
inhibit sync negotiation as follows: The *mask* is a bitmap expressed in hexadecimal (e.g., ff) with 4∗8bits, each bit, if set to 1, disabling sync negotiation for the corresponding target. Note that this only applies to (some of the) real SCSI busses, but not, e.g., to internal IDE. The bytes are used up from right to left by SCSI bus drivers using this convention.

−S include kernel debug symbols (for use by -D).

−Z Force load via chip memory. Won't work if kernel is larger than the chip memory size or on the DraCo.

Note: Because the loadbsd program can only read kernels from a AmigaOS filesystem, the file */netbsd* is often not the same as the actual kernel booted. This can cause some programs to fail. However, note that you can use third-party Berkeley filesystems such as bffs to access the NetBSD root partition from AmigaOS.

FILES

/netbsd	system kernel
/usr/mdec/bootxx_ffs	RDB device primary boot block
/usr/mdec/bootxx_fd	floppy disk primary boot block
/usr/mdec/boot.amiga	secondary bootstrap
/boot.amiga	secondary bootstrap (installed)

SEE ALSO

`ddb`(4), `fsck_ffs`(8), `installboot`(8), `newfs`(8), `savecore`(8), `shutdown`(8)

BUGS

Due to code size restrictions, you can't currently use an old-style file system (created with `newfs`(8) -O or with NetBSD 0.9) with the boot block. You can use **`loadbsd`** to boot from AmigaOS, or upgrade the file system with *`fsck_ffs -c 2`*.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

When the NetBSD kernel is booted normally (using one of the two methods discussed below), it initializes itself and proceeds to boot the system. An automatic consistency check of the file systems takes place, and unless this fails, the system comes up to multi-user operations. The proper way to shut the system down is with the `shutdown`(8) command.

If the system crashes, it will enter the kernel debugger, `ddb`(4), if it is configured in the kernel. If the debugger is not present, or the debugger is exited, the system will attempt a dump to the configured dump device (which will be automatically recovered with `savecore`(8) during the next boot cycle). After the dump is complete (successful or not), the system will attempt a reboot.

Booting NetBSD using the bootloader

When a bootable NetBSD partition is created by means of `installboot`(8) and the boot-preference bit in the NVRAM is either invalid or set to NetBSD , the Atari BIOS will automatically start the NetBSD bootloader. By default it will load the kernel image `/netbsd` and attempts to boot it into multi-user mode. This behaviour can be changed by either keeping the `Alternate` or the `Right-Shift` key pressed during the boot. When the `Alternate` key is pressed, the bootstrap is aborted, causing the BIOS to continue scanning the disks for a bootable partition (this is compatible with AHDI 3.0). Pressing the `Right-Shift` key during the boot, causes the boot loader to enter the interactive mode. In interactive mode, the command line looks like:

```
[OS-type][boot-path][boot-options]
```

Each component of the command can be omitted in which case the defaults indicated will be used.

OS-type:

.netbsd (the default)
.linux
.asv
.tos

If something other than `.netbsd` is specified, control is returned to the BIOS with the boot preference set to the selected type. Due to limitations of the BIOS however, the search for bootblocks is continued rather than restarted.

boot-path: This gives you the opportunity to boot another kernel, say: `/netbsd.old`. The default is `/netbsd`

boot-options: These options are a subset of the `loadbsd`(8) options.

−a	Boot into multi-user mode (the default)
−b	Prompt for the root file system device, the system crash dump device, and the path to `init`(8).
−d	Enter the kernel debugger
−q	Boot in quiet mode
−v	Boot in verbose mode

Booting using the loadbsd program

When you want (or have to) start NetBSD from GEM, you have to use the `loadbsd`(8) program that is supplied on the kernel-floppy. The loadbsd command line specification is:

loadbsd [**−abdhqstvwDV**] [**−S** *amount*] [**−T** *amount*] *kernel-path*

Description of options:

−a Boot automatically into multi-user mode.

−b Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

−d Enter the kernel debugger after booting.

−h Print a help screen that tries to explain the same options as mentioned here.

−o *outputfile*
Write all output to the file *outputfile*.

−q Boot in quiet mode.

−s Tell NetBSD only to use ST compatible RAM.

−t Test loading of the kernel but don't start NetBSD.

−v Boot in verbose mode.

−w Wait for a keypress before exiting loadbsd. This is useful when starting this program under GEM.

−D Show debugging output while booting the kernel.

−S *amount*
Set the amount of available ST compatible RAM in bytes. Normally this value is set automatically from the values initialized by the BIOS.

−T *amount*
Set the amount of available TT compatible RAM in bytes. Normally this value is set automatically from the values initialized by the BIOS.

−V Print the version of `loadbsd`(8) that you are using.

kernel-path
This is a GEMDOS path specification of the kernel to boot.

Note: Because the loadbsd program can only read kernels from a GEMDOS filesystem, the file */netbsd* is usually not the same as the actual kernel booted. This can cause some programs to fail.

FILES

`/netbsd` system kernel

SEE ALSO

`ddb`(4), `savecore`(8), `shutdown`(8)

NAME

boot — system bootstrapping procedures

DESCRIPTION

Cobalt Networks' MIPS-based Microservers (now known as Sun Server Appliances) that can run NetBSD/cobalt can use any of the following boot procedures:

- bootstrap NetBSD from disk using the standard Cobalt Firmware boot sequence
- bootstrap NetBSD from disk using the NetBSD boot loader
- network bootstrap NetBSD using the standard Cobalt Firmware means from a TCP/IP LAN with DHCP and NFS.

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cobalt Boot Sequence

The first program to take a control after reboot or at power-on is the Cobalt Firmware. The Firmware can load a compressed kernel from disk, subject to a few limitations. The Firmware expects the disk to contain DOS-style partition information with the first partition being a boot one which is special in that it should reside close to the beginning of the disk and must contain an ext2 file system with a `boot` directory which is treated specially by the Firmware. The default sequence is pretty straightforward, the Firmware finds the boot partition, mounts the ext2 file system from it and tries to load a compressed kernel image from the `boot` directory. The name of the kernel image differs from machine to machine and this is the reason for having multiple copies of NetBSD kernel installed under different names. The following kernel image names are known to be in use by certain Cobalt flavors:

```
/boot/vmlinuz.gz
/boot/vmlinux.gz
/boot/vmlinux-nfsroot.gz
/boot/vmlinux_RAQ.gz
/boot/vmlinux_raq-2800.gz
```

where `/boot` is the directory on the boot partition.

The Firmware console provides the means to alter the default boot sequence and/or to specify boot parameters. Pressing '⟨space⟩' right after the Firmware printed its greeting brings the Firmware console prompt and pressing '?' at the prompt prints a help screen with all commands supported by the Firmware. For example, the 'bfd' command can be used to boot a kernel image:

```
Cobalt: bfd /boot/<kernel image> [options]
```

where "options" are the kernel options.

Bootstrap from disk using the standard Firmware sequence

The Firmware enters the standard boot sequence after reboot or at power-on when no front-panel buttons are pressed and the Firmware console is not used to change the boot procedure. At boot time, the Firmware checks the hardware, prints the banner and performs the standard Cobalt boot sequence. There are a few culprits tightly connected to this boot method. First of all, the kernel must be compressed. Second, the Firmware enforces a hard restriction on the kernel size (it cannot exceed approximately 1000000/2500000 bytes compressed/uncompressed) resulting in a lock-up should this requirement not be fulfilled. For NetBSD, another pitfall is that the uncompressed kernel should be copied to the root directory to make certain system binaries (such as e.g. netstat) work, and the kernel images in the `boot` directory should always be in sync with the ones installed in the root directory.

Bootstrap from disk using the NetBSD boot loader

The NetBSD boot loader is an attempt to break through the limitations enforced by the Firmware loader. The main idea is to make the Firmware load the NetBSD boot loader and let the latter take care of loading the kernel. To achieve this goal, multiple copies of the boot loader are installed in the `boot` directory on the boot partition, one copy per each kernel image name the Cobalt Firmware might look for. The NetBSD kernel is located in the root directory (usually `/dev/wd0a`) like it is on other platforms. Once running, the boot loader prints a banner similar to the following:

```
>> NetBSD/cobalt 1.6ZG Bootloader, Revision 0.1 [@0x81000000]
>> (user@buildhost, builddate)
>> Memory:            32768 k
>> Firmware boot string:    root=/dev/hda1 ro
Loading: wd0a:netbsd
2249104+195856 [81872+73284]=0x27af90
Starting at 0x80001000
```

by default, the boot loader uses "`wd0a:netbsd`" as kernel specification which corresponds to the file **`netbsd`** on partition "a" of the NetBSD MBR partition of the first hard disk known to the Firmware (which is an IDE or similar device - see the **BUGS** section). In case this fails, the boot loader will try a few alternative kernel image names and if this also fails the loader will repeat the whole procedure for all other NetBSD slices (if any) and will load the first kernel image found.

Boot loader Options

It is possible to specify some options to the boot loader by breaking into the Firmware and using the "bfd" command:

```
Cobalt: bfd /boot/boot.gz [options]
```

The boot loader allows the following options:

`nbsd=` [*device*:][*filename*] [**`-acdqsv`**]

The default *device* will be set to *wd0a* which is the first NetBSD partition on the first drive, as numbered by the Firmware. To boot from an alternate disk, the full name of the device should be given at the prompt. *device* is of the form *xdNx* where *xd* is the device from which to boot, *N* is the unit number, and *x* is the partition letter.

The following list of supported devices may vary from installation to installation:

wd Hard disks as numbered by the BIOS. This includes ST506, IDE, ESDI, RLL disks on a WD100[2367] or lookalike controller(s), and SCSI disks on SCSI controllers recognized by the Firmware.

The default *filename* is `netbsd`; if the boot loader fails to successfully open that image, it then tries `netbsd.gz` (expected to be a kernel image compressed by `gzip`(1)), followed by `netbsd`, `netbsd.gz`, `onetbsd`, `onetbsd.gz`, `netbsd.bak`, `netbsd.bak.gz`, `netbsd.old`, `netbsd.old.gz`, `netbsd.cobalt`, `netbsd.cobalt.gz`, `netbsd.elf`, and finally `netbsd.elf.gz`. Alternate system images can be loaded by just specifying the name of the image.

Options are:

`-a` Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

`-c` Bring the system up into the device configuration manager. From here the device locators can be tuned to the hardware; see `userconf`(4).

-d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

-q Boot the system in quiet mode.

-s Bring the system up in single-user mode.

-v Boot the system in verbose mode.

It is always a good idea to have a small rescue kernel in the `boot` directory. In an emergency case, this will allow you to use the Firmware 'bfd' command to boot the rescue image:

```
Cobalt: bfd /boot/netbsd.gz
```

Network bootstrap using the standard Firmware sequence

The Cobalt Firmware allows to boot a kernel over the network, with all the limitations of the Firmware loader described above. The simplest method is to break into the Firmware prompt and use "bfd" command to specify where to boot from:

```
Cobalt: bfd /netbsd.gz nfsroot=/home/raq/root
```

The Firmware is picky about syntax and in general, so if things fail mysteriously, try to conform to the conventions described above. For netbooting, you need to NFS-export the directory given to "nfsroot=", and the named kernel (`netbsd.gz`) needs to be executable and in that directory. You will also need to setup `rarpd`(8) and `dhcpd`(8). Once the kernel is loaded with the command line values, the data given via DHCP is used to mount the root filesystem. Here is a known working DHCP entry:

```
host raq {
    hardware ethernet 0:10:e0:0:52:62;      # raq MAC
    fixed-address 10.0.0.15;                # raq address
    filename "/netbsd.gz";                  # kernel name in root-path
    option root-path "/home/raq/root";      # absolute dir on nfs server
    server-name="10.0.0.3";                 # IP of nfs server
}
```

Another option is to hold down the left and right cursor buttons during power-on which executes the command

```
bfd /boot/vmlinux.gz root=/dev/nfs nfsroot=/nfsroot,
```

resulting in a netboot. On RaQ 1's, the default kernel name is `vmlinux_RAQ.gz` and on RaQ 2's, it is `vmlinux_raq-2800.gz`.

FILES

`/boot/boot.gz`	boot program code loaded by the Firmware loader
`/boot/netbsd.gz`	`gzip`(1)-compressed rescue system code
`/netbsd`	system code
`/netbsd.gz`	`gzip`(1)-compressed system code

SEE ALSO

`ddb`(4), `userconf`(4), `fdisk`(8), `halt`(8), `reboot`(8), `shutdown`(8)

BUGS

The NetBSD boot loader supports booting off IDE hard drives only. This is less a bug of the boot loader code than a shortcoming of the Cobalt Firmare and shall be considered as such.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Dreamcast consoles can only boot from the built-in GD-ROM drive. Insert a bootable CD-R containing the NetBSD kernel and turn on the power.

FILES

`/netbsd` system code

SEE ALSO

ddb(4), userconf(4), halt(8), reboot(8), shutdown(8)

HISTORY

The **boot** man page appeared in NetBSD 2.0.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts On an HP300, the boot procedure uses the boot ROM to load a boot program from an LIF format directory at the beginning of an attached disk. The `/usr/mdec` directory contains a disk boot programs which should be placed in a new pack automatically by `newfs`(8) when the "a" partition file system on the pack is created.

This *boot* program finds the corresponding file on the given device (*netbsd* by default), loads that file into memory, and starts the program at the entry address specified in the program header.

The boot program can be interrupted by typing '^C' (ctrl-C). This will force the boot program to interactively prompt for a system to boot. If not interrupted, it will boot from the device from which the boot program itself was loaded.

The file specifications used for an interactive boot are of the form:

```
device(unit, minor)
```

where *device* is the type of the device to be searched, *unit* is 8 ∗ the HP-IB number plus the unit number of the disk or tape, and *minor* is the disk partition or tape file number. Normal line editing characters can be used when typing the file specification. Currently, "rd" and "sd" are the only valid *device* specifiers.

For example, to boot from the 'a' file system of unit 0 on HP-IB 2, type `rd(16, 0)netbsd` to the boot prompt. For tapes, the minor device number gives a file offset.

In an emergency, the bootstrap methods described in the paper *Installing 4.3bsd on the HP300* can be used to boot from a distribution tape.

FILES

`/netbsd`	system code
`/usr/mdec/bootrd`	LIF format boot block
`/usr/mdec/installboot`	program to install boot blocks

SEE ALSO

`halt`(8), `reboot`(8), `shutdown`(8)

NAME

boot — hp700 system bootstrapping procedures

DESCRIPTION

System starts

When powered on, after a panic, or if the system is rebooted via `reboot`(8) or `shutdown`(8), the hp700 firmware ("PDC") will proceed to its initialization, and will boot an operating system if autoboot is enabled.

Boot process description

System boot blocks are provided as a "LIF" (Logical Interchange Format) archive, either on a disk device, or via the network, using the *bootp* or *rboot* protocols, depending on the PDC version.

PDC concepts

If autoboot is enabled, the PDC will attempt to boot from the specified "boot path" value. If no "boot path" has been specified, the PDC will then scan for bootable devices and boot from the first found, after a few seconds allowing the user to interrupt the boot process. If autoboot is disabled, the PDC will enter interactive mode, after an optional device scan. In all cases, it is possible to enter interactive mode by holding the escape key during the selftests, or when prompted to do so to abort the current operation, unless the PDC has been configured in "secure mode".

ISL interaction

"ISL" stands for "Initial System Loader" and is the **boot** program in NetBSD. On all versions of the PDC except for the 712 and 725 models the "boot" command (see below) will be followed by the question: "Interact with IPL (Y, N, or Cancel)?>" where a positive answer will invoke an interactive prompt in the **boot** program later and negative will thus suppress it. A cancellation will abort the boot process.

On the 712 and 725 models firmware an additional "isl" argument should be given to the "boot" command to invoke the **boot** interactive prompt. The default behaviour is a non-interactive boot process.

Old PDC operation

This version is used on the following models: 705, 7x0, 715/33/50/75, 725/50/75, 735, 755. There are two levels of interactive commands in this version. The first level is a short menu:

```
b)   Boot from specified device
s)   Search for bootable device
a)   Enter Boot Administration mode
x)   Exit and continue boot sequence

Select from menu:
```

which provides the following commands:

b boot from a device found during the scan, either with its short "P#" form, or a complete name specification. For example, to boot from the SCSI disk with id 6 off the built-in (first) controller, one would enter **b** *scsi.6.0*.

s rescan for bootable devices.

a enter the second part of interactive mode.

x resume an interrupted boot sequence.

The "Boot Administration" mode, recognizable with its *BOOT_ADMIN>* prompt, controls the various boot options. The complete list of commands depends on the machine and PDC version. The following list only mentions commands impacting the boot process.

AUTOSELECT

Displays or changes the autoboot setting. If autoselect is set to "on", the PDC will always attempt to boot the first bootable device found in this order:

1. Boot device *path* setting.
2. SCSI devices connected to the built-in SCSI controller, the highest ID numbers being preferred.
3. Network *rboot* server (see also `rbootd`(8)).
4. Other SCSI devices connected to secondary controllers, the highest ID numbers being preferred.

If the *primary path* setting defines a bootable device, no device scan will occur.

BOOT

Boots off the specified device. It is similar to the **b** command from the short menu above. The "primary" and "alternate" path settings may be booted with **boot** *pri* and **boot** *alt* respectively.

PATH

Displays or changes the boot and console devices. The boot device is defined as the "primary" path, and another setting may be stored as the "alternate" path for rescue purposes. For example, to define the primary boot path to the SCSI disk with ID 5 connected to the built-in controller, one would enter **path primary** *scsi.5*

When invoked without parameters, **path** will list the various path settings.

Modern PDC operation

Machines equipped with 7100LC, 7200, or 7300LC CPU types are usually blessed with a different kind of PDC. There is only one interactive mode, with a *BOOT_ADMIN>* prompt, which provides both boot settings and commands. The complete list of commands depends on the machine and PDC version. The following list only mentions commands impacting the boot process.

auto boot

Displays or changes the autoboot setting. If **auto boot** is set to "on", the PDC will always attempt to boot. The booted device chosen will depend on the **auto search** setting.

auto search

Displays or changes the device scan setting. If **auto search** is set to "on", the PDC will attempt to boot the first bootable device found in this order:

1. Boot device *path* setting.
2. SCSI devices connected to the built-in SCSI controller, the highest ID numbers being preferred.
3. Network *bootp* server (see also `dhcpd`(8)).
4. Other SCSI devices connected to secondary controllers, the highest ID numbers being preferred.

If **auto search** is set to "off" and the primary boot path points to a bootable device, no device scan will occur.

Note that setting **auto search** to "on" will force autoboot, regardless of the **auto boot** value.

boot

Boots off the specified device. The "primary" and "alternate" path settings may be booted with **boot** *pri* and **boot** *alt* respectively.

path

Displays or changes the boot and console devices. The boot device is defined as the "primary" path, and another setting may be stored as the "alternate" path for rescue purposes. For example, to define the primary boot path to the SCSI disk with ID 5 connected to the built-in controller, one would enter **path pri** *scsi.5*.

When invoked without parameters, **path** will list the various path settings.

Abnormal system termination

If the system crashes, it will enter the kernel debugger, `ddb`(4), if it is configured in the kernel. If the crash occurred during initialization and the debugger is not present or is exited, the kernel will halt the system. If the crash occurred during normal operation and the debugger is not present or is exited, the system will attempt a dump to the configured dump device (which will be automatically recovered with `savecore`(8) during the next multi-user boot cycle), and after the dump is complete (successful or not) the kernel will attempt a reboot.

FILES

`boot.lif`	network bootstrap and kernel combined image
`/netbsd`	default NetBSD system kernel
`/usr/mdec/xxboot`	primary bootstrap for “ffs” file system
`/usr/mdec/boot`	system bootstrap (usually also installed as `/boot`)

SEE ALSO

`ddb`(4), `dhcpd`(8), `halt`(8), `init`(8), `installboot`(8), `rbootd`(8), `reboot`(8), `savecore`(8), `shutdown`(8)

NAME

boot — system bootstrapping procedures

DESCRIPTION

Windows CE machines with StrongARM CPUs use the hpcboot(8) program to boot NetBSD.

Power fail and crash recovery

Unfortunately, NetBSD can't reboot itself at power-up or after crashes. The machine will go through the cold reset and boot into Windows CE. You will have to restart NetBSD manually using hpcboot(8).

Once NetBSD starts, an automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts

On cold reset Windows CE handheld machines attempt to boot the Windows CE operating system from the boot ROM. The boot ROM is usually not rewritable, so you cannot erase or damage Windows CE image.

You can't boot NetBSD directly, skipping Windows CE. The NetBSD bootloader, hpcboot(8), is provided as a Windows CE application program instead. Though the bootloader is an application program, it blows the entire running Windows CE, its data, and its settings away from RAM (but not ROM!) when the kernel boots successfully. If NetBSD is halted the machine will go through the cold reset and will reboot into Windows CE.

Normal Operation

Please, refer to the hpcboot(8) manual page.

FILES

hpcboot.exe	bootloader program for Windows CE

SEE ALSO

hpcboot(8)

BUGS

There is no general way to launch the bootloader automatically, as only a few Windows CE machines provide an "auto run" mechanism.

This port doesn't support kloader(4), which means that when the system is rebooted, it goes back to Windows CE.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

Unfortunately, on most machines, the system can't reboot itself at power-up or after crashes. You might have to restart the system manually. Once the system starts, an automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts

Typical MIPS based Windows CE Handheld machines attempt to boot Windows CE operating system in its own boot ROM. You can't boot the NetBSD directly skipping Windows CE. The NetBSD bootloaders are provided as application programs on Windows CE instead. You can choose pbsdboot(8) or hpcboot(8). Though the bootloaders are application programs, they blow away the entire Windows CE OS and its settings when the kernel boots successfully.

Normal Operation

Once running, a familiar window will appear. You can choose the machine type, kernel file location and kernel boot options with a GUI and push the button named "[boot]" to boot NetBSD.

Automatic mode

The bootloaders have an "auto boot" option. If you enable this option, the specified kernel will be loaded automatically after a countdown.

FILES

/netbsd	system kernel
/netbsd.gz	gzip-compressed kernel
pbsdboot1.exe	bootloader executable file for Windows CE version 1.01
pbsdboot.exe	bootloader executable file for Windows CE
hpcboot.exe	new bootloader executable file for Windows CE

SEE ALSO

kloader(4), hpcboot(8), pbsdboot(8)

BUGS

There is no general way to launch a bootloader automatically while a few Windows CE machine provide an "auto run" mechanism.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Windows CE machines with SuperH CPUs use the hpcboot(8) program to boot NetBSD. Once running, NetBSD can reboot itself if kloader(4) is configured in the kernel.

Power fail and crash recovery

Unfortunately, NetBSD can't reboot itself at power-up or after crashes. The machine will go through the cold reset and boot into Windows CE. You will have to restart NetBSD manually using hpcboot(8).

Once NetBSD starts, an automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts

On cold reset Windows CE handheld machines attempt to boot the Windows CE operating system from the boot ROM. The boot ROM is usually not rewritable, so you cannot erase or damage Windows CE image.

You can't boot NetBSD directly, skipping Windows CE. The NetBSD bootloader, hpcboot(8), is provided as a Windows CE application program instead. Though the bootloader is an application program, it blows the entire running Windows CE, its data, and its settings away from RAM (but not ROM!) when the kernel boots successfully. If NetBSD is halted the machine will go through the cold reset and will reboot into Windows CE.

Normal Operation

Please, refer to the hpcboot(8) manual page.

FILES

hpcboot.exe	bootloader program for Windows CE

SEE ALSO

kloader(4), hpcboot(8)

BUGS

There is no general way to launch the bootloader automatically, as only a few Windows CE machines provide an "auto run" mechanism.

NAME

boot — system bootstrapping procedures

DESCRIPTION

IA-32 computers (the IBM PC and its clones) that can run NetBSD/i386 can use any of the following boot procedures, depending on what the hardware and BIOS support:

boot	bootstrap NetBSD from the system BIOS
dosboot(8)	bootstrap NetBSD from MS-DOS
w95boot(8)	bootstrap NetBSD from Windows 95
pxeboot(8)	network bootstrap NetBSD from a TCP/IP LAN with DHCP, TFTP, and NFS.

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts

The 386 PC AT clones attempt to boot the floppy disk drive A (otherwise known as drive 0) first, and failing that, attempt to boot the hard disk C (otherwise known as hard disk controller 1, drive 0). The NetBSD boot-blocks are loaded and started either by the BIOS, or by a boot selector program (such as OS-BS, BOOTEASY, the OS/2 Boot Menu or NetBSD's boot-selecting master boot record - see mbr(8)).

Normal Operation

Once running, a banner similar to the following will appear:

```
>> NetBSD BIOS Boot, revision 3.0
>> (user@buildhost, builddate)
>> Memory: 637/15360 k
Press return to boot now, any other key for boot menu
booting hd0a:netbsd - starting in 5
```

After a countdown, the system image listed will be loaded. In the example above, it will be "hd0a:netbsd" which is the file **netbsd** on partition "a" of the NetBSD MBR partition of the first hard disk known to the BIOS (which is an IDE or similar device - see the **BUGS** section).

Pressing a key within the time limit, or before the boot program starts, will enter interactive mode. When using a short or 0 timeout, it is often useful to interrupt the boot by holding down a shift key, as some BIOSes and BIOS extensions will drain the keystroke buffer at various points during POST.

If present, the file /boot.cfg will be used to configure the behaviour of the boot loader including setting the timeout, choosing a console device, altering the banner text and displaying a menu allowing boot commands to be easily chosen. See boot.cfg(5).

Diagnostic Output

If the first stage boot fails to load the boot, it will print a terse message indicating the reason for the failure. The possible error messages and their cause are listed in mbr(8).

If the first stage boot succeeds, the banner will be shown and the error messages should be self-explanatory.

Interactive mode

In interactive mode, the boot loader will present a prompt, allowing input of these commands:

boot [*device*:][*filename*] [**−acdqsvxz**]

The default *device* will be set to the disk that the boot loader was loaded from. To boot from an alternate disk, the full name of the device should be given at the prompt. *device* is of the form *xd* [*N*[*x*]] where *xd* is the device from which to boot, *N* is the unit number, and *x* is the partition letter.

The following list of supported devices may vary from installation to installation:

hd	Hard disks as numbered by the BIOS. This includes ST506, IDE, ESDI, RLL disks on a WD100[2367] or lookalike controller(s), and SCSI disks on SCSI controllers recognized by the BIOS.
fd	Floppy drives as numbered by the BIOS.

The default *filename* is `netbsd`; if the boot loader fails to successfully open that image, it then tries `netbsd.gz` (expected to be a kernel image compressed by gzip), followed by `netbsd.old`, `netbsd.old.gz`, `onetbsd`, and finally `onetbsd.gz`. Alternate system images can be loaded by just specifying the name of the image.

Options are:

−a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

−c Bring the system up into the device configuration manager. From here the device locators can be tuned to the hardware; see `userconf`(4).

−d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

−q Boot the system in quiet mode.

−s Bring the system up in single-user mode.

−v Boot the system in verbose mode.

−x Boot the system with debug messages enabled.

−z Boot the system in silent mode.

consdev *dev*

Immediately switch the console to the specified device *dev* and reprint the banner. *dev* must be one of *pc*, *com0*, *com1*, *com2*, *com3*, *com0kbd*, *com1kbd*, *com2kbd*, *com3kbd*, or *auto*. See **Console Selection Policy** in `boot_console`(8).

dev [*device*]

Set the default drive and partition for subsequent filesystem operations. Without an argument, print the current setting. *device* is of the form specified in **boot**.

help

Print an overview about commands and arguments.

ls [path]

Print a directory listing of `path`, containing inode number, filename, and file type. `path` can contain a device specification.

quit

Reboot the system.

In an emergency, the bootstrap methods described in the NetBSD installation notes for the i386 architecture can be used to boot from floppy or other media, or over the network.

FILES

`/boot`	boot program code loaded by the primary bootstrap
`/boot.cfg`	optional configuration file
`/netbsd`	system code
`/netbsd.gz`	gzip-compressed system code
`/usr/mdec/boot`	master copy of the boot program (copy to /boot)
`/usr/mdec/bootxx_fstype`	primary bootstrap for filesystem type fstype, copied to the start of the NetBSD partition by `installboot`(8).

SEE ALSO

`ddb`(4), `userconf`(4), `boot.cfg`(5), `boot_console`(8), `dosboot`(8), `halt`(8), `installboot`(8), `mbr`(8), `pxeboot`(8), `reboot`(8), `shutdown`(8), `w95boot`(8)

BUGS

Any *`filename`* specified after the boot options, e.g.:

```
boot -d netbsd.test
```

is ignored, and the default kernel is booted.

Hard disks are always accessed by BIOS functions. Unit numbers are BIOS device numbers which might differ from numbering in the NetBSD kernel or physical parameters (e.g., SCSI slave numbers). There isn't any distinction between "sd" and "wd" devices at the bootloader level. This is less a bug of the bootloader code than a shortcoming of the PC architecture. The default disk device's name printed in the starting message is derived from the "type" field of the NetBSD disklabel (if it is a hard disk).

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery Normally, the NetBSD kernel on the mac68k architecture is booted from the native operating system by means of an application program. When the kernel takes over, it initializes itself and proceeds to boot the system. An automatic consistency check of the file systems takes place, and unless this fails, the system comes up to multi-user operations. The proper way to shut the system down is with the `shutdown`(8) command.

If the system crashes, it will enter the kernel debugger, `ddb`(4), if it is configured in the kernel. If the debugger is not present, or the debugger is exited, the system will attempt a dump to the configured dump device (which will be automatically recovered with `savecore`(8) during the next boot cycle). After the dump is complete (successful or not), the system will attempt a reboot.

On most mac68k machines with "soft-power" after the IIcx, the power switch can be physically rotated and locked in the 'on' position. The native OS can be configured to automatically start the NetBSD boot program. Additionally, the NetBSD boot program can be configured to boot NetBSD without intervention. When a system is so configured, it can crash or lose power and reboot back to a fully multi-user state without any intervention.

The boot application The boot application runs in the native OS on the system. It has a dialog where booting preferences may be changed and an option whereby these options may be saved. The preferences are stored in the program itself, not in a preferences folder--thus allowing two separate copies of the program to be configured differently (e.g. to boot different netbsd or netbsd.test, or to boot from two different drives).

One option that may be specified is a boot to single-user mode. This stops the boot process very early on and allows system maintenance. If one wishes to provide some security at this phase of the boot, remove the `secure` option from ttye0 in the `ttys`(5) file.

Another useful option that may be specified is the "serial console" option. This will allow a serial device (terminal or computer) to act as a console for the system. This device must be configured to use 9600 baud, eight bits, no parity, and one stop bit (9600-N81). Either the printer port or the modem port (tty01 and tty00, respectively) may be used for this.

It is sometimes useful to boot a kernel that resides in a folder in native OS rather than from the usual location in the NetBSD file system. A radio button is supplied for this purpose. Note that some programs will not run properly if the kernel is not found as */netbsd* within the NetBSD file system.

FILES

`/netbsd` system kernel

SEE ALSO

`ddb`(4), `ttys`(5), `savecore`(8), `shutdown`(8)

NAME

boot — Macppc system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in `fsck`(8), and unless this fails, the system will resume multi-user operations.

Cold starts

The boot ROM performs a Power On Self Test (POST) then loads Open Firmware. Depending on the Open Firmware variable '`auto-boot?`' it will either stop at the Open Firmware prompt or attempt to boot an operating system. Depending on the contents of the '`use-nvramrc?`', '`boot-command`', '`boot-device`', and '`boot-file`' Open Firmware variables, it will attempt to boot MacOS, MacOS X, or NetBSD.

To boot NetBSD, Open Firmware loads the bootloader `ofwboot`(8) from the specified '`boot-device`'. The bootloader then loads the kernel from the '`boot-file`', (if it exists). Otherwise, it tries to load (in the following order): `netbsd`, `netbsd.gz`, or `netbsd.macppc` on the "a" partition of the same device that had the bootloader.

Open Firmware Commands

An essential but incomplete list of Open Firmware commands follows. A more thorough list is contained in the FAQ.

http://www.NetBSD.org/ports/macppc/faq.html#ofw-use

boot [*boot-device* [*boot-file*]] [*options*]

Boot an operating system. The default arguments for this command are taken from the Open Firmware environment variables:

boot-device	primary bootloader location
boot-file	kernel location
options	flags passed to the kernel

reset-all

Reset the system, and proceed as specified by the '`use-nvramrc?`' and '`auto-boot?`' variables. If '`use-nvramrc?`' is set to '`true`', then the system will attempt to execute the commands stored in the '`nvramrc`' variable. If '`auto-boot?`' is set to '`true`', the system will attempt to use the values stored in '`boot-command`', '`boot-device`', and '`boot-file`' to boot the system. If '`auto-boot?`' is set to '`false`', the system will halt at the Open Firmware prompt.

shut-down

Power off the system.

setenv *variable value*

Set an Open Firmware variable, e.g.,

```
setenv auto-boot? false
setenv boot-device hd:,\ofwboot.xcf
setenv boot-file netbsd-GENERIC.gz
```

set-default *variable*

Set an Open Firmware variable to its default value.

printenv [*variable*]

Show Open Firmware variables and values.

eject fd

Eject floppy disk on systems with on-board floppy drives.

mac-boot

Attempt to boot MacOS on an Open Firmware 3 system.

bye

Attempt to boot MacOS on an Open Firmware 1.0.5, 2.0.x, or 2.4 system.

Open Firmware Variables

An essential but incomplete list of Open Firmware variables follows. A more thorough list is contained in the FAQ.

http://www.NetBSD.org/ports/macppc/faq.html#ofw-variables

auto-boot?	What Open Firmware will do at system startup or reset: *true* automatically bootstrap an operating system using values from the 'boot-command', 'boot-device', and 'boot-file' variables. *false* stop at the Open Firmware prompt.
use-nvramrc?	If 'true' runs commands in variable 'nvramrc'.
real-base	Kernel memory location. *Do not modify this value on Open Firmware 3 systems — you may damage your computer*. All other Open Firmware versions should use F00000.
load-base	Bootloader memory location. *Do not modify this value on Open Firmware 3 systems — you may damage your computer*. All other Open Firmware versions should use 600000.
boot-command	The command to use for booting. Typically, the default of 'boot' is used.
boot-device	Device from which to load primary bootloader. Value depends on a variety of factors. See ofwboot(8).
boot-file	Kernel location. Value depends on a variety of factors. See ofwboot(8).
input-device	What type of console input device (ADB keyboard, USB keyboard, or serial port) . *kbd* ADB keyboard on models with ADB, USB keyboard on models with USB, and built-in keyboard on laptops. This is the default on some Open Firmware 2.0.x machines and all Open Firmware 2.4 and 3 machines. *ttya* 'Modem' serial port on machines with serial ports. Properties are 38400 bps, 8 bits, no parity, 1 stop bit, no handshaking. This is the default on all Open Firmware 1.0.5 systems and some Open Firmware 2.0.x systems. *ttyb* 'Printer' serial port on machines with serial ports. Properties are the same as the 'Modem' port. *scca* Serial port on Xserve models. Properties are 57600 bps, 8 bits, no parity, 1 stop bit, no handshaking.

output-device	What type of console output device (On-board video, AGP video, PCI video, built-in LCD, or serial console). Value depends on a variety of factors. See ofwboot(8) and **http://www.NetBSD.org/ports/macppc/faq.html#ofw-input-output-devices**
nvramrc	If 'use-nvramrc?' is set to true, these FORTH commands will be run when the computer is reset

Normal Operation

When Open Firmware loads the primary bootloader, it will print something like the following:

```
loading XCOFF
tsize=CC50 dsize=14AC bsize=2668 entry=640000
SECTIONS:
.text    00640000 00640000 0000CC50 000000E0
.data    0064D000 0064D000 000014AC 0000CD30
.bss     0064E4B0 0064E4B0 00002668 00000000
loading .text, done..
loading .data, done..
clearing .bss, done..
```

When ofwboot(8) is started, it prints something like the following:

```
>> NetBSD/macppc OpenFirmware Boot, Revision 1.7
>> (autobuild@tgm.daemon.org, Thu Feb  6 17:50:27 UTC 2003)
```

When ofwboot(8) is loading the kernel, it prints something like the following:

```
4395364+254568 [220144+193803]=0x4d477c
 start=0x100000
```

When the NetBSD kernel has started it prints a banner similar to the following:

```
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

NetBSD 1.6ZC (GENERIC) #0: Tue Sep 30 13:09:10 UTC 2003
       autobuild@tgm.NetBSD.org:/autobuild/HEAD/macppc/OBJ/autobuild/HEAD/src/s
```

After bootstrap

Once the NetBSD/macppc kernel is booted normally it initializes itself and proceeds to start the system. An automatic consistency check of the file systems takes place, and unless this fails, the system comes up to multi-user operation.

The proper way to shut the system down is with the shutdown(8) command.

If the system crashes, it will enter the kernel debugger, ddb(4), if it is configured in the kernel. If the crash occurred during initialization and the debugger is not present or is exited, the kernel will halt the system.

If the crash occurred during normal operation and the debugger is not present or is exited, the system will attempt a dump to the configured dump device (which will be automatically recovered with savecore(8) during the next bootstrap cycle), and after the dump is complete (successful or not) the kernel will attempt a reboot.

FILES

`/boot`	NetBSD secondary bootstrap program (Open Firmware 1.x and 2.x)
`/netbsd`	default NetBSD system kernel
`/usr/mdec/bootxx`	NetBSD primary bootstrap program (Open Firmware 1.x and 2.x) a.k.a. "partition zero" bootloader
`/usr/mdec/ofwboot`	NetBSD secondary bootstrap program (Open Firmware 1.x and 2.x)
`/usr/mdec/ofwboot.xcf`	primary bootstrap for netboot and "cd9660" (ISO 9660), "MS-DOS", "HFS", and "HFS+" file systems.

SEE ALSO

`ddb`(4), `intro`(4), `diskless`(8), `halt`(8), `init`(8), `installboot`(8), `ofwboot`(8), `rc`(8), `reboot`(8), `savecore`(8), `shutdown`(8)

http://www.NetBSD.org/ports/macppc/faq.html
http://www.NetBSD.org/docs/network/netboot/

STANDARDS

IEEE Std 1275-1994 ("Open Firmware")
http://playground.sun.com/1275/home.html

BUGS

The device names used by NetBSD/macppc and Open Firmware often have no relation to each other.

Apple Computer's Open Firmware implementation is easily confused. It is best to reboot your computer after a failed boot attempt, **`halt`**, or **`shutdown -h`**. Use the Open Firmware **`reset-all`** command.

Apple Computer's Open Firmware implementation is notoriously bad. Thorough instructions for installing and booting NetBSD are in the install notes (`INSTALL.html`) included with every release of NetBSD.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in `fsck`(8). and unless this fails, the system will resume multi-user operations.

Cold starts from disk

The disk-boot program (`/usr/mdec/bootsd`) will attempt to load `netbsd` from partition A of the boot device, which must currently be an "sd" disk.

Cold starts from tape

The tape-boot program (`/usr/mdec/bootst`) will attempt to load `netbsd` from a SCSI tape drive.

Cold starts over a network

The network boot program (`/usr/mdec/netboot`) will load `netbsd` from the NFS root as determined by the procedure described in `diskless`(8). Note that the MVME147 is unable to boot directly from the network without the help of a small bootloader program (`/usr/mdec/sboot`).

Boot program options

-a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

-d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

-q Boot the system in quiet mode.

-s Bring the system up in single-user mode.

-v Boot the system in verbose mode.

Any extra flags or arguments, or the <*boot string*> after the -- separator are passed to the boot PROM. Other flags are currently ignored.

At any time you can break to the kernel debugger `ddb`(4) (assuming **options DDB** was specified in the kernel configuration file) by sending a serial line BREAK character. If you do this accidentally you can continue whatever was in progress by typing 'c' followed by the return key.

FILES

`/netbsd`	system code
`/usr/mdec/bootxx`	first-level boot block for disks
`/usr/mdec/stboot`	first-level boot block for tapes
`/usr/mdec/bootsd`	second-level boot block for UFS disks
`/usr/mdec/bootst`	second-level boot block for tapes
`/usr/mdec/netboot`	boot program for NFS (diskless) boot
`/usr/mdec/sboot`	network bootstrap program for MVME147
`/usr/mdec/installboot`	program to install bootxx on a disk

SEE ALSO

`disklabel`(8), `fsck`(8), `halt`(8), `init`(8), `rc`(8), `shutdown`(8), `syslogd`(8)

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts On an NeXT, the boot procedure uses the boot ROM to load a boot program over the network using BOOTP and TFTP. The `/usr/mdec` directory contains a network boot program which should be made available via `tftp`(1). The network boot program will load `netbsd` from the NFS root as determined by the procedure described in `diskless`(8).

FILES

`/netbsd`	system code
`/usr/mdec/boot`	boot program for NFS (diskless) boot

SEE ALSO

`halt`(8), `init`(8), `rc`(8), `reboot`(8), `shutdown`(8)

NAME

boot — system bootstrapping procedures

DESCRIPTION

The NetBSD kernel is started by placing it near the beginning of physical memory and transferring to the entry point. Since the system is not reenterable, it is necessary to read it in from disk or tape each time it is to be bootstrapped.

Power fail and crash recovery

Normally, the system will boot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts

At power up, all DECstation ROMs consult the **haltaction** environment variable in EEPROM to determine whether or not to attempt to boot automatically. If this variable is set to 'h', the ROM prints a prompt on the console and waits for user commands. If set to 'b', the ROM attempts to autoboot.

DECSTATION 2100 and 3100

On the DECstation 2100 and 3100, the path used for automatic booting is stored in the **bootpath** environment variable. The path is made up of a device type specifier (e.g., rz, tz, mop or tftp) followed by a triplet in the form (x,y,z), followed by a filename to load.

Within the triplet, x is the controller (always 0), y is the SCSI id of the drive to boot from or 0 for net boots, and z is the partition to boot from (usually 0 for SCSI devices, always zero for network booting). For both disk and network boots, () may be specified instead of (0,0,0).

The filename is optional for bootp/tftp and mop booting, since in these cases the network protocol can be used to determine which file to boot. When booting off the tape, no filename should be specified. When booting off of disk, the filename is optional but is usually specified. If no filename is specified when booting off disk, the following filenames are tried in order: **netbsd.pmax**, **netbsd**, **netbsd.gz**, **netbsd.bak**, **netbsd.old**, **onetbsd**, **gennetbsd**. Generally, the kernel is named **netbsd**.

An example bootpath setting would be:

```
setenv bootpath rz(0,1,0)netbsd
```

At the PROM prompt, the user may boot NetBSD with either the **auto** or the **boot** command. If the **auto** command is used, the **−a** argument is passed to the kernel, requesting a multi-user boot; otherwise the **−s** argument is passed, requesting that NetBSD boot to single user mode.

When either the **boot** or the **auto** command is issued with no arguments, the kernel specified in the bootpath environment variable is booted. With the **boot** command, an alternative kernel may be specified with the **−f** flag, followed by the path of the kernel to boot, as described above. For example:

```
boot -f rz(0,4,0)netbsd.new
```

TURBOCHANNEL DECstations

On TurboChannel machines (all DECstation 5000 models), the boot path is specified in the boot environment variable, along with any arguments to be passed to the kernel. Note that to specify boot arguments (e.g., **−a**) when setting the **boot** environment variable, the filename and arguments must be enclosed in quotes. For example:

```
setenv boot "3/rz4/netbsd -a"
```

The device from which to boot is specified as the TurboChannel slot number, a TurboChannel-option-specific device name, and a path to the file to load, all separated by slashes. You can get a list of the devices installed in your TurboChannel slots (as well as any built-in devices which appear as TurboChannel slots) by typing the **cnfg** command at the boot prompt. You can get more detailed information about a specific TurboChannel option by typing **cnfg** followed by the slot number of that option.

For SCSI devices, the option-specific device identifier is either rz# for disks or tz# for tapes, where # is the SCSI id of the device. For network devices, the option-specific protocol identifier is either mop or tftp. Filename requirements are as for the DECstation 2100 and 3100.

To start NetBSD from the boot prompt, the **boot** command must be used. With no arguments, this simply boots the default kernel with the default arguments as set with **setenv boot**. If no boot environment variable is set or if an alternative kernel is to be booted, the path of that kernel may be specified after the boot command as described above, and any arguments may be passed similarly. For example:

```
boot 3/rz4/netbsd.new -a
```

KERNEL ARGUMENTS

The kernel supports the following arguments:

a	Autoboot -- try and boot to multi-user mode without further input.
m	Use a miniroot already present in memory.
n	Prompt for the root file system device, the system crash dump device, and the path to init(8).
N	Do not prompt for the root file system device, the system crash dump device, and the path to init(8). If the configured-in devices are present, use them.
s	Boot only to single-user mode.

Since DECstation PROMs also parse any arguments with a leading "-", and reject unrecognized options, arguments other than "a" or "s" should be specified after the kernel name with no leading "-". For example:

```
boot 3/rz4/netbsd ns
```

SEE ALSO

ddb(4), halt(8), init(8), installboot(8), rc(8), reboot(8), savecore(8), shutdown(8)

HISTORY

The **boot** command is currently under development.

NAME

boot — system bootstrapping procedures

SYNOPSIS

boot

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in fsck(8), and unless this fails, the system will resume multi-user operations.

Cold starts

The prep architecture does not allow the direct booting of a kernel from the hard drive. Instead it requires a complete boot image to be loaded. This boot image contains a NetBSD kernel, which will then provide access to the devices on the machine. The image can be placed on any device that the firmware considers a bootable device. Usually this is either a SCSI disk, tape, CD-ROM, or floppy drive.

Boot program options

The prep architecture and bootloader does not support any option parsing at the boot prompt.

Boot partition

The prep port requires a special boot partition on the primary boot device in order to load the kernel. This partition consists of a PC-style i386 partition label, a small bootloader, and a kernel image. The prep firmware looks for a partition of type 0x41 (65) and expects the bootloader, immediately followed by the kernel, to be there. The mkbootimage(8) command needs to be used to generate this image.

FILES

/netbsd	system code
/usr/mdec/boot	system bootstrap
/usr/mdec/boot_com0	system bootstrap with serial console

SEE ALSO

disklabel(8), fsck(8), halt(8), init(8), installboot(8), mkbootimage(8), rc(8), shutdown(8), syslogd(8)

NAME

boot — sgimips system bootstrapping procedures

DESCRIPTION

Silicon Graphics MIPS-based computers all feature essentially similar firmware systems. However, as of the Indigo R4x00 series (IP20), quasi- ARCS (Advanced RISC Computing Specification) compatible features are also present. All known PROM implementations support loading executables from disk devices, as well as from the network via BOOTP and TFTP.

Disk Booting

SGI provides a small filesystem at the beginning of each bootable disk called a Volume Header, which contains a boot loader and other standalone utilities. Booting NetBSD requires that we write our bootloader into to the volume header using sgivol(8).

Once a bootloader is present in the volume header, it may be executed directly by the PROM either manually, or at boot time using the "OSLoader" PROM environment variable. The NetBSD bootloader will obtain the kernel filename to boot from the PROM or EEPROM. This is specified by setting the PROM environment variable "OSLoadFilename" to an appropriate value. For instance, "/netbsd.ecoff".

For example, the following will configure the PROM to use the bootloader "aoutboot" to load the kernel "netbsd.old"

```
setenv OSLoader aoutboot
setenv OSLoadFilename netbsd.old
```

Network Booting

The system firmware will obtain an IP address, TFTP server address, and an optional filename from the BOOTP server and download it via TFTP. The PROM's configurable network address environment variable "netaddr" must match the address provided by the BOOTP server.

An example BOOTP entry for dhcpd(8) follows:

```
host indigo3k {
        hardware ethernet 08:00:69:42:42:42;
        fixed-address 192.168.0.2;
        option host-name "indigo3k.foo";
        #filename "/netbsd.ecoff";
        next-server 192.168.0.1;
        option root-path "/export/indigo3k/root";
        server-name "192.168.0.1";
}
```

To boot a kernel named "netbsd.ecoff" the user would type:

```
boot -f bootp():/netbsd.ecoff
```

See dhcpd.conf(5) for more information on configuring as a BOOTP server.

SEE ALSO

dhcpd.conf(5), dhcpd(8), sgivol(8)

CAVEATS

Some older PROM revisions do not support loading of ELF images. The build system automatically prepares ECOFF versions, which are correctly interpreted.

BUGS

NetBSD does not support booting from disk on systems lacking an ARCS-compatible firmware (presently supported systems include Personal Iris and Indigo R3000). It is possible to work around this by creating a sufficiently large volume header and placing the kernel in it, or by network booting.

Some firmware revisions have a bug, which precludes them from communicating with TFTP servers using ports above 32767. When using NetBSD as the TFTP server, this problem may be worked around as follows:

```
sysctl -w net.inet.ip.anonportmin=20000
sysctl -w net.inet.ip.anonportmax=32767
```

Another bug exists in some firmware revisions, which precludes the PROM from communicating with TFTP servers that employ PMTU (Path MTU) discovery. This bug may be worked around by disabling PMTU on the TFTP server. This does not presently affect NetBSD servers.

This man page is horribly incomplete.

NAME

boot — system bootstrapping procedures

SYNOPSIS

boot [**−adqsv**] [-- ⟨*boot string*⟩]

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in `fsck`(8), and unless this fails, the system will resume multi-user operations.

Cold starts

The Sun boot firmware, either old-style or new-style (Open Boot Prom), performs a Power On Self Test (POST), and then will boot an operating system according to configuration in Open Firmware environment variables.

Boot program options

−a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

−d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `gdb`(1).

−C Boot kernel in compat mode. Starting with revision 1.14 (introduced on 2003/03/01), the sparc boot program loads the NetBSD kernel at its linked virtual address. This feature requires a kernel built after 2003/02/21 (corresponding to kernel version 1.6Q). To load older kernels, the **−C** option must be used, which loads the kernel at physical address 0x4000. The size of a kernel loaded in this way is limited to approximately 3MB.

−q Boot the system in quiet mode.

−s Bring the system up in single-user mode.

−v Boot the system in verbose mode.

Any extra flags or arguments, or the ⟨*boot string*⟩ after the -- separator are passed to the boot PROM. Other flags are currently ignored.

The SPARC boot ROM comes in two flavours: an "old-style" ROM is used in sun4 machines, while a "new-style" ROM can be found on sun4c and sun4m models. The "new-style" SPARC boot ROM is a full-featured Forth system with emacs key bindings. It can be put in "old-style" user-interface compatibility mode (in which case it shows a simple '>' prompt), but this is essentially useless. However, by default on sun4c models, the ROM runs in old-mode; to enter new-mode type 'n'. The ROM then shows a Forth-style "ok" prompt. It is recommended to have the ROM always start in its native "new-style" mode. Utter the following incantation in new-mode to force the ROM to always start in new-mode.

```
ok setenv sunmon-compat? false
```

The ROM will normally load the kernel from "sd(0,0,0)vmunix". To change the default so that NetBSD will be loaded from somewhere else, type the following

```
ok setenv boot-from sd(0,0,0)netbsd
```

On newer SPARC machines, there are various aliases to access common devices. A typical list of usable boot devices (extracted from the output of the Open Boot PROM command **devalias**) is:

```
floppy          /obio/SUNW,fdtwo
net-aui         /iommu/sbus/ledma@f,400010:aui/le@f,c00000
net-tpe         /iommu/sbus/ledma@f,400010:tpe/le@f,c00000
```

```
net          /iommu/sbus/ledma@f,400010/le@f,c00000
disk         /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
cdrom        /iommu/sbus/espdma@f,400000/esp@f,800000/sd@6,0:d
tape         /iommu/sbus/espdma@f,400000/esp@f,800000/st@4,0
tape1        /iommu/sbus/espdma@f,400000/esp@f,800000/st@5,0
tape0        /iommu/sbus/espdma@f,400000/esp@f,800000/st@4,0
disk3        /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
disk2        /iommu/sbus/espdma@f,400000/esp@f,800000/sd@2,0
disk1        /iommu/sbus/espdma@f,400000/esp@f,800000/sd@1,0
disk0        /iommu/sbus/espdma@f,400000/esp@f,800000/sd@0,0
```

For new-style machines, if a device specification includes a partition letter (for example *cdrom* in above list), that partition is used by default, otherwise the first (a) partition is used. If booting from the net device, there is no partition involved.

At any time you can break back to the ROM by pressing the 'L1' and 'a' keys at the same time (if the console is a serial port the same is achieved by sending a 'break'). If you do this accidentally you can continue whatever was in progress by typing 'go'.

OPEN BOOT PROM ENVIRONMENT VARIABLES

This section only applies to new-style machines.

All Open Boot PROM environment variables can be printed with the **printenv** command and changed with the **setenv** command. The boot process relevant variables and their suggested value for booting NetBSD are:

```
auto-boot?          true
boot-file
boot-device         disk
diag-switch?        false
```

Of course you may select any other boot device, if you do not want to boot from the device aliased to *disk*, see the discussion on devices above.

OPEN BOOT PROM ABBREVIATED COMMAND SUMMARY

This section only applies to new-style machines.

The following Open Boot PROM commands are related to the boot process:

```
boot               boot the system from the default device
boot device filename arguments
                   boot the specified device, filename and arguments
probe-ide          list devices on the primary IDE controller
probe-ide-all      list devices on all known IDE controllers
probe-scsi         list devices on the primary SCSI controller
probe-scsi-all     list devices on all known SCSI controllers
reset              reset the system
```

For disk and tape devices, the boot device is specified as '/path/device@target,lun:partition'.

PROM MONITOR ABBREVIATED COMMAND SUMMARY

This section only applies to old-style machines.

The following PROM monitor commands are related to the boot process:

```
b       boot the system from the default device
b device filename arguments
```

```
                boot the specified device, filename and arguments
        b?      list boot device types
        k2      reset the system
```

For SCSI disk and tape devices, the boot device is specified as 'device(controller,unit,partition)', where 'unit' is the hexidecimal value of the SCSI id of the target multiplied by eight plus the lun, and 'partition' is the partition number, starting from 0.

FILES

`/netbsd`	system code
`/boot`	system bootstrap

SEE ALSO

`sparc64/boot`(8), `disklabel`(8), `fsck`(8), `halt`(8), `init`(8), `installboot`(8), `rc`(8), `shutdown`(8), `syslogd`(8)

BUGS

On sun4 machines, the NetBSD sparc boot loader can only boot from RAID partitions that start at the beginning of the disk.

On sun4 and early PROM version sun4c machines, the PROM can only boot from the first 1Gb of the disk.

On later PROM version sun4c and early PROM version sun4m machines, the PROM can only boot from the first 2Gb of the disk.

On later PROM version sun4m machines, the PROM can only boot from the first 4Gb of the disk.

NAME

boot, **ofwboot** — system bootstrapping procedures

SYNOPSIS

boot [**−adqsv**] [-- ⟨*boot string*⟩]

DESCRIPTION

Sun UltraSPARC systems support booting from locally attached storage media (e.g. hard disk, CD-ROM) , and booting over Ethernet networks using BOOTP.

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in fsck(8), and unless this fails, the system will resume multi-user operations.

Cold starts

The Sun Open Firmware performs a Power On Self Test (POST) , and then will boot an operating system according to configuration in Open Firmware environment variables.

Boot program options

−a Prompt for the root file system device, the system crash dump device, and the path to init(8).

−d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see gdb(1).

−q Boot the system in quiet mode.

−s Bring the system up in single-user mode.

−v Boot the system in verbose mode.

Any extra flags or arguments, or the ⟨*boot string*⟩ after the -- separator are passed to the boot PROM. Other flags are currently ignored.

At any time you can halt the running system and get back to the Open Firmware. If the console is the Sun framebuffer and keyboard, press the 'STOP' and 'A' keys at the same time on the keyboard. On older models of Sun keyboards, the 'STOP' key is labelled 'L1'.

If the console is a serial port the same is achieved by sending a 'BREAK'.

If you do this accidentally, you can continue whatever was in progress with the **go** command.

BOOT DEVICES

Since machines vary greatly in the way their devices are connected, there are aliases defined by the firmware. You can either use the fully qualified Open Firmware path of a device node, or the alias.

The secondary boot loader, **ofwboot**, takes **boot** commands virtually the same as Open Firmware. Thus, the following examples apply equally to **ofwboot** as well as Open Firmware.

A typical list of usable boot devices (extracted from the output of the Open Firmware command **devalias**) is:

```
net                       /sbus/SUNW,hme@e,8c00000
disk                      /sbus/SUNW,fas@e,8800000/sd@0,0
cdrom                     /sbus/SUNW,fas@e,8800000/sd@6,0:f
disk6                     /sbus/SUNW,fas@e,8800000/sd@6,0
disk5                     /sbus/SUNW,fas@e,8800000/sd@5,0
disk4                     /sbus/SUNW,fas@e,8800000/sd@4,0
```

```
disk3                    /sbus/SUNW,fas@e,8800000/sd@3,0
disk2                    /sbus/SUNW,fas@e,8800000/sd@2,0
disk1                    /sbus/SUNW,fas@e,8800000/sd@1,0
disk0                    /sbus/SUNW,fas@e,8800000/sd@0,0
```

If a device specification includes a partition letter (for example *cdrom* in above list), that partition is used by default, otherwise the first (a) partition is used. If booting from the net device, there is no partition involved.

The boot device is an optional first part of the boot string, if no device is specified the default device is used (see below).

FIRMWARE ENVIRONMENT VARIABLES

All Open Firmware environment variables can be printed with the **printenv** command and changed with the **setenv** command. The boot process relevant variables and their suggested value for booting NetBSD are:

boot-command	boot
auto-boot?	true
boot-file	
boot-device	disk
diag-switch?	false

Of course you may select any other boot device, if you do not want to boot from the device aliased to *disk*, see the discussion on devices above.

FILES

/netbsd	system code
/ofwboot	system bootstrap
/usr/mdec/ofwboot.net	alternate bootstrap when booting from the network, see diskless(8) for details.

EXAMPLES

Boot from CD-ROM:

```
boot cdrom
```

Note that some multi-architecture CDs are not able to use the default sparc64 partition for CD-ROMs (f), so they may require an explicit partition letter, for example

```
boot cdrom:c
```

When using external SCSI CD-ROM drives it is important to know two things: the Sun firmware expects the SCSI ID to be six, and the drive must support 512-byte block reads, in addition to the standard 2048-byte reads.

Use

```
boot net -sd
```

to boot single user from network and break into the kernel debugger as soon as possible.

Use

```
boot net tftp:netbsd -a
```

to boot a kernel named netbsd obtained via tftp and have it ask for root filesystem, swap partition and init location once it is up.

During installation from a different operating system

```
boot disk:b
```

is used to boot a "miniroot" filesystem from the swap partition.

SEE ALSO

`sparc/boot`(8), `disklabel`(8), `diskless`(8), `fsck`(8), `halt`(8), `init`(8), `installboot`(8), `rc`(8), `shutdown`(8), `syslogd`(8)

STANDARDS

Sun developed its firmware and promoted it to become IEEE Std 1275-1994 ("Open Firmware").

`http://playground.sun.com/1275/`

BUGS

NetBSD provides no way to boot UltraSPARC systems from floppy disks. This is unlikely to change, due to very low demand for this feature.

NAME

boot — system bootstrapping procedures

SYNOPSIS

b [*dev* [(*cntrl*, *unit*, *part*)]] [*file*] [**−adqsv**]

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in `fsck`(8), and unless this fails, the system will resume multi-user operations.

Selecting the device and kernel to boot

Normally, the **b** command alone is sufficient to boot the system, as the PROM chooses a default boot device *dev* if none is specified. The PROM chooses the first device present on the system from the following ordered list:

sd	SCSI disk
ie	Intel Ethernet
ec	3Com Ethernet

Unless specified, the controller number *cntrl*, unit number *unit*, and partition number *part* default to zero, which is almost always correct.

The controller number can be specified if there is more than one of the given device in the system. For example, use "ie(1,,)" to boot off of the second Intel Ethernet in the system.

The unit number specifies one of the many devices attached to a controller. The exact meaning and values vary depending on the device name. For example, "sd(,18,)" boots the disk at target 6 on the first SCSI controller, 18 being the target number 6, multiplied by 4, and given in hexadecimal.

The partition number specifies one of the many partitions on a device. The exact meaning and values vary depending on the device name. For example, "sd(,18,1)" boots the second partition on the disk at target 6 on the first SCSI controller.

The PROM only loads a first-stage boot program, currently either `/usr/mdec/bootxx` (for a disk boot), or `/usr/mdec/bootyy` (for a network boot). This first-stage boot program then loads the second-stage boot program from the same device, currently either `/usr/mdec/ufsboot` (for a disk boot), or `/usr/mdec/netboot` (for a network boot).

The second-stage boot program will then attempt to load the kernel named *file* (or `vmunix` if none is specified). The second-stage disk boot program `/usr/mdec/ufsboot` loads the kernel from the same device that it was loaded from, while the second-stage network boot program `/usr/mdec/netboot` will load the kernel from the NFS root as determined by the procedure described in `diskless`(8).

Boot program options

−a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

−d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

−q Boot the system in quiet mode.

−s Bring the system up in single-user mode.

−v Boot the system in verbose mode.

Other flags are currently ignored.

At any time you can break back to the ROM by pressing the 'L1' and 'a' keys at the same time (if the console is a serial port the same is achieved by sending a 'break'). If you do this accidentally you can continue whatever was in progress by typing 'c' followed by the return key.

FILES

`/netbsd`	system code
`/usr/mdec/bootxx`	first-level boot block for disks
`/usr/mdec/bootyy`	first-level boot block for NFS (diskless) boot
`/usr/mdec/netboot`	boot program for NFS (diskless) boot
`/usr/mdec/ufsboot`	second-level boot program for UFS disks
`/usr/sbin/installboot`	program to install bootxx on a disk

SEE ALSO

`disklabel`(8), `fsck`(8), `halt`(8), `init`(8), `rc`(8), `shutdown`(8), `syslogd`(8)

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed as described in `fsck`(8), and unless this fails, the system will resume multi-user operations.

Cold starts

A disk-boot program (`/usr/mdec/ufsboot`) will attempt to load `netbsd` from partition A of the boot device, which must currently be an "sd" disk. Alternatively, network boot program (`/usr/mdec/netboot`) will load `netbsd` from the NFS root as determined by the procedure described in `diskless`(8).

Boot program options

-a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

-d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

-q Boot the system in quiet mode.

-s Bring the system up in single-user mode.

-v Boot the system in verbose mode.

Any extra flags or arguments, or the ⟨*boot string*⟩ after the -- separator are passed to the boot PROM. Other flags are currently ignored.

At any time you can break back to the ROM by pressing the 'L1' and 'a' keys at the same time (if the console is a serial port the same is achieved by sending a 'break'). If you do this accidentally you can continue whatever was in progress by typing 'c' followed by the return key.

FILES

`/netbsd`	system code
`/usr/mdec/bootxx`	first-level boot block for disks
`/usr/mdec/netboot`	boot program for NFS (diskless) boot
`/usr/mdec/ufsboot`	second-level boot program for UFS disks
`/usr/mdec/installboot`	program to install bootxx on a disk

SEE ALSO

`disklabel`(8), `fsck`(8), `halt`(8), `init`(8), `rc`(8), `shutdown`(8), `syslogd`(8)

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery Normally, the system will reboot itself at power-up or after crashes. Provided the auto-restart is enabled on the machine front panel, an automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts These are processor-type dependent. On an 11/780, there are two floppy files for each disk controller, both of which cause boots from unit 0 of the root file system of a controller located on mba0 or uba0. One gives a single user shell, while the other invokes the multi-user automatic reboot. Thus these files are HPS and HPM for the single and multi-user boot from MASSBUS RP06/RM03/RM05 disks, UPS and UPM for UNIBUS storage module controller and disks such as the EMULEX SC-21 and AMPEX 9300 pair, RAS and RAM to boot from MSCP controllers and disks such as the RA81, or HKS and HKM for RK07 disks. There is also a script for booting from the default device, which is normally a copy of one of the standard multi-user boot scripts, but which may be modified to perform other actions or to boot from a different unit. The situation on the 8600 is similar, with scripts loaded from the console RL02.

Giving the command

```
>>>BOOT HPM
```

would boot the system from (e.g.) an RP06 and run the automatic consistency check as described in `fsck`(8). (Note that it may be necessary to type control-P and halt the processor to gain the attention of the LSI-11 before getting the >>> prompt.) The command

```
>>>BOOT ANY
```

invokes a version of the boot program in a way which allows you to specify any system as the system to be booted. It reads from the console a device specification (see below) followed immediately by a pathname.

The scripts may be modified for local configuration if necessary. The flags are placed in register 11 (as defined in ⟨`sys/reboot.h`⟩). The boot device is specified in register 10. The encoding of this register is also defined in ⟨`sys/reboot.h`⟩. The current encoding has a historical basis, and is shown in the following table:

bits	usage
0-7	boot device type (the device major number)
8-15	disk partition
16-19	drive unit
20-23	controller number
24-27	adaptor number (UNIBUS or MASSBUS as appropriate)

The adaptor number corresponds to the normal configuration on the 11/750, and to the order in which adaptors are found on the 11/780 and 8600 (generally the same as the numbers used by UNIX).

On an 11/750, the reset button will boot from the device selected by the front panel boot device switch. In systems with RK07's, position B normally selects the RK07 for boot. This will boot multi-user. To boot from RK07 with boot flags you may specify

```
>>>B/-n DMA0
```

where, giving a *n* of 1 causes the boot program to ask for the name of the system to be bootstrapped, giving a *n* of 2 causes the boot program to come up single user, and a *n* of 3 causes both of these actions to occur. The "DM" specifies RK07, the "A" represents the adaptor number (UNIBUS or MASSBUS), and the "0" is the drive unit number. Other disk types which may be used are DB (MASSBUS), DD (TU58), and DU (UDA-50/RA disk). A non-zero disk partition can be used by adding (partition times 1000 hex) to *n*.

The boot procedure on the Micro VAX II is similar. A switch on the back panel sets the power-up action to autoboot or to halt. When halted, the processor may be booted using the same syntax as on the 11/750.

The 11/750 boot procedure uses the boot ROMs to load block 0 off of the specified device. The /usr/mdec directory contains a number of bootstrap programs for the various disks which should be placed in a new pack by `disklabel`(8). Similarly, the Micro VAX II boot procedure loads a boot parameter block from block 0 of the disk. The **`rdboot`** "bootstrap" contains the correct parameters for an MSCP disk such as the RD53.

On any processor, the *boot* program finds the corresponding file on the given device (`netbsd` by default), loads that file into memory location zero, and starts the program at the entry address specified in the program header (after clearing off the high bit of the specified entry address).

The file specifications used with "BOOT ANY" or "B/3" are of the form:

```
device(adaptor,controller,unit,minor)
```

where *`device`* is the type of the device to be searched, *`adaptor`* is the UNIBUS or MASSBUS number of the adaptor to which the device is attached, *`controller`* is the unit number of the controller or MASSBUS tape formatter on that adaptor, *`unit`* is the unit number of the disk or transport slave unit of the tape, and *`minor`* is the disk partition or tape file number. Leading adaptor or controller numbers default to 0. Normal line editing characters can be used when typing the file specification. The following list of supported devices may vary from installation to installation:

hp	MASSBUS disk drive
up	UNIBUS storage module drive
ht	TE16,TU45,TU77 on MASSBUS
kra	storage module on a KDB50
mt	TU78 on MASSBUS
hk	RK07 on UNIBUS
ra	storage module on a MSCP-compatible UNIBUS controller
rb	storage module on a 730 IDC
rl	RL02 on UNIBUS
tm	TM11 emulation tape drives on UNIBUS
tms	TMSCP-compatible tape
ts	TS11 on UNIBUS
ut	UNIBUS TU45 emulator

For example, to boot from a file system which starts at cylinder 0 of unit 0 of a MASSBUS disk, type `hp(0,0)netbsd` to the boot prompt; `hp(2,0,1,0)netbsd` would specify drive 1 on MASSBUS adaptor 2; `up(0,0)netbsd` would specify a UNIBUS drive, `hk(0,0)netbsd` would specify an RK07 disk drive, `ra(1,0,0,0)netbsd` would specify a UDA50 disk drive on a second UNIBUS, and `rb(0,0)netbsd` would specify a disk on a 730 IDC. For tapes, the minor device number gives a file offset; `mt(1,2,3,4)` would specify the fifth file on slave 3 of the formatter at `drive` 2 on mba 1.

On an 11/750 with patchable control store, microcode patches will be installed by *boot* if the file `psc750.bin` exists in the root of the filesystem from which the system is booted.

In an emergency, the bootstrap methods described in the paper *Installing and Operating 4.3bsd* can be used to boot from a distribution tape.

FILES

`/netbsd`	system code
`/boot`	system bootstrap

`/usr/mdec/xxboot`	sector-0 boot block for 750, xx is disk type
`/usr/mdec/bootxx`	second-stage boot for 750, xx is disk type
`/pcs750.bin`	microcode patch file on 750

SEE ALSO

`arff`(8), `halt`(8), `reboot`(8), `shutdown`(8)

HISTORY

The **`boot`** command appeared in 4.0BSD.

NAME

boot — system bootstrapping procedures

DESCRIPTION

Power fail and crash recovery

Normally, the system will reboot itself at power-up or after crashes. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

Cold starts

The X68000/X68030 system boots from the device which is determined by the configuration of battery-backuped SRAM. By default, the boot ROM attempts to boot from floppy disk drives (from 0 to 3) first, and then attempts to boot from hard disk (SASI or SCSI). On the NetBSD/x68k, booting from SCSI disks (sd??) and 2HD floppy disks (fd?a, fd?c) is currently supported.

Bootstrapping from a floppy

When the floppy disk is selected as the boot device, the initial program loader of the IOCS (firmware) reads the `fdboot_ufs` program at the top of the disk, and then the fdboot_ufs program loads the `/boot` program from the FFS or LFS file system. Normally, the `/boot` program then loads the NetBSD kernel `/netbsd` from the same floppy. In addition, the `/boot` program has abilities to uncompress gzip'ed kernels, to read the kernel from other disks of other file systems etc (see below).

For floppy disks, `fdboot_ustar` is also provided to read large kernels which do not fit one a single floppy.

Bootstrapping from a SCSI hard disk

When a SCSI hard disk is selected as the boot device, the initial program loader on the SCSI host adapter's ROM reads the operating system-independent IPL menu program at the top of the disk. The IPL menu program recognizes the partition table, and selects the partition to read the operating system kernel. During this phase, when the HELP key on the keyboard is pressed, the IPL menu program displays the partition menu of that disk to prompt the user to select the boot partition (although the NetBSD implementation of the IPL menu, `/usr/mdec/mboot`, does not have this functionality).

Next, the IPL menu reads the OS-dependent boot program from the top of the selected partition. For NetBSD FFS/LFS file systems `sdboot_ufs` is used. The `sdboot_ufs` program then loads the `/boot` program from that partition.

Normal Operation

Once running, a banner similar to the following will appear:

```
NetBSD Multi-boot, Revision 1.1
(user@buildhost, builddate)
Press return to boot now, any other key for boot menu
booting sd0a:netbsd - starting in 5
```

After a countdown, the system image listed will be loaded. (In the example above, it will be "`sd0a:netbsd`" which is the file **netbsd** on partition "a" of the NetBSD SCSI hard disk of ID 0. Pressing a key within the time limit will enter interactive mode.

Interactive mode

In interactive mode, the boot loader will present a prompt, allowing input of these commands:

boot [*device*:][*filename*] [**−adqsv**]

The default *device* will be set to the disk that the boot loader was loaded from. To boot from an alternate disk, the full name of the device should be given at the prompt. *device* is of the form *xd*[*N*[*x*]] where *xd* is the device from which to boot, *N* is the unit number, and *x* is the partition letter.

The following list of supported devices may vary from installation to installation:

sd	SCSI disks on a controller recognized by the IOCS. The unit number is the SCSI ID.
fd	Floppy drives as numbered by the IOCS.

The default *filename* is `netbsd`; if the boot loader fails to successfully open that image, it then tries `netbsd.gz` (expected to be a kernel image compressed by `gzip`(1)). Alternate system images can be loaded by just specifying the name of the image.

Options are:

-a Prompt for the root file system device, the system crash dump device, and the path to `init`(8).

-d Bring the system up in debug mode. Here it waits for a kernel debugger connect; see `ddb`(4).

-q Boot the system in quiet mode.

-s Bring the system up in single-user mode.

-v Boot the system in verbose mode.

help Print an overview about commands and arguments.

ls [path]
Print a directory listing of `path`, containing inode number, filename and file type. `path` can contain a device specification.

halt Reboot the system.

Model-specific notes

Note for X68030+MC68030 systems: Nothing special to be attended to; you can boot NetBSD just like as other operating systems such as Human68k and OS-9.

Note for X68030/040turbo(68040 accelerator by BEEPs) systems: NetBSD can boot under 040 mode. It can also boot under 030 mode if you have MC68030 on the board.

Note for X68000/Xellent30(68030 accelerator by TSR)+MC68030 systems: In order to boot NetBSD, you must choose 030 mode by using `CH30.SYS`, which must reside in the battery-backuped SRAM.

Note for X68000/Jupiter-X(68040/060 accelerator by FTZ-net) systems: The system must be in 040/060 processor mode.

FILES

`/netbsd`	system code
`/netbsd.gz`	gzip-compressed system code
`/usr/mdec/xxboot_ufs`	boot block (read by installboot), xx is disktype
`/usr/mdec/boot`	source of /boot (can be just copied to the root directory)
`/boot`	main part of the boot program

SEE ALSO

`reboot`(2), `disklabel`(8), `halt`(8), `reboot`(8), `shutdown`(8)

NAME

boot26 — Bootloader for NetBSD/acorn26

SYNOPSIS

∗**boot26** [**−acdqsv**] [*file*]

DESCRIPTION

boot26 is a program that runs under RISC OS and launches the NetBSD/acorn26 kernel. It needs to be installed in a RISC OS filesystem and given file type FFA (Module). The kernel it is to load also needs to be stored in a RISC OS filesystem.

It takes the following options, which set flags in the *boothowto* variable in the booted kernel:

−a (RB_ASKNAME) Cause the kernel to prompt the user for the name of the device containing the root filesystem. This also causes **boot26** to prompt for the name of the kernel to be loaded.

−s (RB_SINGLE) Cause the kernel to ask **init** to boot into single-user mode.

−d (RB_KDB) Cause the kernel to enter the kernel debugger as soon as possible.

−c (RB_USERCONF) Enter the in-kernel device configuration manager before attaching any devices.

−q (RB_QUIET) Cause the kernel to emit fewer messages than normal while starting up.

−v (RB_VERBOSE) Cause the kernel to emit more messages than normal while starting up.

boot26 attempts to load the kernel from the RISC OS file specified as *file*, or from netbsd if *file* is not specified. The file must be an ELF image, and may have been compressed using gzip(1).

Use as a module

boot26 is implemented as a RISC OS relocatable module. It can be loaded into memory by running ∗RMLoad boot26. After this, NetBSD can be booted by running ∗boot26 as usual, but the command will be handled by the module.

It should also be possible to arrange for **boot26** to be loaded from ROM (e.g. from the ROM on an expansion card), in which case NetBSD could be made to boot automatically by making **boot26** the configured language using ∗Configure Language.

Screen display

When it starts up, **boot26** displays the current memory map. Each character in the map represents one page of (physical) RAM. The ticks along the top are to stop you getting lost. The characters in the map indicate what the memory's being used for (actually where it's logically mapped):

```
0 -> zero-page
+ -> boot26 workspace
* -> Free space (boot26 wants to put the kernel here)
d -> RAM disc
s -> System sprite area
m -> RMA
h -> System heap/stack
f -> Font cache
S -> Screen memory
```

On a machine with 32k pages (which is all NetBSD/acorn26 supports), the left half of the first line is potential screen memory, and hence not used by **boot26**. The next page is usually zero page under RISC OS, and is used for zero page under NetBSD as well. The next is usually the system heap under RISC OS, and is used for process 0's kernel stack under NetBSD. The next is used for the message buffer under NetBSD. Pages from there on are used to load the kernel, and must be free if **boot26** is to do so successfully. Future boot-loaders should load the kernel into whatever pages are free, then kick out RISC OS and shuffle them into the right shape. This is left as an exercise for the enthusiastic reader.

FILES

`/usr/mdec/boot26,ffa`	The location of **boot26** in the NetBSD filesystem.

SEE ALSO

`gzip`(1), `reboot`(2), `ddb`(4), `userconf`(4), `init`(8)

HISTORY

boot26 was introduced in NetBSD 1.6 as a replacement for the original NetBSD/arm26 bootloader, which was written in BBC BASIC.

BUGS

boot26 cannot load kernels from a NetBSD filesystem.

NAME

boot32 — Bootloader for NetBSD/acorn32

SYNOPSIS

*****boot32** [**-acdqsv**] [*root=rootdir*] [*file*]

DESCRIPTION

boot32 is a program that runs under RISC OS and launches the NetBSD/acorn32 kernel. It needs to be installed in a RISC OS filesystem and given file type FFA (Module). The kernel it is to load also needs to be stored in a RISC OS filesystem.

It takes the following standard NetBSD options, which set flags in the *boothowto* variable in the booted kernel (not all flags may be effective):

-a (RB_ASKNAME) Cause the kernel to prompt the user for the name of the device containing the root filesystem. This also causes **boot32** to prompt for the name of the kernel to be loaded.

-s (RB_SINGLE) Cause the kernel to ask **init** to boot into single-user mode.

-d (RB_KDB) Cause the kernel to enter the kernel debugger as soon as possible.

-c (RB_USERCONF) Enter the in-kernel device configuration manager before attaching any devices.

-q (RB_QUIET) Cause the kernel to emit fewer messages than normal while starting up.

-v (RB_VERBOSE) Cause the kernel to emit more messages than normal while starting up.

boot32 attempts to load the kernel from the RISC OS file specified as *file*, or from netbsd if *file* is not specified. The file must be an ELF image, and may have been compressed using gzip(1).

Use as a module

boot32 is implemented as a RISC OS relocatable module. It can be loaded into memory by running *RMLoad boot32. After this, NetBSD can be booted by running *boot32 as usual, but the command will be handled by the module.

It should also be possible to arrange for **boot32** to be loaded from ROM (e.g., from the ROM on an expansion card), in which case NetBSD could be made to boot automatically by making **boot32** the configured language using *Configure Language.

Screen display

When it starts up, **boot32** displays the number of 4 kilobyte memory pages it has been delegated by RISC-OS and gives a summary about the memory map as reported by RISC-OS followed by a table of physical memory ranges available to the bootloader. All this information is mainly for bughunting booting problems.

It then checks its internal structures and kicks out RISC-OS, relocates all memory pages loaded in to their final destinations and kickstarts **boot32**.

FILES

/usr/mdec/boot32,ffa The location of **boot32** in the NetBSD filesystem.

SEE ALSO

gzip(1), reboot(2), ddb(4), userconf(4), init(8)

HISTORY

boot32 was introduced in NetBSD 1.6 as a replacement for the original NetBSD/arm32 bootloader, which was written in BBC BASIC.

BUGS

boot32 cannot load kernels from a NetBSD filesystem.

NAME

`boot_console` — selection of a console device in the i386 bootloader

DESCRIPTION

The NetBSD i386 bootloader selects a console device for its user interaction and passes information about it to the NetBSD kernel. When booting from the system BIOS, the console device and properties are saved in the primary bootstrap by `installboot`(8). For other boot procedures (such as `dosboot`(8)) the selection process is controlled by bootloader compile-time options and system setup at the bootloader startup time. The selection may be changed on-the-fly from within the bootloader.

Serial Console Options

The compile-time options (to be set in the booter's "Makefile") are:

SUPPORT_SERIAL=*`policy`*
enables support for serial input/output. By default this option is not set and the standard PC keyboard and display are always used as the console device. See **Console Selection Policy** below for valid values of *`policy`*.

DIRECT_SERIAL
causes direct hardware access to be used for serial input / output. With this option, software handshake (XON/XOFF) is used for flow control. Without this option, BIOS functions are employed for serial port handling, which require hardware handshake lines to be completely wired.

CONSPEED=*`integer`*
sets the baud-rate for the serial console. This option has only an effect when used in combination with the "`DIRECT_SERIAL`" option above, otherwise, the default setting of 9600 baud is used. The value of *`integer`* must be something that makes sense as a serial port baud rate.

COMCONS_KEYPRESS
Require a character input within seven (7) seconds from serial console device to be selected.

Console Selection Policy

The actual policy for the console selection is determined by the value of "`SUPPORT_SERIAL`" The following options are available:

CONSDEV_PC
Force use of the standard PC keyboard and display as the console.

CONSDEV_COM0 `. . .` *CONSDEV_COM3*
Use the serial port with the corresponding BIOS number as the console. No attempt is made to verify connectivity on the selected port. If the port is not known to the BIOS, it falls back to "`CONSDEV_PC`". (Note: This feature can be deliberately used for console selection if the serial ports have been disabled in the BIOS.)

CONSDEV_COM0KBD `. . .` *CONSDEV_COM3KBD*
If the port is known to the BIOS, and output of a character to the port succeeds (and if "`DIRECT_SERIAL`" is defined the RS-232 "modem ready" status is on after the character is output), the port is used as console. If the port is not known to the BIOS, or the test output fails, it falls back to "`CONSDEV_PC`".

CONSDEV_AUTO
Auto-select the console. All serial ports known to the BIOS are probed in sequence. If output of a character to the port succeeds (and if "`DIRECT_SERIAL`" is defined the RS-232 "modem ready" status is on after the character is output), the port is used as console. If no serial port passes the check, "`CONSDEV_PC`" is used. The progress of the selection process is shown at the PC display as digits corresponding to the serial port number currently probed.

FILES

`/sys/arch/i386/stand/{bios,dos,net}boot/Makefile` compile time options for the boot programs.

SEE ALSO

`console`(4), `boot`(8), `installboot`(8)

BUGS

The value of `SERIAL_POLICY` should be settable through a boot configuration option. However traditionally there was no non-volatile storage available on the PC platform. This requires console auto-selection methods which can be inconvenient and/or unstable in some situations. The selection policy should be adapted to the local hardware configuration, which might require code changes. (Some BIOS versions, particularly those used on large servers and in embedded and single-board industrial computers, have integrated support for serial consoles. The boot loader should query for these settings if possible.)

The serial communication parameters (byte-size, parity, stop-bits) are not settable (either at compile time or run time). The default parameters are "8 N 1".

The baud rate is not settable when using BIOS I/O. It should be settable at compile time with "`CONSPEED`" just as it is when using "`DIRECT_SERIAL`". The default speed is 9600 baud (the maximum for BIOS I/O).

NAME

bootpd, bootpgw − Internet Boot Protocol server/gateway

SYNOPSIS

bootpd [**−i −s −t** timeout **−d** level **−c** chdir−path] [*bootptab* [*dumpfile*]]
bootpgw [**−i −s −t** timeout **−d** level] server

DESCRIPTION

Bootpd implements an Internet Bootstrap Protocol (BOOTP) server as defined in RFC951, RFC1532, and RFC1533. *Bootpgw* implements a simple BOOTP gateway which can be used to forward requests and responses between clients on one subnet and a BOOTP server (i.e. *bootpd*) on another subnet. While either *bootpd* or *bootpgw* will forward BOOTREPLY packets, only *bootpgw* will forward BOOTREQUEST packets.

One host on each network segment is normally configured to run either *bootpd* or *bootpgw* from *inetd* by including one of the following lines in the file */etc/inetd.conf*:

```
bootps dgram udp wait root /usr/sbin/bootpd bootpd bootptab
bootps dgram udp wait root /usr/sbin/bootpgw bootpgw server
```

This mode of operation is referred to as "inetd mode" and causes *bootpd* (or *bootpgw*) to be started only when a boot request arrives. If it does not receive another packet within fifteen minutes of the last one it received, it will exit to conserve system resources. The **−t** option controls this timeout (see OPTIONS).

It is also possible to run *bootpd* (or *bootpgw*) in "standalone mode" (without *inetd*) by simply invoking it from a shell like any other regular command. Standalone mode is particularly useful when *bootpd* is used with a large configuration database, where the start up delay might otherwise prevent timely response to client requests. (Automatic start up in standalone mode can be done by invoking *bootpd* from within */etc/rc.local*, for example.) Standalone mode is less useful for *bootpgw* which has very little start up delay because it does not read a configuration file.

Either program automatically detects whether it was invoked from inetd or from a shell and automatically selects the appropriate mode. The **−s** or **−i** option may be used to force standalone or inetd mode respectively (see OPTIONS).

OPTIONS

−t *timeout*
: Specifies the *timeout* value (in minutes) that a *bootpd* or *bootpgw* process will wait for a BOOTP packet before exiting. If no packets are received for *timeout* minutes, then the program will exit. A timeout value of zero means "run forever". In standalone mode, this option is forced to zero.

−d *debug−level*
: Sets the *debug−level* variable that controls the amount of debugging messages generated. For example, -d4 or -d 4 will set the debugging level to 4. For compatibility with older versions of *bootpd*, omitting the numeric parameter (i.e. just -d) will simply increment the debug level by one.

−c *chdir−path*
: Sets the current directory used by *bootpd* while checking the existence and size of client boot files. This is useful when client boot files are specified as relative pathnames, and *bootpd* needs to use the same current directory as the TFTP server (typically /tftpboot). This option is not recognized by *bootpgw*.

−i
: Force inetd mode. This option is obsolete, but remains for compatibility with older versions of *bootpd*.

−s
: Force standalone mode. This option is obsolete, but remains for compatibility with older versions of *bootpd*.

bootptab
: Specifies the name of the configuration file from which *bootpd* loads its database of known clients and client options (*bootpd* only).

dumpfile
: Specifies the name of the file that *bootpd* will dump its internal database into when it receives a SIGUSR1 signal (*bootpd* only). This option is only recognized if *bootpd* was compiled with the -DDEBUG flag.

server
: Specifies the name of a BOOTP server to which *bootpgw* will forward all BOOTREQUEST packets it receives (*bootpgw* only).

OPERATION

Both *bootpd* and *bootpgw* operate similarly in that both listen for any packets sent to the *bootps* port, and both simply forward any BOOTREPLY packets. They differ in their handling of BOOTREQUEST packets.

When *bootpgw* is started, it determines the address of a BOOTP server whose name is provided as a command line parameter. When *bootpgw* receives a BOOTREQUEST packet, it sets the "gateway address" and "hop count" fields in the packet and forwards the packet to the BOOTP server at the address determined earlier. Requests are forwarded only if they indicate that the client has been waiting for at least three seconds.

When *bootpd* is started it reads a configuration file, (normally */etc/bootptab*) that initializes the internal database of known clients and client options. This internal database is reloaded from the configuration file when *bootpd* receives a hangup signal (SIGHUP) or when it discovers that the configuration file has changed.

When *bootpd* receives a BOOTREQUEST packet, it looks for a database entry matching the client request. If the client is known, *bootpd* composes a BOOTREPLY packet using the database entry found above, and sends the reply to the client (possibly using a gateway). If the client is unknown, the request is discarded (with a notice if debug > 0).

If *bootpd* is compiled with the -DDEBUG option, receipt of a SIGUSR1 signal causes it to dump its internal database to the file */etc/bootpd.dump* or the dumpfile specified as a command line parameter.

During initialization, both programs determine the UDP port numbers to be used by calling *getservbyname* (which normally uses */etc/services).* Two service names (and port numbers) are used:

bootps − BOOTP Server listening port
bootpc − BOOTP Client destination port

If the port numbers cannot be determined using *getservbyname* then the values default to bootps=67 and bootpc=68.

FILES

/etc/bootptab	Database file read by *bootpd*.
/etc/bootpd.dump	Debugging dump file created by *bootpd*.
/etc/services	Internet service numbers.
/tftpboot	Current directory typically used by the TFTP server and *bootpd*.

BUGS

Individual host entries must not exceed 1024 characters.

CREDITS

This distribution is currently maintained by Walter L. Wimer <walt+@cmu.edu>.

The original BOOTP server was created by Bill Croft at Stanford University in January 1986.

The current version of *bootpd* is primarily the work of David Kovar, Drew D. Perkins, and Walter L. Wimer, at Carnegie Mellon University.

Enhancements and bug−fixes have been contributed by:
(in alphabetical order)
Danny Backx <db@sunbim.be>

John Brezak <brezak@ch.hp.com>
Frank da Cruz <fdc@cc.columbia.edu>
David R. Linn <drl@vuse.vanderbilt.edu>
Jim McKim <mckim@lerc.nasa.gov>
Gordon W. Ross <gwr@mc.com>
Jason Zions <jazz@hal.com>

SEE ALSO

bootptab(5), inetd(8), tftpd(8)

DARPA Internet Request For Comments:

RFC951	Bootstrap Protocol
RFC1532	Clarifications and Extensions for the Bootstrap Protocol
RFC1533	DHCP Options and BOOTP Vendor Extensions

NAME

bootpef − BOOTP Extension File compiler

SYNOPSIS

bootpef [-*c chdir*] [-*d debug-level*] [-*f config-file*] [*client-name* [...]]

DESCRIPTION

bootpef builds the *Extension Path* files described by RFC 1497 (tag 18). If any *client-name* arguments are specified, then *bootpef* compiles the extension files for only those clients.

OPTIONS

−c *chdir−path*
: Sets the current directory used by *bootpef* while creating extension files. This is useful when the extension file names are specified as relative pathnames, and *bootpef* needs to use the same current directory as the TFTP server (typically /tftpboot).

−d *debug−level*
: Sets the *debug−level* variable that controls the amount of debugging messages generated. For example, -d4 or -d 4 will set the debugging level to 4.

−f *config−file*
: Set the name of the config file that specifies the option data to be sent to each client.

SEE ALSO

bootpd(8), tftpd(8)

REFERENCES

RFC951
: BOOTSTRAP PROTOCOL (BOOTP)

RFC1497
: BOOTP Vendor Information Extensions

NAME

bootpref — set NVRAM boot preference

SYNOPSIS

bootpref [-v] [-b os] [-d delay] [-l lang] [-k kbd] [-s id] [-f fmt] [-1] [-2] [-e sep] [-c colours] [-n] [-p] [-t] [-v] [-4] [-8] [-o] [-O] [-x] [-X] [-i] [-I]

DESCRIPTION

bootpref views and sets the NVRAM boot preferences.

The program options are:

-V	verbose output (when setting preferences)
-b *netbsd*	set the boot OS to NetBSD
-b *tos*	set the boot OS to TOS
-b *linux*	set the boot OS to Linux
-b *systemv*	set the boot OS to System V
-b *none*	set the boot OS to none
-d *delay*	set the boot delay to *delay* seconds, where *delay* is a value between 0 and 255
-l *english*	set the language to English
-l *german*	set the language to German
-l *french*	set the language to French
-l *spanish*	set the language to Spanish
-l *italian*	set the language to Italian
-k *american*	set the keyboard layout to American
-k *german*	set the keyboard layout to German
-k *french*	set the keyboard layout to French
-k *british*	set the keyboard layout to British
-k *spanish*	set the keyboard layout to Spanish
-k *italian*	set the keyboard layout to Italian

−k *sw f*

−k *swiss french*
set the keyboard layout to Swiss (French)

−k *sw g*

−k *swiss german*
set the keyboard layout to Swiss (German)

−s *id* set the SCSI id to *id*, where *id* is a value between 0 and 7

−f *mmddyy*

−f *ddmmyy*

−f *yymmdd*

−f *yyddmm* set the date format

−1 set the date format to 12 hour clock

−2 set the date format to 24 hour clock

−e *sep* set the date format separator to *sep*

−c *colours*
set the number of *colours* - 2, 4, 16, 256 or 65535

−n set the video mode to *NTSC*

−p set the video mode to *PAL*

−t set the video mode to *TV*

−v set the video mode to *VGA*

−4 set the video mode to *40 columns*

−8 set the video mode to *80 columns*

−o set the video mode to *overscan*

−O set the video mode to *no overscan*

−x set the video mode to *ST compatibility*

−X set the video mode to *no ST compatibility*

−i set the video mode to *interlace* (TV), *double line* (VGA)

−I set the video mode to *no interlace/double line*

All strings can be specified by their shortest abbreviation

If no parameters are specified, **bootpref** shows the current boot preferences.

SEE ALSO

`installboot`(8)

HISTORY

The **`bootpref`** command first appeared in NetBSD 1.4.

AUTHORS

Julian Coleman

BUGS

Setting the boot OS to *`none`* will cause the machine not to boot from the hard disk.

The majority of the parameters are not used under NetBSD.

NAME

bootptest − send BOOTP queries and print responses

SYNOPSIS

bootptest [**−f** *bootfile*] [**−h**] [**−m** *magic_number*] *server−name* [*template-file*]

DESCRIPTION

bootptest sends BOOTP requests to the host specified as *server−name* at one−second intervals until either a response is received, or until ten requests have gone unanswered. After a response is received, **bootptest** will wait one more second listening for additional responses.

OPTIONS

−f *bootfile*
Fill in the boot file field of the request with *bootfile*.

−h Use the hardware (Ethernet) address to identify the client. By default, the IP address is copied into the request indicating that this client already knows its IP address.

−m *magic_number*
Initialize the first word of the vendor options field with *magic_number*.

A *template-file* may be specified, in which case **bootptest** uses the (binary) contents of this file to initialize the *options* area of the request packet.

CREDITS

The bootptest program is a combination of original and derived works. The main program module (bootptest.c) is original work by Gordon W. Ross <gwr@mc.com>. The packet printing module (print-bootp.c) is a slightly modified version of a file from the BSD tcpdump program.

This program includes software developed by the University of California, Lawrence Berkeley Laboratory and its contributors. (See the copyright notice in print-bootp.c)

SEE ALSO

bootpd(8)

REFERENCES

RFC951
BOOTSTRAP PROTOCOL (BOOTP)

RFC1048
BOOTP Vendor Information Extensions

NAME

bounce − Postfix delivery status reports

SYNOPSIS

bounce [generic Postfix daemon options]

DESCRIPTION

The **bounce**(8) daemon maintains per-message log files with delivery status information. Each log file is named after the queue file that it corresponds to, and is kept in a queue subdirectory named after the service name in the **master.cf** file (either **bounce**, **defer** or **trace**). This program expects to be run from the **master**(8) process manager.

The **bounce**(8) daemon processes two types of service requests:

- Append a recipient (non-)delivery status record to a per-message log file.
- Enqueue a delivery status notification message, with a copy of a per-message log file and of the corresponding message. When the delivery status notification message is enqueued successfully, the per-message log file is deleted.

The software does a best notification effort. A non-delivery notification is sent even when the log file or the original message cannot be read.

Optionally, a bounce (defer, trace) client can request that the per-message log file be deleted when the requested operation fails. This is used by clients that cannot retry transactions by themselves, and that depend on retry logic in their own client.

STANDARDS

RFC 822 (ARPA Internet Text Messages)
RFC 2045 (Format of Internet Message Bodies)
RFC 2822 (ARPA Internet Text Messages)
RFC 3462 (Delivery Status Notifications)
RFC 3464 (Delivery Status Notifications)
RFC 3834 (Auto-Submitted: message header)

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically, as **bounce**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

2bounce_notice_recipient (postmaster)
The recipient of undeliverable mail that cannot be returned to the sender.

backwards_bounce_logfile_compatibility (yes)
Produce additional **bounce**(8) logfile records that can be read by Postfix versions before 2.0.

bounce_notice_recipient (postmaster)
The recipient of postmaster notifications with the message headers of mail that Postfix did not deliver and of SMTP conversation transcripts of mail that Postfix did not receive.

bounce_size_limit (50000)
The maximal amount of original message text that is sent in a non-delivery notification.

bounce_template_file (empty)
Pathname of a configuration file with bounce message templates.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_notice_recipient (postmaster)
The recipient of postmaster notifications with the message headers of mail that cannot be delivered within $delay_warning_time time units.

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.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

internal_mail_filter_classes (empty)
What categories of Postfix-generated mail are subject to before-queue content inspection by non_smtpd_milters, header_checks and body_checks.

mail_name (Postfix)
The mail system name that is displayed in Received: headers, in the SMTP greeting banner, and in bounced mail.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

notify_classes (resource, software)
The list of error classes that are reported to the postmaster.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

FILES

/var/spool/postfix/bounce/* non-delivery records
/var/spool/postfix/defer/* non-delivery records
/var/spool/postfix/trace/* delivery status records

SEE ALSO

bounce(5), bounce message template format
qmgr(8), queue manager
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager

syslogd(8), system logging

LICENSE

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

httpd — hyper text transfer protocol version 1.1 daemon

SYNOPSIS

httpd [**−HVXbenrus**] [**−C** *suffix cgihandler*] [**−I** *port*]
[**−M** *suffix type encoding encoding11*] [**−S** *server_software*]
[**−c** *cgibin*] [**−i** *address*] [**−p** *pubdir*] [**−v** *virtualroot*] [**−x** *index*]
[**−Z** *cert privkey*] *slashdir* [*myname*]

DESCRIPTION

The **httpd** program reads a *HTTP* request from the standard input, and sends a reply to the standard output. Besides ˜user translation and virtual hosting support (see below), all file requests are from *slashdir* directory. The server uses *myname* as its name, which defaults to the local hostname, obtained from gethostname(3) (but see the **−v** option for virtual hosting.) **httpd** is designed to be small, simple and relatively featureless, hopefully increasing its security.

OPTIONS

The following options are available:

−b This option enables daemon mode, where **httpd** detaches from the current terminal, running in the background and servicing HTTP requests.

−C *suffix cgihandler*
This option adds a new CGI handler program for a particular file type. The *suffix* should be any normal file suffix, and the *cgihandler* should be a full path to an interpreter. This option is the only way to enable CGI programs that exist outside of the cgibin directory to be executed. Multiple **−C** options may be passed.

−c *cgibin*
This option enables the CGI/1.1 interface. The *cgibin* directory is expected to contain the CGI programs to be used. **httpd** looks for URL's in the form of */cgi-bin/<scriptname>* where *<scriptname>* is a valid CGI program in the *cgibin* directory. In other words, all CGI URL's must begin with */cgi-bin/*. Note that the CGI/1.1 interface is not available with *˜user* translation.

−e This option causes **httpd** to not clear the environment when used with either the **−t** or **−U** options.

−H This option causes directory index mode to hide files and directories that start with a period, except for **..**. Also see **−X**.

−I *port* This option is only valid with the **−b** option. It causes *port* to use used as the port to bind daemon mode. The default is the "http" port.

−i *address*
This option is only valid with the **−b** option. It causes *address* to use used as the address to bind daemon mode. If otherwise unspecified, the address used to bind is derived from the *myname*, which defaults to the name returned by gethostname(3).

−M *suffix type encoding encoding11*
This option adds a new entry to the table that converts file suffixes to content type and encoding. This option takes four additional arguments containing the file prefix, its "Content-Type", "Content-Encoding" and "Content-Encoding" for HTTP/1.1 connections, respectively. If any of these are a single "-" (dash), the empty string is used instead. Multiple **−M** options may be passed.

−n This option stops **httpd** from doing IP address to name resolution of hosts for setting the `REMOTE_HOST` variable before running a CGI program. This option has no effect without the **−c** option.

−p *pubdir*
This option changes the default user directory for */˜user/* translations from "public_html" to *pubdir*.

−r This option forces pages besides the "index.html" (see the **−X** option) page to require that the Referrer: header be present and refer to this web server, otherwise a redirect to the "index.html" page will be returned instead.

−S *server_software*
This option sets the internal server version to *server_software*.

−s This option forces logging to be set to stderr always.

−t *chrootdir*
When this option is used, **httpd** will chroot to the specified directory before answering requests. Every other path should be specified relative to the new root, if this option is used. Note that the current environment is normally replaced with an empty environment with this option, unless the **−e** option is also used.

−U *username*
This option causes **httpd** to switch to the user and the groups of *username* after initialization. This option, like **−t** above, causes **httpd** to clear the environment unless the **−e** option is given.

−u This option enables the transformation of Uniform Resource Locators of the form */˜user/* into the the directory `˜user/public_html` (but see the **−p** option above).

−V This option sets the default virtual host directory to *slashdir*. If no directory exists in *virtualroot* for the request, then *slashdir* will be used. The default behaviour is to return 404 (Not Found.)

−v *virtualroot*
This option enables virtual hosting support. Directories in *virtualroot* will be searched for a matching virtual host name, when parsing the HTML request. If a matching name is found, it will be used as both the server's real name, [*myname*], and as the *slashdir*. See the **EXAMPLES** section for an example of using this option.

−X This option enables directory indexing. A directory index will be generated only when the default file (i.e. `index.html` normally) is not present.

−x *index* This option changes the default file read for directories from "index.html" to *index*.

−Z *certificate_path privatekey_path*
This option sets the path to the server certificate file and the private key file in pem format. It also causes **httpd** to start SSL mode.

Note that in **httpd** versions 20031005 and prior that supported the **−C** and **−M** options, they took a single space-separated argument that was parsed. since version 20040828, they take multiple options (2 in the case of **−C** and 4 in the case of **−M**.)

INETD CONFIGURATION

As **httpd** uses `inetd`(8) by default to process incoming TCP connections for HTTP requests (but see the **−b** option), **httpd** has little internal networking knowledge. (Indeed, you can run it on the command line with little change of functionality.) A typical `inetd.conf`(5) entry would be:

```
http stream tcp  nowait:600 _httpd /usr/libexec/httpd httpd /var/www
http stream tcp6 nowait:600 _httpd /usr/libexec/httpd httpd /var/www
```

This would serve web pages from `/var/www` on both IPv4 and IPv6 ports. The *:600* changes the requests per minute to 600, up from the `inetd`(8) default of 40.

Using the NetBSD `inetd`(8), you can provide multiple IP-address based HTTP servers by having multiple listening ports with different configurations.

EXAMPLES

To configure set of virtual hosts, one would use an `inetd.conf`(5) entry like:

```
http stream tcp  nowait:600 _httpd /usr/libexec/httpd httpd -v /var/vroot /var/www
```

and inside `/var/vroot` create a directory (or a symlink to a directory) with the same name as the virtual host, for each virtual host. Lookups for these names are done in a case-insensitive manner.

To use **httpd** with PHP, one must use the **−C** option to specify a CGI handler for a particular file type. Typically this, this will be like:

```
httpd -C .php /usr/pkg/bin/php /var/www
```

NOTES

This server supports the *HTTP/0.9*, *HTTP/1.0* and *HTTP/1.1* standards. Support for these protocols is very minimal and many optional features are not supported.

httpd can be compiled without CGI support (NO_CGIBIN_SUPPORT), user transformations (NO_USER_SUPPORT), directory index support (NO_DIRINDEX_SUPPORT), daemon mode support (NO_DAEMON_MODE), and dynamic MIME content (NO_DYNAMIC_CONTENT), and SSL support (NO_SSL_SUPPORT) by defining the listed macros when building **httpd**.

HTTP BASIC AUTHORISATION

httpd has support for HTTP Basic Authorisation. If a file named `.htpasswd` exists in the directory of the current request, **httpd** will restrict access to documents in that directory using the RFC 2617 HTTP "Basic" authentication scheme.

Note: This does not recursively protect any sub-directories.

The `.htpasswd` file contains lines delimited with a colon containing usernames and passwords hashed with `crypt`(3), for example:

```
heather:$1$pZWI4tH/$DzDPl63i6VvVRv2lJNV7k1
jeremy:A.xewbx2DpQ8I
```

On NetBSD, the `pwhash`(1) utility may be used to generate hashed passwords.

While **httpd** distributed with NetBSD has support for HTTP Basic Authorisation enabled by default, the portable distribution it is excluded. Compile **httpd** with "-DDO_HTPASSWD" on the compiler command line to enable this support. It may require linking with the crypt library, using "-lcrypt".

FILES

httpd looks for a couple of special files in directories that allow certain features to be provided on a per-directory basis. In addition to the `.htpasswd` used by HTTP basic authorisation, if a `.bzdirect` file is found (contents are irrelevant) **httpd** will allow direct access even with the **−r** option. If a `.bzredirect` symbolic link is found, **httpd** will perform a smart redirect to the target of this symlink. The target is assumed to live on the same server. If a `.bzabsredirect` symbolic link is found, **httpd** will redirect to the absolute url pointed to by this symlink. This is useful to redirect to different servers.

SSL SUPPORT

httpd has support for SSLv2, SSLv3, and TLSv1 protocols that is included by default. It requires linking with the crypto and ssl library, using "-lcrypto -lssl". To disable SSL SUPPORT compile **httpd** with "-DNO_SSL_SUPPORT" on the compiler command line.

SEE ALSO

`inetd.conf`(5), `inetd`(8)

HISTORY

The **httpd** program was first written in perl, based on another perl http server called "tinyhttpd". It was then rewritten from scratch in perl, and then once again in C. It was known for many years as "bozohttpd". "bozohttpd" version 20060517 was integrated into NetBSD 5.0 as **httpd**. The focus has always been simplicity and security, with minimal features and regular code audits. This manual documents **httpd** version 20080303.

AUTHORS

httpd was written by Matthew R. Green ⟨mrg@eterna.com.au⟩.

The large list of contributors includes:

- Arnaud Lacombe ⟨alc@netbs.dorg⟩ provided some clean up for memory leaks
- Christoph Badura ⟨bad@bsd.de⟩ provided Range: header support
- Julian Coleman ⟨jdc@coris.org.uk⟩ provided an IPv6 bugfix
- Chuck Cranor ⟨chuck@research.att.com⟩ provided cgi-bin support fixes, and more
- Andrew Doran ⟨ad@netbsd.org⟩ provided directory indexing support
- Per Ekman ⟨pek@pdc.kth.se⟩ provided a fix for a minor (non-security) buffer overflow condition
- Zak Johnson ⟨zakj@nox.cx⟩ provided cgi-bin enhancements
- Jun-ichiro itojun Hagino, KAME ⟨itojun@iijlab.net⟩ provided initial IPv6 support
- Martin Husemann ⟨martin@netbsd.org⟩ provided .bzabsredirect support
- Roland Illig ⟨roland.illig@gmx.de⟩ provided some off-by-one fixes
- Nicolas Jombart ⟨ecu@ipv42.net⟩ provided fixes for HTTP basic authorisation support
- Thomas Klausner ⟨wiz@danbala.ifoer.tuwien.ac.at⟩ provided many fixes and enhancements for the man page
- Johnny Lam ⟨jlam@netbsd.org⟩ provided man page fixes
- Luke Mewburn ⟨lukem@netbsd.org⟩ provided many various fixes, including cgi-bin fixes & enhancements, HTTP basic authorisation support and much code clean up
- Jeremey Reed ⟨reed@netbsd.org⟩ provided several clean up fixes, and man page updates
- Scott Reynolds ⟨scottr@netbsd.org⟩ provided various fixes
- Tyler Retzlaff ⟨rtr@eterna.com.au⟩ provided SSL support, cgi-bin fixes and much other random other stuff
- Steve Rumble ⟨rumble@ephemeral.org⟩ provided the **−V** option.
- ISIHARA Takanori ⟨ishit@oak.dti.ne.jp⟩ provided a man page fix

- ⟨xs@kittenz.org⟩ provided chroot and change-to-user support, and other various fixes

There are probably others I have forgotten (let me know if you care)

BUGS

httpd does not handled HTTP/1.1 chunked input from the client yet.

NAME

brconfig — configure network bridge parameters

SYNOPSIS

brconfig −a
brconfig *bridge*
brconfig *bridge command* [*args ...*]

DESCRIPTION

The **brconfig** utility is used to configure network bridge parameters and retrieve network bridge parameters and status from the kernel. The bridging function is implemented by the `bridge`(4) driver.

A network bridge creates a logical link between two or more IEEE 802 networks that use the same (or "similar enough") framing format. For example, it is possible to bridge Ethernet and 802.11 networks together, but it is not possible to bridge Ethernet and Token Ring together.

Bridge interfaces are created using the `ifconfig`(8) command's "create" sub-command. All other bridge configuration is performed using **brconfig**.

The options are as follows:

−a Display the status of all bridge devices present on the system. This flag is mutually exclusive with all other sub-commands.

All other operations require that a bridge be specified. If a bridge is specified with no sub-commands, the status of that bridge is displayed. The following sub-commands are available:

up Start forwarding packets on the bridge.

down Stop forwarding packets on the bridge.

add *interface*
Add the interface named by *interface* as a member of the bridge. The interface is put into promiscuous mode so that it can receive every packet sent on the network.

delete *interface*
Remove the interface named by *interface* from the bridge. Promiscuous mode is disabled on the interface when it is removed from the bridge.

maxaddr *size*
Set the size of the bridge address cache to *size*. The default is 100 entries.

timeout *seconds*
Set the timeout of address cache entries to *seconds* seconds. If *seconds* is zero, then address cache entries will not be expired. The default is 1200 seconds.

deladdr *address*
Delete *address* from the address cache.

flush Delete all dynamically-learned addresses from the address cache.

flushall
Delete all addresses, including static addresses, from the address cache.

discover *interface*
Mark an interface as a "discovering" interface. When the bridge has no address cache entry (either dynamic or static) for the destination address of a packet, the bridge will forward the packet to all member interfaces marked as "discovering". This is the default for all interfaces added to a bridge.

-discover *interface*
: Clear the "discovering" attribute on a member interface. For packets without the "discovering" attribute, the only packets forwarded on the interface are broadcast or multicast packets and packets for which the destination address is known to be on the interface's segment.

ipf
: Enable packet filtering with `pfil`(9) on the bridge. The current implementation passes all ARP and RARP packets through the bridge while filtering IP and IPv6 packets through the configured packet filter, such as `ipf`(4) or `pf`(4). Other packet types are blocked.

learn *interface*
: Mark an interface as a "learning" interface. When a packet arrives on such an interface, the source address of the packet is entered into the address cache as being a destination address on the interface's segment. This is the default for all interfaces added to a bridge.

-learn *interface*
: Clear the "learning" attribute on a member interface.

stp *interface*
: Enable Spanning Tree protocol on *interface*. The `bridge`(4) driver has support for the IEEE 802.1D Spanning Tree protocol (STP). Spanning Tree is used to detect and remove loops in a network topology.

-stp *interface*
: Disable Spanning Tree protocol on *interface*. This is the default for all interfaces added to a bridge.

maxage *seconds*
: Set the time that a Spanning Tree protocol configuration is valid. The default is 20 seconds. The minimum is 1 second and the maximum is 255 seconds.

fwddelay *seconds*
: Set the time that must pass before an interface begins forwarding packets when Spanning Tree is enabled. The default is 15 seconds. The minimum is 1 second and the maximum is 255 seconds.

hellotime *seconds*
: Set the time between broadcasting of Spanning Tree protocol configuration messages. The default is 2 seconds. The minimum is 1 second and the maximum is 255 seconds.

priority *value*
: Set the bridge priority for Spanning Tree. The default is 32768. Allowed numerical values range from 0 (highest priority) to 65535 (lowest priority).

ifpriority *interface value*
: Set the Spanning Tree priority of *interface* to *value*. The default is 128. The minimum is 0 and the maximum is 255.

ifpathcost *interface value*
: Set the Spanning Tree path cost of *interface* to *value*. The default is 55. The minimum is 0 and the maximum is 65535.

EXAMPLES

The following, when placed in the file `/etc/ifconfig.bridge0`, will cause a bridge called 'bridge0' to be created, add the interfaces 'ray0' and 'fxp0' to the bridge, and then enable packet forwarding. Such a configuration could be used to implement a simple 802.11-to-Ethernet bridge (assuming the 802.11 interface is in ad-hoc mode).

```
create
!brconfig $int add ray0 add fxp0 up
```

Consider a system with two 4-port Ethernet boards. The following placed in the file `/etc/ifconfig.bridge0` will cause a bridge consisting of all 8 ports with Spanning Tree enabled to be created:

```
create
!brconfig $int \
    add tlp0 stp tlp0 \
    add tlp1 stp tlp1 \
    add tlp2 stp tlp2 \
    add tlp3 stp tlp3 \
    add tlp4 stp tlp4 \
    add tlp5 stp tlp5 \
    add tlp6 stp tlp6 \
    add tlp7 stp tlp7 \
    up
```

SEE ALSO

`bridge`(4), `pf`(4), `ifconfig.if`(5), `ifconfig`(8), `ipf`(8), `pfil`(9)

HISTORY

The **brconfig** utility first appeared in NetBSD 1.6.

AUTHORS

The `bridge`(4) driver and **brconfig** utility were originally written by Jason L. Wright ⟨jason@thought.net⟩ as part of an undergraduate independent study at the University of North Carolina at Greensboro.

This version of the **brconfig** utility was written from scratch by
Jason R. Thorpe ⟨thorpej@wasabisystems.com⟩.

NAME

btattach — attach serial lines as Bluetooth HCI interfaces

SYNOPSIS

btattach [**−dfop**] [**−i** *speed*] [*type*] *tty speed*

DESCRIPTION

btattach is used to assign a *tty* line to a Bluetooth Host Controller Interface using the `btuart`(4) or `bcsp`(4) line disciplines, and can optionally initialize the line for a given device *type* before activating the line discipline.

Supported types are:

bcm2035	Broadcom BCM2035
bcsp	Generic BCSP (BlueCore Serial Protocol)
bgb2xx	Philips BGB2xx module
btuart	Generic UART (this is the default)
csr	Cambridge Silicon Radio Casira serial adaptor, or Brainboxes serial dongle (BL642)
ericsson	Ericsson based modules
digi	Digianswer based cards
st	ST Microelectronics minikits based on STLC2410/STLC2415
stlc2500	ST Microelectronics minikits based on STLC2500
swave	Silicon Wave kits
texas	Texas Instruments modules

When the line discipline is activated, **btattach** detaches and sleeps until it receives a `SIGHUP`.

The command line options are as follows:

−d	debug mode. print initialization IO and do not detach.
−f	Enable flow control.
−i *speed*	Specify an alternate *speed* for the Bluetooth module to use during the initialization phase.
−o	Enable odd parity.
−p	Enable parity (even parity).

Only the super-user may attach a Bluetooth HCI interface.

FILES

`/var/run/btattach-`{tty}.pid

SEE ALSO

`bcsp`(4), `bluetooth`(4), `btuart`(4), `btconfig`(8)

BUGS

Not all *type* initializations have been tested.

HISTORY

The **btattach** program was written with reference to `hciattach`(8) as provided with the BlueZ tools for Linux and first appeared in NetBSD 5.0.

AUTHORS

KIYOHARA Takashi ⟨kiyohara@kk.iij4u.or.jp⟩
Iain Hibbert

NAME

btconfig — configure bluetooth devices

SYNOPSIS

btconfig [**−svz**] [*device* [*parameters*]]
btconfig [**−l**]

DESCRIPTION

btconfig is used to configure Bluetooth devices. If the *device* is given, but no parameters, then **btconfig** will print information about the device. If no *device* is given, a basic list of devices will be printed.

When the **−l** flag is used, just the device names will be printed.

COMMANDS

The following parameters may be specified with **btconfig**:

up	Enable Bluetooth Device.
down	Disable Bluetooth Device.
pscan	Enable Page Scan. This enables incoming connections to the device.
-pscan	Disable Page Scan.
iscan	Enable Inquiry Scan. This puts the device in Discoverable mode.
-iscan	Disable Inquiry Scan.
encrypt	Enable encryption. This will cause the device to request encryption on all baseband connections, and will only work if authentication is also enabled.
-encrypt	Disable encryption.
auth	Enable authentication. This will cause the device to request authentication for all baseband connections.
-auth	Disable authentication.
switch	Enable Role Switching.
-switch	Disable Role Switching.
hold	Enable Hold Mode.
-hold	Disable Hold Mode.
sniff	Enable Sniff Mode.
-sniff	Disable Sniff Mode.
park	Enable Park Mode.
-park	Disable Park Mode.
name *name*	Set human readable name of device.
ptype *type*	Set packet types. *type* is a 16 bit hex value specifying packet types that will be requested by outgoing ACL connections. By default, all packet types that the device supports are enabled, see bluetooth specifications for more information if you want to change this.

class *class* Set class of device. *class* is a 3 byte hex value the value of which declares the device capabilities. See Bluetooth Assigned Numbers documents at `https://www.bluetooth.org/` for details of constructing a "Class of Device" value. As a starter, 0x020104 means Desktop Computer, with Networking available.

fixed Set fixed pin type.

variable Set variable pin type.

inquiry Perform device Discovery from the specified device and print details.

rssi Enable Remote Signal Strength Indicator (RSSI) in inquiry results. This will only work if the device features indicate ⟨RSSI with inquiry result⟩.

-rssi Disable Remote Signal Strength Indicator (RSSI) in inquiry results.

reset Perform a hard reset on the device and re-initialise system state.

voice Set Voice Setting. [This should be 0x0060 for now]

pto Set Page Timeout value. This is a decimal value in milliseconds.

scomtu Change SCO mtu value. This is a decimal value, see ubt(4) for reasons why you may need to do this.

All parameters are parsed before any device operations take place. Each time the **−v** flag is given, verbosity levels will be increased.

Super-user privileges are required to change device configurations.

DIAGNOSTICS

Messages indicating the specified device does not exist, the requested address is unknown, or the user is not privileged and tried to alter an device's configuration.

SEE ALSO

bluetooth(4), bt3c(4), btuart(4), ubt(4)

HISTORY

The **btconfig** command was written for NetBSD 4.0 by Iain Hibbert under the sponsorship of Itronix, Inc.

BUGS

The output is very messy.

NAME

btdevctl — Bluetooth remote device control utility

SYNOPSIS

btdevctl [**-A** | **-D**] [**-qv**] [**-m** *mode*] **-a** *address* **-d** *device* **-s** *service*

DESCRIPTION

The **btdevctl** utility is used to configure bluetooth devices in the system. Normally, **btdevctl** will perform an SDP query to the remote device as needed, and cache the results in the `/var/db/btdevctl.plist` file for later use. If neither Attach nor Detach is specified, **btdevctl** will display the configuration.

The options are:

-A Attach device

-a *address*
Remote device address. The *address* may be given as BDADDR or a name. If a name was specified, **btdevctl** attempts to resolve the name via `bt_gethostbyname`(3).

-D Detach device

-d *device*
Local device address. May be given as BDADDR or device name.

-m *mode* Connection link mode. The following modes are supported:

none	clear previously set mode.
auth	require devices be paired, see `btpin`(1).
encrypt	auth, plus enable encryption.
secure	encryption, plus change of link key.

When configuring the HID service, **btdevctl** will set 'auth' by default, or 'encrypt' for keyboard devices.

-q Ignore any cached data and perform a SDP query for the given *service*.

-s *service*
Service to configure. Known services are:

`HID`	Human Interface Device.
`HF`	Handsfree.
`HSET`	Headset.

-v Be verbose.

For device configurations to persist across boots, add entries to the `/etc/bluetooth/btdevctl.conf` file and set the `rc.conf`(5) variable **btdevctl** to YES.

EXIT STATUS

The **btdevctl** utility exits 0 on success, and >0 if an error occurs.

FILES

```
/etc/bluetooth/btdevctl.conf
/dev/bthub
/var/db/btdevctl.plist
```

SEE ALSO

`btpin`(1), `bthidev`(4), `bthub`(4), `btsco`(4), `rc.conf`(5)

HISTORY

Parts of the **`btdevctl`** program originated in the FreeBSD **`bthidcontrol`** program.

AUTHORS

Iain Hibbert for Itronix, Inc.
Maksim Yevmenkin ⟨m_evmenkin@yahoo.com⟩

NAME

bthcid — Bluetooth Link Key/PIN Code Manager

SYNOPSIS

bthcid [**-fn**] [**-d** *device*] [**-m** *mode*] [**-s** *socket_name*]
bthcid [**-h**]

DESCRIPTION

The **bthcid** daemon handles Link Key and PIN code requests for Bluetooth devices. It opens a raw HCI socket and listens for the following HCI events.

`Link_Key_Request`
: **bthcid** scans the `/var/db/bthcid.keys` file for a cached link key matching the remote device BD_ADDR and, if found, the `Link_Key_Request_Reply` will be sent back to the device, otherwise the `Link_Key_Request_Negative_Reply` will be sent.

`Link_Key_Notification`
: When a new link key is created by the device, it will be cached for future use in the `/var/db/bthcid.keys` link keys file, which will be created if it does not already exist.

`PIN_Code_Request`
: The **bthcid** daemon checks its PIN cache for a matching remote device entry. If no PIN is found, the **bthcid** daemon will send a message to any PIN clients that have registered, with the device details and a timeout value. When no clients are available or the timeout has expired, **bthcid** will send a `PIN_Code_Request_Negative_Reply` back to the device. When a PIN is found, or if a client responds within the timeout period, a `PIN_Code_Request_Reply` will be sent back to the device.

PINs received from clients will be cached for 5 minutes until used, and may be added to the cache prior to pairing with the `btpin`(1) utility.

Some of the functionality of **bthcid** can be handled by the Bluetooth controller directly, and cached Link Keys may be examined, deleted or moved to device storage using the `btkey`(1) program.

The command line options are as follows:

-d *device*
: Specify the local Bluetooth device address. The default is BDADDR_ANY.

-f Run in foreground (do not detach).

-h Display usage message and exit.

-m Specify the file mode access bits for the PIN client socket. The default is to allow readwrite access to user and group (0660).

-n Do not listen for PIN clients.

-s *socket_name*
: Specify the socket name to listen on for PIN clients. The default path is `/var/run/bthcid`.

FILES

`/var/db/bthcid.keys`
`/var/run/bthcid`
`/var/run/bthcid.pid`

SEE ALSO

`btpin`(1), `btkey`(1), `bluetooth`(4), `btconfig`(8)

HISTORY

The **bthcid** daemon first appeared in FreeBSD 5.3 as **hcsecd**. It was ported to NetBSD 4.0 with its present name and extended to support PIN clients by Iain Hibbert under the sponsorship of Itronix, Inc.

AUTHORS

Maksim Yevmenkin ⟨m_evmenkin@yahoo.com⟩
Iain Hibbert

NAME

catman — format cat pages from man pages

SYNOPSIS

catman [**−knpsw**] [**−m** *directory*] [*sections*]
catman [**−knpsw**] [**−M** *directory*] [*sections*]

DESCRIPTION

catman creates formatted versions of the on-line manual pages from their nroff(1) source. Manual pages whose formatted versions are missing or out of date are regenerated. If manual pages are regenerated, **catman** also regenerates the whatis database.

The optional *sections* argument is one word, and contains the section numbers of all the sections to be checked. For example, if *sections* is "13f8", the manual pages in sections 1, 3f, and 8 will be checked and regenerated. If no *sections* argument is provided, **catman** will try to operate on all of the known manual sections.

The options are as follows:

−k Ignore errors from nroff when building manpages.

−n Do not create the whatis database.

−p Display the commands that would have been executed, but do not actually execute them.

−s Perform work silently; do not echo commands as they are executed. This flag is ignored if **−p** is specified.

−w Only create the whatis database.

−m *directory*
Add *directory* to the set of directories to be updated.

−M *directory*
Update manual pages in *directory*.

SEE ALSO

apropos(1), man(1), whatis(1)

BUGS

Currently does not handle hard links.

NAME

ccdconfig — configuration utility for the concatenated disk driver

SYNOPSIS

ccdconfig [**-cv**] *ccd ileave* [*flags*] *dev* [*...*]
ccdconfig -C [**-v**] [**-f** *config_file*]
ccdconfig -u [**-v**] *ccd* [*...*]
ccdconfig -U [**-v**] [**-f** *config_file*]
ccdconfig -g [**-M** *core*] [**-N** *system*] [*ccd* [...]]

DESCRIPTION

ccdconfig is used to dynamically configure and unconfigure concatenated disk devices, or ccds. For more information about the ccd, see ccd(4).

The options are as follows:

-c Configure a ccd. This is the default behavior of **ccdconfig**.

-C Configure all ccd devices listed in the ccd configuration file.

-f *config_file*
When configuring or unconfiguring all devices, read the file config_file instead of the default /etc/ccd.conf.

-g Dump the current ccd configuration in a format suitable for use as the ccd configuration file. If no arguments are specified, every configured ccd is dumped. Otherwise, the configuration of each listed ccd is dumped.

-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.

-u Unconfigure a ccd.

-U Unconfigure all ccd devices listed the ccd configuration file.

-v Causes **ccdconfig** to be verbose.

A ccd is described on the command line and in the ccd configuration file by the name of the ccd, the interleave factor, the ccd configuration flags, and a list of one or more devices. The flags may be represented as a decimal number, a hexadecimal number, a comma-separated list of strings, or the word "none". The flags are as follows:

Symbolic	*Numeric*	*Comment*
CCDF_UNIFORM	0x02	Use uniform interleave. The size of all components is clamped to that of the smallest component.
CCDF_NOLABEL	0x04	Ignore raw disklabel. Useful when creating a new ccd.

/etc/ccd.conf

The file /etc/ccd.conf is used to configure **ccdconfig** if **-C** or **-U** is used. Each line of the configuration file contains arguments as per the **-c** argument: *ccd ileave* [*flags*] *dev* [*...*]

A '#' is a comment, and everything to end of line is ignored. A '\' at the end of a line indicates that the next line should be concatenated with the current. A '\' preceding any character (other than the end of line) prevents that character's special meaning from taking effect.

See **EXAMPLES** for an example of `/etc/ccd.conf`.

FILES

/etc/ccd.conf - default ccd configuration file.

EXAMPLES

The following command, executed from the command line, would configure ccd0 with 4 components (/dev/sd2e, /dev/sd3e, /dev/sd4e, /dev/sd5e), and an interleave factor of 32 blocks.

```
# ccdconfig ccd0 32 0 /dev/sd2e /dev/sd3e /dev/sd4e /dev/sd5e
```

An example `/etc/ccd.conf`:

```
#
# /etc/ccd.conf
# Configuration file for concatenated disk devices
#

# ccd       ileave  flags   component devices
ccd0        16      none    /dev/sd2e /dev/sd3e
```

SEE ALSO

`ccd`(4), `ccd.conf`(5), `rc`(8)

HISTORY

The **`ccdconfig`** command first appeared in NetBSD 1.1.

NAME

cgdconfig — configuration utility for the cryptographic disk driver

SYNOPSIS

cgdconfig [**-npv**] [**-V** *vmeth*] *cgd dev* [*paramsfile*]
cgdconfig -C [**-nv**] [**-f** *configfile*]
cgdconfig -U [**-nv**] [**-f** *configfile*]
cgdconfig -G [**-nv**] [**-i** *ivmeth*] [**-k** *kgmeth*] [**-o** *outfile*] *paramsfile*
cgdconfig -g [**-nv**] [**-i** *ivmeth*] [**-k** *kgmeth*] [**-o** *outfile*] *alg* [*keylen*]
cgdconfig -s [**-nv**] [**-i** *ivmeth*] *cgd dev alg* [*keylen*]
cgdconfig -u [**-nv**] *cgd*

DESCRIPTION

cgdconfig is used to configure and unconfigure cryptographic disk devices (cgds) and to maintain the configuration files that are associated with them. For more information about cryptographic disk devices see cgd(4).

The options are as follows:

-C	Configure all the devices listed in the cgd configuration file.
-f *configfile*	Specify the configuration file explicitly, rather than using the default configuration file /etc/cgd/cgd.conf.
-G	Generate a new paramsfile (to stdout) using the values from *paramsfile* which will generate the same key. This may need to prompt for multiple passphrases.
-g	Generate a paramsfile (to stdout).
-i *ivmeth*	Specify the IV method (default: encblkno).
-k *kgmeth*	Specify the key generation method (default: pkcs5_pbkdf2/sha1).
-o *outfile*	When generating a *paramsfile*, store it in *outfile*.
-p	Read all passphrases from stdin rather than /dev/tty. Passphrases are separated by newlines. Users of this flag must be able to predict the order in which passphrases are prompted. If this flag is specified then verification errors will cause the device in question to be unconfigured rather than prompting for the passphrase again.
-s	Read the key from stdin.
-U	Unconfigure all the devices listed in the cgd configuration file.
-u	Unconfigure a cgd.
-V *vmeth*	Specify the verification method (default: none).
-v	Be verbose. May be specified multiple times.

For more information about the cryptographic algorithms and IV methods supported, please refer to cgd(4).

Key Generation Methods

To generate the key which it will use, **cgdconfig** evaluates all of the key generation methods in the parameters file and uses the exclusive-or of the outputs of all the methods. The methods and descriptions are as follows:

pkcs5_pbkdf2/sha1	This method requires a passphrase which is entered at configuration time. It is a salted hmac-based scheme detailed in "PKCS#5 v2.0: Password-Based Cryptography Standard", RSA Laboratories, March 25, 1999, pages 8-10. PKCS

	#5 was also republished as RFC 2898.
pkcs5_pbkdf2	This is an earlier, slightly incorrect and deprecated implementation of the above algorithm. It is retained for backwards compatibility with existing parameters files, and will be removed. Existing parameters files should be converted to use the correct method using the **−G** option, and a new passphrase.
storedkey	This method stores its key in the parameters file.
randomkey	The method simply reads `/dev/random` and uses the resulting bits as the key. It does not require a passphrase to be entered. This method is typically used to present disk devices that do not need to survive a reboot, such as the swap partition. It is also handy to facilitate overwriting the contents of a disk volume with meaningless data prior to use.
urandomkey	The method simply reads `/dev/urandom` and uses the resulting bits as the key. This is similar to the `randomkey` method, but it guarantees that cgdconfig will not stall waiting for hard-random bits (useful when configuring a cgd for swap at boot time). Note, however, that some or all of the bits used to generate the key may be obtained from a pseudo-random number generator, which may not be as secure as the entropy based hard-random number generator.
shell_cmd	This method executes a shell command via `popen`(3) and reads the key from stdout.

Verification Method

The verification method is how **cgdconfig** determines if the generated key is correct. If the newly configured disk fails to verify, then **cgdconfig** will regenerate the key and re-configure the device. It only makes sense to specify a verification method if at least of the key generation methods is error prone, e.g., uses a user-entered passphrase. The following verification methods are supported:

none	perform no verification.
disklabel	scan for a valid disklabel.
ffs	scan for a valid FFS file system.
re-enter	prompt for passphrase twice, and ensure entered passphrases are identical. This method only works with the pkcs5_pbkdf2/sha1 and pkcs5_pbkdf2 key generators.

/etc/cgd/cgd.conf

The file `/etc/cgd/cgd.conf` is used to configure **cgdconfig** if either of **−C** or **−U** are specified. Each line of the file is composed of either two or three tokens: cgd, target, and optional paramsfile.

A '#' character is interpreted as a comment and indicates that the rest of the line should be ignored. A '\' at the end of a line indicates that the next line is a continuation of the current line.

See **EXAMPLES** for an example of `/etc/cgd/cgd.conf`.

Parameters File

The Parameters File contains the required information to generate the key and configure a device. These files are typically generated by the **−g** flag and not edited by hand. When a device is configured the default parameters file is constructed by taking the basename of the target disk and prepending `/etc/cgd/` to it. E.g., if the target is `/dev/sd0h`, then the default parameters file will be `/etc/cgd/sd0h`.

It is possible to have more than one parameters file for a given disk which use different key generation methods but will generate the same key. To create a parameters file that is equivalent to an existing parameters file, use **cgdconfig** with the **−G** flag. See **EXAMPLES** for an example of this usage.

The parameters file contains a list of statements each terminated with a semi-colon. Some statements can contain statement-blocks which are either a single unadorned statement, or a brace-enclosed list of semicolon terminated statements. Three types of data are understood:

integer	a 32 bit signed integer.
string	a string.
base64	a length-encoded base64 string.

The following statements are defined:

algorithm *string*
  Defines the cryptographic algorithm.

iv-method *string*
  Defines the IV generation method.

keylength *integer*
  Defines the length of the key.

verify_method *string*
  Defines the verification method.

keygen *string statement_block*
  Defines a key generation method. The *statement_block* contains statements that are specific to the key generation method.

The keygen statement's statement block may contain the following statements:

key *string*
  The key. Only used for the storedkey key generation method.

cmd *string*
  The command to execute. Only used for the shell_cmd key generation method.

iterations *integer*
  The number of iterations. Only used for pkcs5_pbkdf2/sha1 and pkcs5_pbkdf2.

salt *base64*
  The salt. Only used for pkcs5_pbkdf2/sha1 and pkcs5_pbkdf2.

FILES

`/etc/cgd/`	configuration directory, used to store paramsfiles.
`/etc/cgd/cgd.conf`	cgd configuration file.

EXAMPLES

To set up and configure a cgd that uses AES with a 192 bit key in CBC mode with the IV Method 'encblkno' (encrypted block number):

```
# cgdconfig -g -o /etc/cgd/wd0e aes-cbc 192
# cgdconfig cgd0 /dev/wd0e
/dev/wd0e's passphrase:
```

When using verification methods, the first time that we configure the disk the verification method will fail. We overcome this by supplying **-V** *re-enter* when we configure the first time to set up the disk. Here is the sequence of commands that is recommended:

```
# cgdconfig -g -o /etc/cgd/wd0e -V disklabel aes-cbc
# cgdconfig -V re-enter cgd0 /dev/wd0e
/dev/wd0e's passphrase:
```

```
              re-enter device's passphrase:
              # disklabel -e -I cgd0
              # cgdconfig -u cgd0
              # cgdconfig cgd0 /dev/wd0e
              /dev/wd0e's passphrase:
```

To create a new parameters file that will generate the same key as an old parameters file:

```
              # cgdconfig -G -o newparamsfile oldparamsfile
              old file's passphrase:
              new file's passphrase:
```

To configure a cgd that uses Blowfish with a 200 bit key that it reads from stdin:

```
        # cgdconfig -s cgd0 /dev/sd0h blowfish-cbc 200
```

An example parameters file which uses PKCS#5 PBKDF2:

```
        algorithm aes-cbc;
        iv-method encblkno;
        keylength 128;
        verify_method none;
        keygen pkcs5_pbkdf2/sha1 {
                iterations 39361;
                salt AAAAgMoHiYonye6Kog \
                     dYJAobCHE=;
        };
```

An example parameters file which stores its key locally:

```
        algorithm       aes-cbc;
        iv-method       encblkno;
        keylength       256;
        verify_method   none;
        keygen storedkey key AAABAK3QO6d7xzLfrXTdsgg4 \
                            ly2TdxkFqOkYYcbyUKu/f60L;
```

An example `/etc/cgd/cgd.conf`:

```
        #
        # /etc/cgd/cgd.conf
        # Configuration file for cryptographic disk devices
        #

        # cgd          target         [paramsfile]
        cgd0           /dev/wd0e
        cgd1           /dev/sd0h      /usr/local/etc/cgd/sd0h
```

Note that this will store the parameters file as `/etc/cgd/wd0e`. And use the entered passphrase to generate the key.

DIAGNOSTICS

cgdconfig: could not calibrate pkcs5_pbkdf2 An error greater than 5% in calibration occured. This could be the result of dynamic processor frequency scaling technology. Ensure that the processor clock frequency remains static throughout the program's execution.

SEE ALSO

`cgd`(4)

"PKCS #5 v2.0: Password-Based Cryptography Standard", RSA Laboratories, March 25, 1999.

HISTORY

The **`cgdconfig`** utility appeared in NetBSD 2.0.

BUGS

Since **`cgdconfig`** uses `getpass`(3) to read in the passphrase, it is limited to 128 characters.

NAME

chat − Automated conversational script with a modem

SYNOPSIS

chat [*options*] *script*

DESCRIPTION

The *chat* program defines a conversational exchange between the computer and the modem. Its primary purpose is to establish the connection between the Point-to-Point Protocol Daemon (*pppd*) and the remote's *pppd* process.

OPTIONS

−f <*chat file*>
: Read the chat script from the chat *file*. The use of this option is mutually exclusive with the chat script parameters. The user must have read access to the file. Multiple lines are permitted in the file. Space or horizontal tab characters should be used to separate the strings.

−t <*timeout*>
: Set the timeout for the expected string to be received. If the string is not received within the time limit then the reply string is not sent. An alternate reply may be sent or the script will fail if there is no alternate reply string. A failed script will cause the *chat* program to terminate with a non-zero error code.

−r <*report file*>
: Set the file for output of the report strings. If you use the keyword *REPORT*, the resulting strings are written to this file. If this option is not used and you still use *REPORT* keywords, the *stderr* file is used for the report strings.

−e Start with the echo option turned on. Echoing may also be turned on or off at specific points in the chat script by using the *ECHO* keyword. When echoing is enabled, all output from the modem is echoed to *stderr*.

−E Enables environment variable substitution within chat scripts using the standard *$xxx* syntax.

−v Request that the *chat* script be executed in a verbose mode. The *chat* program will then log the execution state of the chat script as well as all text received from the modem and the output strings sent to the modem. The default is to log through the SYSLOG; the logging method may be altered with the −S and −s flags. SYSLOGs are logged to facility LOG_LOCAL2.

−V Request that the *chat* script be executed in a stderr verbose mode. The *chat* program will then log all text received from the modem and the output strings sent to the modem to the stderr device. This device is usually the local console at the station running the chat or pppd program.

−s Use stderr. All log messages from '−v' and all error messages will be sent to stderr.

−S Do not use the SYSLOG. By default, error messages are sent to the SYSLOG. The use of −S will prevent both log messages from '−v' and error messages from being sent to the SYSLOG (to facility LOG_LOCAL2).

−T <*phone number*>
: Pass in an arbitrary string, usually a phone number, that will be substituted for the \T substitution metacharacter in a send string.

−U <*phone number 2*>
: Pass in a second string, usually a phone number, that will be substituted for the \U substitution metacharacter in a send string. This is useful when dialing an ISDN terminal adapter that requires two numbers.

script If the script is not specified in a file with the *−f* option then the script is included as parameters to the *chat* program.

CHAT SCRIPT

The *chat* script defines the communications.

A script consists of one or more "expect−send" pairs of strings, separated by spaces, with an optional "subexpect−subsend" string pair, separated by a dash as in the following example:

```
ogin:−BREAK−ogin: ppp ssword: hello2u2
```

This line indicates that the *chat* program should expect the string "ogin:". If it fails to receive a login prompt within the time interval allotted, it is to send a break sequence to the remote and then expect the string "ogin:". If the first "ogin:" is received then the break sequence is not generated.

Once it received the login prompt the *chat* program will send the string ppp and then expect the prompt "ssword:". When it receives the prompt for the password, it will send the password hello2u2.

A carriage return is normally sent following the reply string. It is not expected in the "expect" string unless it is specifically requested by using the \r character sequence.

The expect sequence should contain only what is needed to identify the string. Since it is normally stored on a disk file, it should not contain variable information. It is generally not acceptable to look for time strings, network identification strings, or other variable pieces of data as an expect string.

To help correct for characters which may be corrupted during the initial sequence, look for the string "ogin:" rather than "login:". It is possible that the leading "l" character may be received in error and you may never find the string even though it was sent by the system. For this reason, scripts look for "ogin:" rather than "login:" and "ssword:" rather than "password:".

A very simple script might look like this:

```
ogin: ppp ssword: hello2u2
```

In other words, expect ....ogin:, send ppp, expect ...ssword:, send hello2u2.

In actual practice, simple scripts are rare. At the vary least, you should include sub-expect sequences should the original string not be received. For example, consider the following script:

```
ogin:−−ogin: ppp ssword: hello2u2
```

This would be a better script than the simple one used earlier. This would look for the same login: prompt, however, if one was not received, a single return sequence is sent and then it will look for login: again. Should line noise obscure the first login prompt then sending the empty line will usually generate a login prompt again.

COMMENTS

Comments can be embedded in the chat script. A comment is a line which starts with the **#** (hash) character in column 1. Such comment lines are just ignored by the chat program. If a ’#’ character is to be expected as the first character of the expect sequence, you should quote the expect string. If you want to wait for a prompt that starts with a # (hash) character, you would have to write something like this:

```
# Now wait for the prompt and send logout string
’# ’ logout
```

SENDING DATA FROM A FILE

If the string to send starts with an at sign (@), the rest of the string is taken to be the name of a file to read to get the string to send. If the last character of the data read is a newline, it is removed. The file can be a named pipe (or fifo) instead of a regular file. This provides a way for **chat** to communicate with another program, for example, a program to prompt the user and receive a password typed in.

ABORT STRINGS

Many modems will report the status of the call as a string. These strings may be **CONNECTED** or **NO CARRIER** or **BUSY**. It is often desirable to terminate the script should the modem fail to connect to the remote. The difficulty is that a script would not know exactly which modem string it may receive. On one attempt, it may receive **BUSY** while the next time it may receive **NO CARRIER**.

These "abort" strings may be specified in the script using the *ABORT* sequence. It is written in the script as in the following example:

```
ABORT BUSY ABORT ’NO CARRIER’ ’’ ATZ OK ATDT5551212 CONNECT
```

This sequence will expect nothing; and then send the string ATZ. The expected response to this is the string *OK*. When it receives *OK*, the string ATDT5551212 to dial the telephone. The expected string is *CONNECT*. If the string *CONNECT* is received the remainder of the script is executed. However, should the modem find a busy telephone, it will send the string *BUSY*. This will cause the string to match the abort character sequence. The script will then fail because it found a match to the abort string. If it received the string *NO CARRIER*, it will abort for the same reason. Either string may be received. Either string will terminate the *chat* script.

CLR_ABORT STRINGS

This sequence allows for clearing previously set **ABORT** strings. **ABORT** strings are kept in an array of a pre-determined size (at compilation time); **CLR_ABORT** will reclaim the space for cleared entries so that new strings can use that space.

SAY STRINGS

The **SAY** directive allows the script to send strings to the user at the terminal via standard error. If **chat** is being run by pppd, and pppd is running as a daemon (detached from its controlling terminal), standard error will normally be redirected to the file /etc/ppp/connect−errors.

SAY strings must be enclosed in single or double quotes. If carriage return and line feed are needed in the string to be output, you must explicitly add them to your string.

The SAY strings could be used to give progress messages in sections of the script where you want to have ’ECHO OFF’ but still let the user know what is happening. An example is:

```
ABORT BUSY
ECHO OFF
SAY "Dialing your ISP...\n"
’’ ATDT5551212
TIMEOUT 120
SAY "Waiting up to 2 minutes for connection ... "
CONNECT ’’
SAY "Connected, now logging in ...0
ogin: account
ssword: pass
$ SAY "Logged in OK ...0 etc ...
```

This sequence will only present the SAY strings to the user and all the details of the script will remain hidden. For example, if the above script works, the user will see:

```
Dialing your ISP...
Waiting up to 2 minutes for connection ... Connected, now logging in ...
Logged in OK ...
```

REPORT STRINGS

A **report** string is similar to the ABORT string. The difference is that the strings, and all characters to the next control character such as a carriage return, are written to the report file.

The report strings may be used to isolate the transmission rate of the modem’s connect string and return the value to the chat user. The analysis of the report string logic occurs in conjunction with the other string processing such as looking for the expect string. The use of the same string for a report and abort sequence is probably not very useful, however, it is possible.

The report strings to no change the completion code of the program.

These "report" strings may be specified in the script using the *REPORT* sequence. It is written in the script as in the following example:

```
REPORT CONNECT ABORT BUSY '' ATDT5551212 CONNECT '' ogin: account
```

This sequence will expect nothing; and then send the string ATDT5551212 to dial the telephone. The expected string is *CONNECT*. If the string *CONNECT* is received the remainder of the script is executed. In addition the program will write to the expect−file the string "CONNECT" plus any characters which follow it such as the connection rate.

CLR_REPORT STRINGS

This sequence allows for clearing previously set **REPORT** strings. **REPORT** strings are kept in an array of a pre-determined size (at compilation time); **CLR_REPORT** will reclaim the space for cleared entries so that new strings can use that space.

ECHO

The echo options controls whether the output from the modem is echoed to *stderr*. This option may be set with the *−e* option, but it can also be controlled by the *ECHO* keyword. The "expect−send" pair *ECHO ON* enables echoing, and *ECHO OFF* disables it. With this keyword you can select which parts of the conversation should be visible. For instance, with the following script:

```
ABORT   'BUSY'
ABORT   'NO CARRIER'
OK\r\n  ATD1234567
\r\n    \c
ECHO    ON
CONNECT \c
ogin:   account
```

all output resulting from modem configuration and dialing is not visible, but starting with the *CONNECT* (or *BUSY*) message, everything will be echoed.

HANGUP

The HANGUP options control whether a modem hangup should be considered as an error or not. This option is useful in scripts for dialing systems which will hang up and call your system back. The HANGUP options can be **ON** or **OFF**.
When HANGUP is set OFF and the modem hangs up (e.g., after the first stage of logging in to a callback system), **chat** will continue running the script (e.g., waiting for the incoming call and second stage login prompt). As soon as the incoming call is connected, you should use the **HANGUP ON** directive to reinstall normal hang up signal behavior. Here is an (simple) example script:

```
ABORT   'BUSY'
OK\r\n  ATD1234567
\r\n    \c
CONNECT \c
'Callback login:' call_back_ID
HANGUP OFF
ABORT "Bad Login"
'Callback Password:' Call_back_password
TIMEOUT 120
CONNECT \c
HANGUP ON
ABORT "NO CARRIER"
ogin:−−BREAK−−ogin: real_account
etc ...
```

TIMEOUT

The initial timeout value is 45 seconds. This may be changed using the **−t** parameter.

To change the timeout value for the next expect string, the following example may be used:

```
ATZ OK ATDT5551212 CONNECT TIMEOUT 10 ogin:−−ogin: TIMEOUT 5 assword: hello2u2
```

This will change the timeout to 10 seconds when it expects the login: prompt. The timeout is then changed to 5 seconds when it looks for the password prompt.

The timeout, once changed, remains in effect until it is changed again.

SENDING EOT

The special reply string of *EOT* indicates that the chat program should send an EOT character to the remote. This is normally the End-of-file character sequence. A return character is not sent following the EOT. The EOT sequence may be embedded into the send string using the sequence *ˆD*.

GENERATING BREAK

The special reply string of *BREAK* will cause a break condition to be sent. The break is a special signal on the transmitter. The normal processing on the receiver is to change the transmission rate. It may be used to cycle through the available transmission rates on the remote until you are able to receive a valid login prompt. The break sequence may be embedded into the send string using the *\K* sequence.

ESCAPE SEQUENCES

The expect and reply strings may contain escape sequences. All of the sequences are legal in the reply string. Many are legal in the expect. Those which are not valid in the expect sequence are so indicated.

’’	Expects or sends a null string. If you send a null string then it will still send the return character. This sequence may either be a pair of apostrophe or quote characters.
\b	represents a backspace character.
\c	Suppresses the newline at the end of the reply string. This is the only method to send a string without a trailing return character. It must be at the end of the send string. For example, the sequence hello\c will simply send the characters h, e, l, l, o. *(not valid in expect.)*
\d	Delay for one second. The program uses sleep(1) which will delay to a maximum of one second. *(not valid in expect.)*
\K	Insert a BREAK *(not valid in expect.)*
\n	Send a newline or linefeed character.
\N	Send a null character. The same sequence may be represented by \0. *(not valid in expect.)*
\p	Pause for a fraction of a second. The delay is 1/10th of a second. *(not valid in expect.)*
\q	Suppress writing the string to the SYSLOG. The string ?????? is written to the log in its place. *(not valid in expect.)*
\r	Send or expect a carriage return.
\s	Represents a space character in the string. This may be used when it is not desirable to quote the strings which contains spaces. The sequence ’HI TIM’ and HI\sTIM are the same.
\t	Send or expect a tab character.
\T	Send the phone number string as specified with the *−T* option *(not valid in expect.)*
\U	Send the phone number 2 string as specified with the *−U* option *(not valid in expect.)*
\\	Send or expect a backslash character.
\ddd	Collapse the octal digits (ddd) into a single ASCII character and send that character. *(some characters are not valid in expect.)*
ˆC	Substitute the sequence with the control character represented by C. For example, the character DC1 (17) is shown as ˆQ. *(some characters are not valid in expect.)*

ENVIRONMENT VARIABLES

Environment variables are available within chat scripts, if the *−E* option was specified in the command line. The metacharacter *$* is used to introduce the name of the environment variable to substitute. If the substitution fails, because the requested environment variable is not set, *nothing* is replaced for the variable.

TERMINATION CODES

The *chat* program will terminate with the following completion codes.

0 The normal termination of the program. This indicates that the script was executed without error to the normal conclusion.

1 One or more of the parameters are invalid or an expect string was too large for the internal buffers. This indicates that the program as not properly executed.

2 An error occurred during the execution of the program. This may be due to a read or write operation failing for some reason or chat receiving a signal such as SIGINT.

3 A timeout event occurred when there was an *expect* string without having a "−subsend" string. This may mean that you did not program the script correctly for the condition or that some unexpected event has occurred and the expected string could not be found.

4 The first string marked as an *ABORT* condition occurred.

5 The second string marked as an *ABORT* condition occurred.

6 The third string marked as an *ABORT* condition occurred.

7 The fourth string marked as an *ABORT* condition occurred.

... The other termination codes are also strings marked as an *ABORT* condition.

Using the termination code, it is possible to determine which event terminated the script. It is possible to decide if the string "BUSY" was received from the modem as opposed to "NO DIAL TONE". While the first event may be retried, the second will probably have little chance of succeeding during a retry.

COPYRIGHT

The *chat* program is in public domain. This is not the GNU public license. If it breaks then you get to keep both pieces.

NAME

chkconfig — rc.conf.d management utility

SYNOPSIS

chkconfig [*service*]
chkconfig [**−f**] *service* [*on* | *off*]

DESCRIPTION

The **chkconfig** utility allows a system administrator to easily manage services started by `rc`(8).

With no arguments, **chkconfig** will print the state (on or off) of every service found in the directory `/etc/rc.conf.d/`. The settings will be shown sorted according to the output of `rcorder`(8).

If the *service* is specified as the sole argument, **chkconfig** exits with status 0 if the service is ‘on’ and status 1 if ‘off’. The exit status may be used by shell scripts to test the state of a service, for example:

```
if chkconfig wscons; then
        echo "wscons is on"
else
        echo "wscons is off"
fi
```

The optional third argument allows the specified service to be set to ‘on’ or ‘off’ for the next time its `rc.d`(8) script is run (generally the next system reboot).

In order for **chkconfig** to display service status or to control service state, the service must be managed by **chkconfig**. A **chkconfig** managed service is one that has the `rcorder`(8) keyword “chkconfig” present in the `rc.d`(8) script for that service. If a service’s script does not contain this keyword, it may still be managed by **chkconfig** using the **−f** flag. When this flag is used, **chkconfig** will automatically add the “chkconfig” keyword to the service’s script.

SEE ALSO

`rc`(8), `rcorder`(8)

HISTORY

The **chkconfig** command first appeared in NetBSD 1.6.

AUTHORS

The **chkconfig** program was written by Dan Mercer ⟨dmercer@zembu.com⟩ of Zembu Labs, Inc.

BUGS

The **chkconfig** program currently does not support configuring the options to be passed to services upon startup.

NAME

chown — change file owner and group

SYNOPSIS

chown [**−R** [**−H** | **−L** | **−P**]] [**−fhv**] *owner*[*:group*] *file ...*
chown [**−R** [**−H** | **−L** | **−P**]] [**−fhv**] *:group file ...*

DESCRIPTION

chown sets the user ID and/or the group ID of the specified files.

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 user ID and/or the group ID for the file hierarchies rooted in the files instead of just the files themselves.
−f	Don't report any failure to change file owner or group, nor modify the exit status to reflect such failures.
−h	If *file* is a symbolic link, the owner and/or group of the link is changed.
−v	Cause **chown** to be verbose, showing files as they are processed.

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 **−L** option cannot be used together with the **−h** option.

The *owner* and *group* operands are both optional, however, one must be specified. If the *group* operand is specified, it must be preceded by a colon (“:”) character.

The *owner* may be either a user name or a numeric user ID. The *group* may be either a group name or a numeric group ID. Since it is valid to have a user or group name that is numeric (and doesn't have the numeric ID that matches its name) the name lookup is always done first. Preceding an ID with a “#” character will force it to be taken as a number.

The ownership of a file may only be altered by a super-user for obvious security reasons.

Unless invoked by the super-user, **chown** clears the set-user-id and set-group-id bits on a file to prevent accidental or mischievous creation of set-user-id and set-group-id programs.

The **chown** utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

Previous versions of the **chown** utility used the dot (“.”) character to distinguish the group name. This has been changed to be a colon (“:”) character so that user and group names may contain the dot character.

SEE ALSO

chflags(1), chgrp(1), find(1), chown(2), lchown(2), fts(3), symlink(7)

STANDARDS

The **chown** command is expected to be POSIX 1003.2 compliant.

The **-v** option and the use of ''#'' to force a numeric lookup are extensions to IEEE Std 1003.2 ("POSIX.2").

NAME

chroot — change root directory

SYNOPSIS

chroot [**−u** *user*] [**−g** *group*] [**−G** *group,group,...*] *newroot* [*command*]

DESCRIPTION

The **chroot** command changes its root directory to the supplied directory *newroot* and exec's *command*, if supplied, or an interactive copy of your shell.

If the **−u**, **−g** or **−G** options are given, the user, group and group list of the process are set to these values after the chroot has taken place. See setgid(2), setgroups(2), setuid(2), getgrnam(3) and getpwnam(3).

Note, *command* or the shell are run as your real-user-id.

ENVIRONMENT

The following environment variable is referenced by **chroot**:

SHELL If set, the string specified by SHELL is interpreted as the name of the shell to exec. If the variable SHELL is not set, /bin/sh is used.

SEE ALSO

chdir(2), chroot(2), environ(7)

HISTORY

The **chroot** utility first appeared in 4.4BSD.

SECURITY CONSIDERATIONS

chroot should never be installed setuid root, as it would then be possible to exploit the program to gain root privileges.

NAME

chrtbl — create character classification and upper <-> lower conversion tables

SYNOPSIS

chrtbl [**-o** *ofile*] *ifile*

DESCRIPTION

chrtbl creates character classification and upper <-> lower conversion tables for single byte files. The **chrtbl** command is modelled after the Solaris/SVR4 command. The input file is similar and contains a keyword per line followed by characters or ranges. Valid keywords are:

LC_CTYPE *filename*
: Set the filename for the character classification output.

LC_NUMERIC *filename*
: Set the filename for the numeric formatting output.

isupper *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be upper case.

islower *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be lower case.

isdigit *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be numeric.

isspace *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be space.

ispunct *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be punctuation.

iscntrl *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be control.

isxdigit *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be hexadecimal digits.

isblank *begin-char* [- *end-char*]
: Set the attribute of the specified characters range(s) to be blank.

ul *<upper-char lower-char> ...*
: Specify a case correspondence between upper and lower char.

cswidth *n1,s1:n2,s2:n3,s3*
: Specify the character set byte width (n1,n2,n3) and the screen width(s1,s2,s3) for the 3 character sets.

decimal_point *char*
: Specify the decimal point numeric formatting character.

thousands_sep *char*
: Specify the thousands separator numeric formatting character.

Available options

-o *ofile*
: Print the conversion tables in a human readable (C source) form.

SEE ALSO

`setlocale`(3)

BUGS

Preliminary support of LC_NUMERIC is present, but not currently fully implemented. No support for wide character locales. Support for alternate localized character sets and numeric formatting is currently not implemented.

NAME

cleanup − canonicalize and enqueue Postfix message

SYNOPSIS

cleanup [generic Postfix daemon options]

DESCRIPTION

The **cleanup**(8) daemon processes inbound mail, inserts it into the **incoming** mail queue, and informs the queue manager of its arrival.

The **cleanup**(8) daemon always performs the following transformations:

- Insert missing message headers: (**Resent-**) **From:**, **To:**, **Message-Id:**, and **Date:**.
- Transform envelope and header addresses to the standard *user@fully-qualified-domain* form that is expected by other Postfix programs. This task is delegated to the **trivial-rewrite**(8) daemon.
- Eliminate duplicate envelope recipient addresses.

The following address transformations are optional:

- Optionally, rewrite all envelope and header addresses according to the mappings specified in the **canonical**(5) lookup tables.
- Optionally, masquerade envelope sender addresses and message header addresses (i.e. strip host or domain information below all domains listed in the **masquerade_domains** parameter, except for user names listed in **masquerade_exceptions**). By default, address masquerading does not affect envelope recipients.
- Optionally, expand envelope recipients according to information found in the **virtual**(5) lookup tables.

The **cleanup**(8) daemon performs sanity checks on the content of each message. When it finds a problem, by default it returns a diagnostic status to the client, and leaves it up to the client to deal with the problem. Alternatively, the client can request the **cleanup**(8) daemon to bounce the message back to the sender in case of trouble.

STANDARDS

RFC 822 (ARPA Internet Text Messages)
RFC 2045 (MIME: Format of Internet Message Bodies)
RFC 2046 (MIME: Media Types)
RFC 3463 (Enhanced Status Codes)
RFC 3464 (Delivery status notifications)

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

Table-driven rewriting rules make it hard to express **if then else** and other logical relationships.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically, as **cleanup**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

COMPATIBILITY CONTROLS

undisclosed_recipients_header (To: undisclosed-recipients:;)

Message header that the Postfix **cleanup**(8) server inserts when a message contains no To: or Cc: message header.

Available in Postfix version 2.1 only:

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).

BUILT-IN CONTENT FILTERING CONTROLS

Postfix built-in content filtering is meant to stop a flood of worms or viruses. It is not a general content filter.

body_checks (empty)
Optional lookup tables for content inspection as specified in the **body_checks**(5) manual page.

header_checks (empty)
Optional lookup tables for content inspection of primary non-MIME message headers, as specified in the **header_checks**(5) manual page.

Available in Postfix version 2.0 and later:

body_checks_size_limit (51200)
How much text in a message body segment (or attachment, if you prefer to use that term) is subjected to body_checks inspection.

mime_header_checks ($header_checks)
Optional lookup tables for content inspection of MIME related message headers, as described in the **header_checks**(5) manual page.

nested_header_checks ($header_checks)
Optional lookup tables for content inspection of non-MIME message headers in attached messages, as described in the **header_checks**(5) manual page.

Available in Postfix version 2.3 and later:

message_reject_characters (empty)
The set of characters that Postfix will reject in message content.

message_strip_characters (empty)
The set of characters that Postfix will remove from message content.

BEFORE QUEUE MILTER CONTROLS

As of version 2.3, Postfix supports the Sendmail version 8 Milter (mail filter) protocol. When mail is not received via the smtpd(8) server, the cleanup(8) server will simulate SMTP events to the extent that this is possible. For details see the MILTER_README document.

non_smtpd_milters (empty)
A list of Milter (mail filter) applications for new mail that does not arrive via the Postfix **smtpd**(8) server.

milter_protocol (2)
The mail filter protocol version and optional protocol extensions for communication with a Milter (mail filter) application.

milter_default_action (tempfail)
The default action when a Milter (mail filter) application is unavailable or mis-configured.

milter_macro_daemon_name ($myhostname)
The {daemon_name} macro value for Milter (mail filter) applications.

milter_macro_v ($mail_name $mail_version)
The {v} macro value for Milter (mail filter) applications.

milter_connect_timeout (30s)
The time limit for connecting to a Milter (mail filter) application, and for negotiating protocol options.

milter_command_timeout (30s)
The time limit for sending an SMTP command to a Milter (mail filter) application, and for receiving the response.

milter_content_timeout (300s)
The time limit for sending message content to a Milter (mail filter) application, and for receiving the response.

milter_connect_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after completion of an SMTP connection.

milter_helo_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the SMTP HELO or EHLO command.

milter_mail_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the SMTP MAIL FROM command.

milter_rcpt_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the SMTP RCPT TO command.

milter_data_macros (see postconf -n output)
The macros that are sent to version 4 or higher Milter (mail filter) applications after the SMTP DATA command.

milter_unknown_command_macros (see postconf -n output)
The macros that are sent to version 3 or higher Milter (mail filter) applications after an unknown SMTP command.

milter_end_of_data_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the message end-of-data.

MIME PROCESSING CONTROLS

Available in Postfix version 2.0 and later:

disable_mime_input_processing (no)
Turn off MIME processing while receiving mail.

mime_boundary_length_limit (2048)
The maximal length of MIME multipart boundary strings.

mime_nesting_limit (100)
The maximal recursion level that the MIME processor will handle.

strict_8bitmime (no)
Enable both strict_7bit_headers and strict_8bitmime_body.

strict_7bit_headers (no)
Reject mail with 8-bit text in message headers.

strict_8bitmime_body (no)
Reject 8-bit message body text without 8-bit MIME content encoding information.

strict_mime_encoding_domain (no)
Reject mail with invalid Content-Transfer-Encoding: information for the message/* or multipart/* MIME content types.

AUTOMATIC BCC RECIPIENT CONTROLS

Postfix can automatically add BCC (blind carbon copy) when mail enters the mail system:

always_bcc (empty)
Optional address that receives a "blind carbon copy" of each message that is received by the Postfix mail system.

Available in Postfix version 2.1 and later:

sender_bcc_maps (empty)
Optional BCC (blind carbon-copy) address lookup tables, indexed by sender address.

recipient_bcc_maps (empty)
Optional BCC (blind carbon-copy) address lookup tables, indexed by recipient address.

ADDRESS TRANSFORMATION CONTROLS

Address rewriting is delegated to the **trivial-rewrite**(8) daemon. The **cleanup**(8) server implements table driven address mapping.

empty_address_recipient (MAILER-DAEMON)
The recipient of mail addressed to the null address.

canonical_maps (empty)
Optional address mapping lookup tables for message headers and envelopes.

recipient_canonical_maps (empty)
Optional address mapping lookup tables for envelope and header recipient addresses.

sender_canonical_maps (empty)
Optional address mapping lookup tables for envelope and header sender addresses.

masquerade_classes (envelope_sender, header_sender, header_recipient)
What addresses are subject to address masquerading.

masquerade_domains (empty)
Optional list of domains whose subdomain structure will be stripped off in email addresses.

masquerade_exceptions (empty)
Optional list of user names that are not subjected to address masquerading, even when their address matches $masquerade_domains.

propagate_unmatched_extensions (canonical, virtual)
What address lookup tables copy an address extension from the lookup key to the lookup result.

Available before Postfix version 2.0:

virtual_maps (empty)
Optional lookup tables with a) names of domains for which all addresses are aliased to addresses in other local or remote domains, and b) addresses that are aliased to addresses in other local or remote domains.

Available in Postfix version 2.0 and later:

virtual_alias_maps ($virtual_maps)
Optional lookup tables that alias specific mail addresses or domains to other local or remote address.

Available in Postfix version 2.2 and later:

canonical_classes (envelope_sender, envelope_recipient, header_sender, header_recipient)
What addresses are subject to canonical_maps address mapping.

recipient_canonical_classes (envelope_recipient, header_recipient)
What addresses are subject to recipient_canonical_maps address mapping.

sender_canonical_classes (envelope_sender, header_sender)
What addresses are subject to sender_canonical_maps address mapping.

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.

RESOURCE AND RATE CONTROLS

duplicate_filter_limit (1000)
The maximal number of addresses remembered by the address duplicate filter for **aliases**(5) or **virtual**(5) alias expansion, or for **showq**(8) queue displays.

header_size_limit (102400)
The maximal amount of memory in bytes for storing a message header.

hopcount_limit (50)
The maximal number of Received: message headers that is allowed in the primary message headers.

in_flow_delay (1s)
Time to pause before accepting a new message, when the message arrival rate exceeds the message delivery rate.

message_size_limit (10240000)
The maximal size in bytes of a message, including envelope information.

Available in Postfix version 2.0 and later:

header_address_token_limit (10240)
The maximal number of address tokens are allowed in an address message header.

mime_boundary_length_limit (2048)
The maximal length of MIME multipart boundary strings.

mime_nesting_limit (100)
The maximal recursion level that the MIME processor will handle.

queue_file_attribute_count_limit (100)
The maximal number of (name=value) attributes that may be stored in a Postfix queue file.

Available in Postfix version 2.1 and later:

virtual_alias_expansion_limit (1000)
The maximal number of addresses that virtual alias expansion produces from each original recipient.

virtual_alias_recursion_limit (1000)
The maximal nesting depth of virtual alias expansion.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

delay_warning_time (0h)
The time after which the sender receives the message headers of mail that is still queued.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

myhostname (see ’postconf -d’ output)
The internet hostname of this mail system.

myorigin ($myhostname)
The domain name that locally-posted mail appears to come from, and that locally posted mail is delivered to.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

queue_directory (see ’postconf -d’ output)
The location of the Postfix top-level queue directory.

soft_bounce (no)
Safety net to keep mail queued that would otherwise be returned to the sender.

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".

Available in Postfix version 2.1 and later:

enable_original_recipient (yes)
Enable support for the X-Original-To message header.

FILES

/etc/postfix/canonical*, canonical mapping table
/etc/postfix/virtual*, virtual mapping table

SEE ALSO

trivial-rewrite(8), address rewriting
qmgr(8), queue manager
header_checks(5), message header content inspection
body_checks(5), body parts content inspection
canonical(5), canonical address lookup table format
virtual(5), virtual alias lookup table format
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
ADDRESS_REWRITING_README Postfix address manipulation
CONTENT_INSPECTION_README content inspection

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
clri — clear an inode

SYNOPSIS
clri *special_device inode_number ...*

DESCRIPTION
clri is obsoleted for normal file system repair work by fsck(8).

clri zeros out the inodes with the specified inode number(s) on the filesystem residing on the given *special_device*. The fsck(8) utility is usually run after **clri** to reclaim the zero'ed inode(s) and the blocks previously claimed by those inode(s). Both read and write permission are required on the specified *special_device*.

The primary purpose of this routine is to remove a file which for some reason is not being properly handled by fsck(8). Once removed, it is anticipated that fsck(8) will be able to clean up the resulting mess.

SEE ALSO
fsck(8), fsdb(8)

BUGS
If the file is open, the work of **clri** will be lost when the inode is written back to disk from the inode cache.

NAME

cnwctl — display statistics and control Netwave AirSurfer PC Cards

SYNOPSIS

cnwctl [**-d** *domain*] [**-i** *interface*] [**-k** *scramble-key*] [**-sS** [*rate*]]

DESCRIPTION

The **cnwctl** utility is used to control Netwave AirSurfer PC Cards as well as display statistics. The following options are available:

-d Set the domain of the card to *domain*. The domain must be between 0x000 and 0x1ff. The domains 0x000 through 0x0ff are for access to an add-hoc network. The domains 0x100 through 0x1ff are for access to a Netwave Access Point. The default domain is 0x000. A card may only talk to the access point in its domain, or other cards in its add-hoc domain.

-i Use *interface* as the interface rather than `cnw0`.

-k Set the scramble key to *scramble-key*. The scramble key must be between 0x0000 and 0xffff. Both the source and the destination must use the same scramble key in order to communicate.

-s Display statistics. When the optional argument *rate* (which must be the last argument to the end of the command line) is specified as a non-zero value, statistics will be displayed every *rate* seconds. At the top of each "page" of statistics, column labels will be displayed. The first row of statistics will be totals since boot, subsequent lines are deltas from the previous row. If *rate* is not specified, or is 0 (zero), a single page of statistics will be displayed. These statistics are more detailed and include:

domain	The domain this card is part of.
rx	Number of packets received.
rxoverflow	Number of overflows detected.
rxoverrun	Number of overruns detected.
rxcrcerror	Number of CRC errors detected. Random noise can cause these errors.
rxframe	Number of framing errors detected.
rxerrors	Number of generic errors detected.
rxavail	Number of times a packet was available.
tx	Number of packets requested to be sent.
txokay	Number of packets sent.
txabort	Number of packets aborted (not sent within 9 tries).
txlostcd	Number of times carrier detect was lost.
txerrors	Number of generic transmit errors detected.
txretries	Total number of retries.
*N*x retries	Number of packets which were retried *N* times.

−S Display status read from the hardware. This option is only available to the super user. The *rate* argument is used as with the **−s** option. The following fields are displayed

link integrity field (lif)
A 0 value implies no links.

connection quality (cq)
Probably indicates the quality of the connection to the access point.

spu Unknown meaning.

link quality (lq)
Probably indicated the quality of the link to the access point.

hhc Unknown meaning.

mhs Unknown meaning.

revision The revision numbers of the card.

id The ID of the card.

SEE ALSO

cnw(4)

NAME

compat_30 — setup procedure for backward compatibility on post-3.0 release

SYNOPSIS

options COMPAT_30

DESCRIPTION

The **compat_30** module allows NetBSD to run NetBSD 3.0 executables.

The support is present if the kernel was built with option `COMPAT_30`. It is not available as a loadable module.

Static executables typically need no additional setup. Dynamic binaries may require shared libraries whose major version number changed since NetBSD 3.0, which are listed below. A shadow directory under `/emul` is not used; the libraries can be obtained from a NetBSD 3.0 distribution and installed in the original directories shown, as the major version number in the file name will prevent conflicts. If an upgrade installation from NetBSD 3.0 has been done and these libraries are still present, nothing more need be done.

Libraries needed from 3.0

`/lib/libcrypto.so.2.1` `/lib/libcrypto.so.2`

`/usr/lib/libcrypto.so.2.1` `/usr/lib/libcrypto.so.2`

`/lib/libevent.so.0.2` `/lib/libevent.so.0`

`/usr/lib/libevent.so.0.2` `/usr/lib/libevent.so.0`

`/usr/lib/libg2c.so.2.0` `/usr/lib/libg2c.so.2`

`/usr/lib/libkadm.so.5.0` `/usr/lib/libkadm.so.5`

`/usr/lib/libkafs.so.6.0` `/usr/lib/libkafs.so.6`

`/usr/lib/libkdb.so.5.0` `/usr/lib/libkdb.so.5`

`/usr/lib/libkrb5.so.19.1` `/usr/lib/libkrb5.so.19`

`/usr/lib/libkrb.so.6.0` `/usr/lib/libkrb.so.6`

`/usr/lib/libkstream.so.2.0` `/usr/lib/libkstream.so.2`

`/usr/lib/libmagic.so.0.1` `/usr/lib/libmagic.so.0`

`/usr/lib/libpcap.so.1.4` `/usr/lib/libpcap.so.1`

`/lib/libradius.so.0.0` `/lib/libradius.so.0`

`/usr/lib/libradius.so.0.0` `/usr/lib/libradius.so.0`

`/usr/lib/libssh.so.1.0` `/usr/lib/libssh.so.1`

`/usr/lib/libssl.so.3.0` `/usr/lib/libssl.so.3`

`/usr/lib/libstdc++.so.5.0` `/usr/lib/libstdc++.so.5`

`/lib/libz.so.0.4` `/lib/libz.so.0`

`/usr/lib/libz.so.0.4` `/usr/lib/libz.so.0`

`/usr/lib/libamu.so.2.1` `/usr/lib/libamu.so.2`

IMPLEMENTATION NOTES

COMPAT_30 enables the NetBSD 3.0 versions of the following system calls, whose syscall numbers and argument structures were changed after the 3.0 release to accommodate 64-bit filesystems: fhstat(2), fstat(2), getdents(2), lstat(2), stat(2).

The filehandle structure (formerly *fhandle_t*) was made opaque to userland and variable-sized. A *fh_size* argument was added to related syscalls: fhstat(2), fhstatvfs(2), fhstatvfs1(2), fhopen(2), getfh(2). This changes the API and ABI of those syscalls, COMPAT_30 enables binary compatibility with the old ABI. Source compatibility is not provided, as use of those syscalls is supposed to be rare.

The error code from the socket(2) syscall changed from EPROTONOSUPPORT to EAFNOSUPPORT in the case of an unsupported address family. COMPAT_30 enables binary compatibility with the old ABI. Source compatiblility is not provided.

The *struct ntptimeval* used by ntp_gettime(2) changed with the implementation of timecounters.

SEE ALSO

config(1), fhstat(2), fstat(2), getdents(2), lstat(2), stat(2), options(4)

HISTORY

NetBSD offers back-compatibility options back to NetBSD 0.9, but the first to be documented with a manual page is **compat_30**.

BUGS

The compatible getdents(2) is unable to see directory entries beneath the top layer of a union, even though the real 3.0 **getdents**() did not have that problem.

SECURITY CONSIDERATIONS

Programs with security impact that receive incorrect directory contents from **getdents**() may behave improperly, as when they are unable to find, or find the wrong versions of, important files.

NAME

compat_darwin — setup procedure for running Darwin binaries from MacOS X

DESCRIPTION

NetBSD supports running Darwin binaries. This works on PowerPC ports, and i386 should be supported in the future. For now, most text based and X11 based program should work. Applications using the Quartz displaying system may work, but local display is not yet supported: running MacOS X's Quartz display server on NetBSD is a work in progress.

The Darwin compatibility feature is active for kernels compiled with the `COMPAT_DARWIN`, `COMPAT_MACH`, and `EXEC_MACHO` options enabled.

All Darwin binaries are dynamically linked. As `COMPAT_DARWIN` only emulates the Darwin system calls, you will need various Darwin userland files, such as the shared libraries and the dynamic linker. Theses files are kept in a "shadow root" directory, named `/emul/darwin`. Each time a Darwin binary has to use a file, it will look it up in `/emul/darwin` first. This feature is used to prevent conflict between native and foreign libraries and configuration files.

There are two ways of setting up the `/emul/darwin` tree.

1. The easiest way is to install the Darwin library package in `pkgsrc/emulators/darwin_lib`. This package uses files provided by the OpenDarwin project. Therefore, it does not contain Apple's MacOS X proprietary libraries, which are required in order to run any Quartz-based application. If you need some files not installed by the package, read on.
2. You can also install Darwin or MacOS X files in `/emul/darwin` by hand. To do this, you need a Darwin system. In order to know what libraries a program needs, just use the
   ```
   otool -L program
   ```
   command on Darwin. Alternatively, you can use `ktrace`(1) and `kdump`(1) to discover what files the program attempts to open.

Please note that you need a valid MacOS X license if you copy Apple proprietary libraries and programs from a MacOS X system.

SEE ALSO

`kdump`(1), `ktrace`(1), `options`(4)

AUTHORS

`COMPAT_DARWIN` and `COMPAT_MACH` layers were written by Emmanuel Dreyfus ⟨manu@NetBSD.org⟩ with some help from
Christos Zoulas ⟨christos@NetBSD.org⟩

`EXEC_MACHO` was integrated into NetBSD by
Christos Zoulas ⟨christos@NetBSD.org⟩.

The `pkgsrc/emulators/darwin_lib` package was created by
James Whitwell ⟨abacau@yahoo.com.au⟩.

BUGS

Many. `COMPAT_DARWIN` is still very experimental.

NAME

compat_freebsd — setup procedure for running FreeBSD binaries

DESCRIPTION

NetBSD supports running FreeBSD binaries. Most binaries should work, except programs that use FreeBSD-specific features. These include i386-specific calls, such as syscons utilities. The FreeBSD compatibility feature is active for kernels compiled with the `COMPAT_FREEBSD` option enabled.

A lot of programs are dynamically linked. This means, that you will also need the FreeBSD shared libraries that the program depends on, and the runtime linker. Also, you will need to create a "shadow root" directory for FreeBSD binaries on your NetBSD system. This directory is named `/emul/freebsd`. Any file operations done by FreeBSD programs run under NetBSD will look in this directory first. So, if a FreeBSD program opens, for example, `/etc/passwd`, NetBSD will first try to open `/emul/freebsd/etc/passwd`, and if that does not exist open the 'real' `/etc/passwd` file. It is recommended that you install FreeBSD packages that include configuration files, etc under `/emul/freebsd`, to avoid naming conflicts with possible NetBSD counterparts. Shared libraries should also be installed in the shadow tree.

Generally, you will need to look for the shared libraries that FreeBSD binaries depend on only the first few times that you install a FreeBSD program on your NetBSD system. After a while, you will have a sufficient set of FreeBSD shared libraries on your system to be able to run newly imported FreeBSD binaries without any extra work.

Setting up shared libraries

How to get to know which shared libraries FreeBSD binaries need, and where to get them? Basically, there are 2 possibilities (when following these instructions: you will need to be root on your NetBSD system to do the necessary installation steps).

1. You have access to a FreeBSD system. In this case you can temporarily install the binary there, see what shared libraries it needs, and copy them to your NetBSD system. Example: you have just ftp-ed the FreeBSD binary of SimCity. Put it on the FreeBSD system you have access to, and check which shared libraries it needs by running 'ldd sim':

```
me@freebsd% ldd /usr/local/lib/SimCity/res/sim
    /usr/local/lib/SimCity/res/sim:
            -lXext.6 => /usr/X11R6/lib/libXext.so.6.0 (0x100c1000)
            -lX11.6 => /usr/X11R6/lib/libX11.so.6.0 (0x100c9000)
            -lc.2 => /usr/lib/libc.so.2.1 (0x10144000)
            -lm.2 => /usr/lib/libm.so.2.0 (0x101a7000)
            -lgcc.261 => /usr/lib/libgcc.so.261.0 (0x101bf000)
```

   You would need go get all the files from the last column, and put them under `/emul/freebsd`. This means you eventually have these files on your NetBSD system:

```
/emul/freebsd/usr/X11R6/lib/libXext.so.6.0
/emul/freebsd/usr/X11R6/lib/libX11.so.6.0
/emul/freebsd/usr/lib/libc.so.2.1
/emul/freebsd/usr/lib/libm.so.2.0
/emul/freebsd/usr/lib/libgcc.so.261.0
```

   Note that if you already have a FreeBSD shared library with a matching major revision number to the first column of the **ldd** output, you won't need to copy the file named in the last column to your system, the one you already have should work. It is advisable to copy the shared library anyway if it is a newer version, though. You can remove the old one. So, if you have these libraries on your system:

`/emul/freebsd/usr/lib/libc.so.2.0`

and you find that the ldd output for a new binary you want to install is:

−lc.2 => /usr/lib/libc.so.2.1 (0x10144000)

You won't need to worry about copying `/usr/lib/libc.so.2.1` too, because the program should work fine with the slightly older version. You can decide to replace the libc.so anyway, and that should leave you with:
`/emul/freebsd/usr/lib/libc.so.2.1`

Finally, you must make sure that you have the FreeBSD runtime linker and its config files on your system. You should copy these files from the FreeBSD system to their appropriate place on your NetBSD system (in the `/emul/freebsd` tree):
`usr/libexec/ld.so`
`var/run/ld.so.hints`

2. You don't have access to a FreeBSD system. In that case, you should get the extra files you need from various ftp sites. Information on where to look for the various files is appended below. For now, let's assume you know where to get the files.

Retrieve the following files (from _one_ ftp site to avoid any version mismatches), and install them under `/emul/freebsd` (i.e. `foo/bar` is installed as `/emul/freebsd/foo/bar`):
`sbin/ldconfig`
`usr/bin/ldd`
`usr/lib/libc.so.x.y.z`
`usr/libexec/ld.so`

ldconfig and **ldd** don't necessarily need to be under `/emul/freebsd`, you can install them elsewhere in the system too. Just make sure they don't conflict with their NetBSD counterparts. A good idea would be to install them in `/usr/local/bin` as **ldconfig-freebsd** and **ldd-freebsd**.

Run the FreeBSD ldconfig program with directory arguments in which the FreeBSD runtime linker should look for shared libs. `/usr/lib` are standard, you could run like the following:

```
me@netbsd% mkdir -p /emul/freebsd/var/run
me@netbsd% touch /emul/freebsd/var/run/ld.so.hints
me@netbsd% ldconfig-freebsd /usr/X11R6/lib /usr/local/lib
```

Note that argument directories of ldconfig are mapped to `/emul/freebsd/XXXX` by NetBSD's compat code, and should exist as such on your system. Make sure `/emul/freebsd/var/run/ld.so.hints` is existing when you run FreeBSD's ldconfig, if not, you may lose NetBSD's `/var/run/ld.so.hints`. FreeBSD **ldconfig** should be statically linked, so it doesn't need any shared libraries by itself. It will create the file `/emul/freebsd/var/run/ld.so.hints`. You should rerun the FreeBSD version of the ldconfig program each time you add a new shared library.

You should now be set up for FreeBSD binaries which only need a shared libc. You can test this by running the FreeBSD **ldd** on itself. Suppose that you have it installed as **ldd-freebsd**, it should produce something like:

```
me@netbsd% ldd-freebsd ‘which ldd-freebsd‘
        /usr/local/bin/ldd-freebsd:
                -lc.2 => /usr/lib/libc.so.2.1 (0x1001a000)
```

This being done, you are ready to install new FreeBSD binaries. Whenever you install a new FreeBSD program, you should check if it needs shared libraries, and if so, whether you have them installed in the `/emul/freebsd` tree. To do this, you run the FreeBSD version **ldd** on the new program, and watch its output. **ldd** (see also the manual page for `ldd`(1)) will print a list of shared libraries that the program depends on, in the form -l<majorname> => <fullname>.

If it prints "not found" instead of <fullname> it means that you need an extra library. Which library this is, is shown in <majorname>, which will be of the form XXXX.<N> You will need to find a libXXXX.so.<N>.<mm> on a FreeBSD ftp site, and install it on your system. The XXXX (name) and <N> (major revision number) should match; the minor number(s) <mm> are less important, though it is advised to take the most recent version.

3. In some cases, FreeBSD binary needs access to certain device file. For example, FreeBSD X server software needs FreeBSD `/dev/ttyv0` for ioctls. In this case, create a symbolic link from `/emul/freebsd/dev/ttyv0` to a `wscons`(4) device file like `/dev/ttyE0`. You will need to have at least **`options WSDISPLAY_COMPAT_SYSCONS`** and probably also **`options WSDISPLAY_COMPAT_USL`** in your kernel (see `options`(4) and `wscons`(4)).

Finding the necessary files

Note: the information below is valid as of the time this document was written (June, 1995), but certain details such as names of ftp sites, directories and distribution names may have changed by the time you read this.

The FreeBSD distribution is available on a lot of ftp sites. Sometimes the files are unpacked, and you can get the individual files you need, but mostly they are stored in distribution sets, usually consisting of subdirectories with gzipped tar files in them. The primary ftp sites for the distributions are:

```
ftp.freebsd.org:/pub/FreeBSD
```

Mirror sites are described on:

```
ftp.freebsd.org:/pub/FreeBSD/MIRROR.SITES
```

This distribution consists of a number of tar-ed and gzipped files, Normally, they're controlled by an install program, but you can retrieve files "by hand" too. The way to look something up is to retrieve all the files in the distribution, and ''tar ztvf'' through them for the file you need. Here is an example of a list of files that you might need.

```
Needed                 Files

ld.so                  2.0-RELEASE/bindist/bindist.??
ldconfig               2.0-RELEASE/bindist/bindist.??
ldd                    2.0-RELEASE/bindist/bindist.??
libc.so.2              2.0-RELEASE/bindist/bindist.??
libX11.so.6.0          2.0-RELEASE/XFree86-3.1/XFree86-3.1-bin.tar.gz
libX11.so.6.0          XFree86-3.1.1/X311bin.tgz
libXt.so.6.0           2.0-RELEASE/XFree86-3.1/XFree86-3.1-bin.tar.gz
libXt.so.6.0           XFree86-3.1.1/X311bin.tgz
```

The files called "bindist.??" are tar-ed, gzipped and split, so you can extract contents by "cat bindist.?? | tar zpxf -".

Extract the files from these gzipped tarfiles in your `/emul/freebsd` directory (possibly omitting or afterwards removing files you don't need), and you are done.

BUGS

The information about FreeBSD distributions may become outdated.

NAME

compat_ibcs2 — setup procedure for running iBCS2 binaries

DESCRIPTION

NetBSD supports running Intel Binary Compatibility Standard 2 (iBCS2) binaries. This only applies to i386 systems for now. Binaries are supported from SCO UNIX and other systems derived from AT&T System V.3 UNIX. iBCS2 support is only well tested using SCO binaries. XENIX binaries are also supported although not as well tested. SVR4 binaries are supported by the `COMPAT_SVR4` option.

iBCS2 supports COFF, ELF, and x.out (XENIX) binary formats. Binaries from SCO OpenServer (version 5.x) are the only ELF binaries that have been tested. Most programs should work, but not ones that use or depend on:

kernel internal data structures
STREAMS drivers (other than TCP/IP sockets)
local X displays (uses a STREAMS pipe)
virtual 8086 mode

The iBCS2 compatibility feature is active for kernels compiled with the `COMPAT_IBCS2` option enabled. If support for iBCS2 ELF executables is desired, the `EXEC_ELF32` option should be enabled in addition to `COMPAT_IBCS2`.

Many COFF-format programs and most ELF-format programs are dynamically linked. This means that you will also need the shared libraries that the program depends on. Also, you will need to create a “shadow root” directory for iBCS2 binaries on your NetBSD system. This directory is named `/emul/ibcs2`. Any file operations done by iBCS2 programs run under NetBSD will look in this directory first. So, if an iBCS2 program opens, for example, `/etc/passwd`, NetBSD will first try to open `/emul/ibcs2/etc/passwd`, and if that does not exist open the ‘real’ `/etc/passwd` file. It is recommended that you install iBCS2 packages that include configuration files, etc. under `/emul/ibcs2`, to avoid naming conflicts with possible NetBSD counterparts. Shared libraries should also be installed in the shadow tree.

Generally, you will need to look for the shared libraries that iBCS2 binaries depend on only the first few times that you install an iBCS2 program on your NetBSD system. After a while, you will have a sufficient set of iBCS2 shared libraries on your system to be able to run newly imported iBCS2 binaries without any extra work.

Setting up shared libraries

How to get to know which shared libraries iBCS2 binaries need, and where to get them? Depending on the file type of the executable, there are different possibilities (when following these instructions: you will need to be root on your NetBSD system to do the necessary installation steps).

COFF binaries You can simply copy all of the available shared libraries since they are fairly small in size. The COFF shared libraries are typically found in /shlib and can be obtained from the following sources:

SCO UNIX version 3.x (aka ODT)
SCO UNIX version 5.x (aka OpenServer)
SCO UnixWare
Many versions of SVR4.2/x86

After copying the shared libraries, you should have at least the following files on your system:

```
/emul/ibcs2/shlib/libc_s
/emul/ibcs2/shlib/libnsl_s
/emul/ibcs2/shlib/protlib_s
```

ELF binaries	You can simply copy all of the available shared libraries from the source system or distribution or use `ldd`(1) to determine the libraries required by a specific binary.

After copying the shared libraries, you should have at least the following files on your system:

```
/emul/ibcs2/usr/lib/libc.so.1
/emul/ibcs2/usr/lib/libcrypt.so
/emul/ibcs2/usr/lib/libndbm.so
/emul/ibcs2/usr/lib/libsocket.so.1
```

If you don't have access to a SCO system, you will need to get the extra files you need from a SCO distribution. As of January 1998, SCO sells a copy of SCO OpenServer (iBCS2) and/or SCO UnixWare (SVR4) for personal/non-commercial use for only the cost of shipping (about $20US). The distribution comes on an ISO9660-format CDROM which can be mounted and used to copy the necessary files.

Run the following script to copy the basic set of files from a SCO distribution directory mounted somewhere locally:

/usr/share/examples/emul/ibcs2/ibcs2-setup [directory]

You should now be set up for SCO binaries which only need standard shared libs.

BUGS

The information about SCO distributions may become outdated.

Attempting to a use a nameserver on the local host does not currently work due to an absurd shortcut taken by the iBCS2 network code (remember that there are no kernel sockets).

16/32/64 bit offsets may not be handled correctly in all cases.

NAME

compat_linux — setup procedure for running Linux binaries

DESCRIPTION

NetBSD supports running Linux binaries. This applies to amd64, arm, alpha, i386, m68k, and powerpc systems for now. Both the a.out and ELF binary formats are supported. Most programs should work, including the ones that use the Linux SVGAlib (only on i386). NetBSD amd64 can execute both 32bit and 64bit linux programs. Programs that will not work include some that use i386-specific calls, such as enabling virtual 8086 mode. Currently, sound is only partially supported for Linux binaries (they will probably run, depending on what Linux sound support features are used).

The Linux compatibility feature is active for kernels compiled with the `COMPAT_LINUX` option enabled. If support for Linux a.out executables is desired, the `EXEC_AOUT` option should be enabled in addition to option `COMPAT_LINUX`. Similarly, if support for Linux 32-bit and/or 64-bit ELF executables is desired, the `EXEC_ELF32` and/or `EXEC_ELF64` options (respectively) should be enabled in addition to `COMPAT_LINUX`.

A lot of programs are dynamically linked. This means that you will also need the Linux shared libraries that the program depends on, and the runtime linker. Also, you will need to create a "shadow root" directory for Linux binaries on your NetBSD system. This directory is named `/emul/linux` or `/emul/linux32` for 32bit emulation on 64bit systems. Any file operations done by Linux programs run under NetBSD will look in this directory first. So, if a Linux program opens, for example, `/etc/passwd`, NetBSD will first try to open `/emul/linux/etc/passwd`, and if that does not exist open the 'real' `/etc/passwd` file. It is recommended that you install Linux packages that include configuration files, etc under `/emul/linux`, to avoid naming conflicts with possible NetBSD counterparts. Shared libraries should also be installed in the shadow tree. Filenames that start "/../" are only looked up in the real root.

Generally, you will need to look for the shared libraries that Linux binaries depend on only the first few times that you install a Linux program on your NetBSD system. After a while, you will have a sufficient set of Linux shared libraries on your system to be able to run newly imported Linux binaries without any extra work.

Setting up shared libraries

How to get to know which shared libraries Linux binaries need, and where to get them? Basically, there are 2 possibilities (when following these instructions: you will need to be root on your NetBSD system to do the necessary installation steps).

1. For i386, you can simply install the SuSE shared libs using the `pkgsrc/emulators/suse100_linux` package(s). On PowerPC ports, the `pkgsrc/emulators/linuxppc_lib` will install the needed libraries. If you are on other platforms, or this doesn't supply you with all the needed libraries, read on.

2. You have access to a Linux system. In this case you can temporarily install the binary there, see what shared libraries it needs, and copy them to your NetBSD system. Example: you have just ftp-ed the Linux binary of Doom. Put it on the Linux system you have access to, and check which shared libraries it needs by running 'ldd linuxxdoom':

```
(me@linux) ldd linuxxdoom
        libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
        libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
        libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
```

   You would need go get all the files from the last column, and put them under `/emul/linux`, with the names in the first column as symbolic links pointing to them. This means you eventually have these files on your NetBSD system:

`/emul/linux/usr/X11/lib/libXt.so.3.1.0`
`/emul/linux/usr/X11/lib/libXt.so.3` (symbolic link to the above)
`/emul/linux/usr/X11/lib/libX11.so.3.1.0`
`/emul/linux/usr/X11/lib/libX11.so.3` (symbolic link to the above)
`/emul/linux/lib/libc.so.4.6.29`
`/emul/linux/lib/libc.so.4` (symbolic link to the above)

Note that if you already have a Linux shared library with a matching major revision number to the first column of the `ldd`(1) output, you won't need to copy the file named in the last column to your system, the one you already have should work. It is advisable to copy the shared library anyway if it is a newer version, though. You can remove the old one, as long as you make the symbolic link point to the new one. So, if you have these libraries on your system:

`/emul/linux/lib/libc.so.4.6.27`
`/emul/linux/lib/libc.so.4` -> `/emul/linux/lib/libc.so.4.6.27`

and you find that the **ldd** output for a new binary you want to install is:

libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29

you won't need to worry about copying `/lib/libc.so.4.6.29` too, because the program should work fine with the slightly older version. You can decide to replace the libc.so anyway, and that should leave you with:
`/emul/linux/lib/libc.so.4.6.29`
`/emul/linux/lib/libc.so.4` -> `/emul/linux/lib/libc.so.4.6.29`

Please note that the symbolic link mechanism is *only* needed for Linux binaries, the NetBSD runtime linker takes care of looking for matching major revision numbers itself, you don't need to worry about that.

Finally, you must make sure that you have the Linux runtime linker and its config files on your system. You should copy these files from the Linux system to their appropriate place on your NetBSD system (in the `/emul/linux` tree):
`/lib/ld.so`
`/etc/ld.so.cache`
`/etc/ld.so.config`

3. You don't have access to a Linux system. In that case, you should get the extra files you need from various ftp sites. Information on where to look for the various files is appended below. For now, let's assume you know where to get the files.

   Retrieve the following files (from _one_ ftp site to avoid any version mismatches), and install them under `/emul/linux` (i.e. `/foo/bar` is installed as `/emul/linux/foo/bar`):

   `/sbin/ldconfig`
   `/usr/bin/ldd`
   `/lib/libc.so.x.y.z`
   `/lib/ld.so`

   ldconfig and **ldd** don't necessarily need to be under `/emul/linux`, you can install them elsewhere in the system too. Just make sure they don't conflict with their NetBSD counterparts. A good idea would be to install them in `/usr/local/bin` as **ldconfig-linux** and **ldd-linux**.

   Create the file `/emul/linux/etc/ld.so.conf`, containing the directories in which the Linux runtime linker should look for shared libs. It is a plain text file, containing a directory name on each line. `/lib` and `/usr/lib` are standard, you could add the following:

```
/usr/X11/lib
/usr/local/lib
```

Note that these are mapped to `/emul/linux/XXXX` by NetBSD's compat code, and should exist as such on your system.

Run the Linux **`ldconfig`** program. It should be statically linked, so it doesn't need any shared libraries by itself. It will create the file `/emul/linux/etc/ld.so.cache` You should rerun the Linux version of **`ldconfig`** each time you add a new shared library.

You should now be set up for Linux binaries which only need a shared libc. You can test this by running the Linux **`ldd`** on itself. Suppose that you have it installed as **`ldd-linux`**, it should produce something like:

```
(me@netbsd) ldd-linux ‘which ldd-linux‘
        libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
```

This being done, you are ready to install new Linux binaries. Whenever you install a new Linux program, you should check if it needs shared libraries, and if so, whether you have them installed in the `/emul/linux` tree. To do this, you run the Linux **`ldd`** on the new program, and watch its output. **`ldd`** (see also the manual page for `ldd`(1)) will print a list of shared libraries that the program depends on, in the form ⟨majorname⟩ (⟨jumpversion⟩) => ⟨fullname⟩.

If it prints "not found" instead of ⟨fullname⟩ it means that you need an extra library. Which library this is, is shown in ⟨majorname⟩, which will be of the form libXXXX.so.<N> You will need to find a libXXXX.so.<N>.<mm> on a Linux ftp site, and install it on your system. The XXXX (name) and ⟨N⟩ (major revision number) should match; the minor number(s) ⟨mm⟩ are less important, though it is advised to take the most recent version.

4. Set up linux specific devices:

```
(me@netbsd) cd /usr/share/examples/emul/linux/etc
(me@netbsd) cp LINUX_MAKEDEV /emul/linux/dev
(me@netbsd) cd /emul/linux/dev && sh LINUX_MAKEDEV all
```

Setting up procfs

Some Linux binaries expect procfs to be mounted and that it would contain some Linux specific stuff. If it's not the case, they behave unexpectedly or even crash.

Mount procfs on NetBSD using following command:

```
(me@netbsd) mount_procfs -o linux procfs /emul/linux/proc
```

You can also set up your system so that procfs is mounted automatically on system boot, by putting an entry like the one below to `/etc/fstab`.

```
procfs /emul/linux/proc procfs ro,linux
```

See `mount_procfs`(8) for further information.

Setting up other files

Newer version of Linux use `/etc/nsswitch.conf` for network information, such as NIS and DNS. You must create or get a valid copy of this file and put it in `/emul/linux/etc`.

Finding the necessary files

Note: the information below is valid as of the time this document was first written (March, 1995), but certain details such as names of ftp sites, directories and distribution names may have changed by the time you read this.

Linux is distributed by several groups that make their own set of binaries that they distribute. Each distribution has its own name, like "Slackware" or "Yggdrasil". The distributions are available on a lot of ftp sites. Sometimes the files are unpacked, and you can get the individual files you need, but mostly they are stored in distribution sets, usually consisting of subdirectories with gzipped tar files in them. The primary ftp sites for the distributions are:

```
sunsite.unc.edu:/pub/Linux/distributions
tsx-11.mit.edu:/pub/linux/distributions
```

Some European mirrors:

```
ftp.luth.se:/pub/linux/distributions
ftp.demon.co.uk:/pub/linux/distributions
src.doc.ic.ac.uk:/packages/linux/distributions
```

For simplicity, let's concentrate on Slackware here. This distribution consists of a number of subdirectories, containing separate packages. Normally, they're controlled by an install program, but you can retrieve files "by hand" too. First of all, you will need to look in the `contents` subdir of the distribution. You will find a lot of small textfiles here describing the contents of the separate packages. The fastest way to look something up is to retrieve all the files in the contents subdirectory, and grep through them for the file you need. Here is an example of a list of files that you might need, and in which contents-file you will find it by grepping through them:

```
Needed                  Package

ld.so                   ldso
ldconfig                ldso
ldd                     ldso
libc.so.4               shlibs
libX11.so.6.0           xf_lib
libXt.so.6.0            xf_lib
libX11.so.3             oldlibs
libXt.so.3              oldlibs
```

So, in this case, you will need the packages ldso, shlibs, xf_lib and oldlibs. In each of the contents-files for these packages, look for a line saying "PACKAGE LOCATION", it will tell you on which 'disk' the package is, in our case it will tell us in which subdirectory we need to look. For our example, we would find the following locations:

```
Package                 Location

ldso                    diska2
shlibs                  diska2
oldlibs                 diskx6
xf_lib                  diskx9
```

The locations called `diskXX` refer to the `slakware/XX` subdirectories of the distribution, others may be found in the `contrib` subdirectory. In this case, we could now retrieve the packages we need by retrieving the following files (relative to the root of the Slackware distribution tree):

```
slakware/a2/ldso.tgz
slakware/a2/shlibs.tgz
slakware/x6/oldlibs/tgz
slakware/x9/xf_lib.tgz
```

Extract the files from these gzipped tarfiles in your /emul/linux directory (possibly omitting or afterwards removing files you don't need), and you are done.

Programs using SVGAlib

SVGAlib binaries require some extra care. You need to have **options WSDISPLAY_COMPAT_USL** in your kernel (see wscons(4)), and you will also have to create some symbolic links in the /emul/linux/dev directory, namely:
/emul/linux/dev/console -> /dev/tty
/emul/linux/dev/mouse -> whatever device your mouse is connected to
/emul/linux/dev/ttyS0 -> /dev/tty00
/emul/linux/dev/ttyS1 -> /dev/tty01

Be warned: the first link mentioned here makes SVGAlib binaries work, but may confuse others, so you may have to remove it again at some point.

BUGS

The information about Linux distributions may become outdated.

Absolute pathnames pointed to by symbolic links are only looked up in the shadow root when the symbolic link itself was found by an absolute pathname inside the shadow root. This is not consistent.

Linux executables cannot handle directory offset cookies > 32 bits. Should such an offset occur, you will see the message "linux_getdents: dir offset too large for emulated program". Currently, this can only happen on NFS mounted file systems, mounted from servers that return offsets with information in the upper 32 bits. These errors should rarely happen, but can be avoided by mounting this file system with offset translation enabled. See the **−X** option to mount_nfs(8). The **−2** option to mount_nfs(8) will also have the desired effect, but is less preferable.

NAME

compat_netbsd32 — setup procedure for 32-bit compatibility on 64-bit platform

DESCRIPTION

The **compat_netbsd32** module allows NetBSD/sparc64 to run NetBSD/sparc executables, and NetBSD/amd64 to run NetBSD/i386 executables.

To use **compat_netbsd32**, one must either have `COMPAT_NETBSD32` and `EXEC_ELF32` in the kernel, or load the compat_netbsd32 and exec_netbsd32 kernel modules.

Static executables typically need no additional setup. Dynamic binaries require the dynamic linker plus shared libraries. Most of these files will need to be placed under `/emul/netbsd32`.

The easiest method of installing support for these is via the `emulators/netbsd32_compat14`, `emulators/netbsd32_compat15`, and `emulators/netbsd32_compat16` packages, provided in the NetBSD packages collection. These install 32-bit a.out and ELF compatibility libraries, respectively. The details of what is actually necessary for correct operation are given below. This obviously is handled by the emulator packages.

For a.out compatibility, `/usr/libexec/ld.so` from a 32-bit distribution is required to exist as `/emul/netbsd32/usr/libexec/ld.so`. For 32-bit ELF compatibility, `/usr/libexec/ld.elf_so` needs to be in `/emul/netbsd32/usr/libexec/ld.elf_so`.

The shared libraries for a.out binaries do not live under the `/emul/netbsd32` directory, but under the `/emul/aout` directory, where the a.out dynamic linker will find them.

BUGS

A list of things which fail to work in compatibility mode should be here.

IPC is not well supported.

`sysctl`(3) is not well supported.

NAME

compat_osf1 — setup procedure for running OSF/1 binaries

DESCRIPTION

NetBSD supports running OSF/1 (a.k.a Digital Unix, a.k.a. Tru64) binaries on NetBSD/alpha systems. Most programs should work, including the ones that use the shared object libraries. Programs that make direct MACH system calls will not work. The OSF/1 compatibility feature is active for kernels compiled with the COMPAT_OSF1 option enabled (see options(4)).

To run dynamically linked programs, you will need the OSF/1 shared libraries, runtime linker, and certain configuration files found in /etc. These are installed in a "shadow root" directory called /emul/osf1. Any file operations done by OSF/1 programs run under NetBSD will look in this directory first, and fall back to the file system proper. So, if an OSF/1 program opens /etc/svc.conf, NetBSD will first try to open /emul/osf1/etc/svc.conf, and if that file does not exist it will then try /etc/svc.conf. Shared libraries and configuration specific to OSF/1 should be installed in the shadow tree.

Setting up /emul/osf1

The simple technique is to install pkgsrc/emulators/osf1_lib. (You may also want to install pkgsrc/www/navigator and/or pkgsrc/www/communicator.)

Alternatively, if you have access to an OSF/1 machine and if the licensing details permit, you can copy the contents of:

```
/shlib
/usr/shlib
/etc/sia
/usr/lib/X11/locale
```

(The latter is required to run Netscape Navigator or Communicator.)

Also copy

```
/etc/svc.conf
/usr/ccs/lib/cmplrs/otabase/libots.so
/sbin/loader
```

Or, simply NFS mount the appropriate directories under /emul/osf1.

SEE ALSO

config(1), options(4)

BUGS

Your hostname(1) *must* contain a dot *or* your resolv.conf(5) must contain a search line. Without one of those, the OSF/1 resolver will die and no hostname resolution will be possible.

Certain values in /emul/osf1/etc/svc.conf can cause programs to fail with "Bad system call".

Pathnames pointed to by symbolic links are not looked up in the shadow root when running an OSF/1 executable. This is not consistent.

NAME

compat_pecoff — setup procedure for running Win32 applications (a.k.a. PEACE)

DESCRIPTION

NetBSD has partial support for running Win32 applications. This manual page describes how to run Win32 (and hopefully WinCE in the future) applications on NetBSD. Note that PE (Portable Executable) is a Microsoft extension to the COFF executable file format.

BRIEF INTRODUCTION TO THE WIN32 API

The Win32 API is an application program interface (API) for 32-bit applications for Microsoft Windows 9x/Me/NT/2000. The Win32 API is provided via a set of core DLLs (Dynamically Linked Libraries), including `KERNEL32.DLL`, `USER32.DLL` and `GDI32.DLL`.

The structure of these core DLLs and the interface between the operating system kernel and userland is implementation-dependent. Each implementation must provide its own core DLLs. Therefore, these DLLs are different for Windows 98 and Windows 2000.

`KERNEL32.DLL` is used by all Win32 applications; it provides basic kernel interface such as file access, process control, memory management etc.

`USER32.DLL` is used by most Win32 applications; it provides basic userland functions such as GUI and messaging.

`GDI32.DLL` provides functions to draw images and characters.

`SHELL32.DLL` is the Windows shell support, including file association.

`COMCTL32.DLL` and `COMDLG32.DLL` are GUI components which are commonly used in many applications. `WSOCK32.DLL` provides the networking API. `DDRAW.DLL`, `DSOUND.DLL`, and `DINPUT.DLL` are for DirectX.

Most other DLLs are compatible among all the implementations and therefore can be shared.

NETBSD SUPPORT FOR THE WIN32 API

NetBSD support for Win32 applications is developed by the PEACE Project, and is under active development. Currently it can run some console applications including the Windows 2000 `CMD.EXE` as well as a small number of GUI applications.

The PEACE system consists of three parts: the kernel part, the dynamic loader and the core DLLs.

The kernel part provides loading and executing PE/COFF format executable binaries; i.e. it extends the `execve`(2) system call, just like other binary compatibility options. It is activated by enabling the `COMPAT_PECOFF` kernel option (see `options`(4)), or enabling `/usr/lkm/compat_pecoff.o` and `/usr/lkm/exec_pecoff.o` with `modload`(8).

The dynamic loader is the PE/COFF version of `ld.so`(1). It reads the file header of the executable binary, and loads required DLLs.

The core DLLs implement the actual Win32 API functions as described in the previous section. Since the kernel part does not provide any additional system calls and other kernel interface, all Win32 API functions are implemented on top of the existing NetBSD APIs (system calls and standard libraries such as `libc` and `libX11`).

PREPARING THE PEACE DYNAMIC LOADER AND CORE DLLS

Development snapshots of the dynamic loader can be retrieved from `http://sourceforge.net/project/showfiles.php?group_id=21711`. The file name of snapshot is `peace-i386-ld.so.dll-*.gz`, where '∗' is replaced with the snapshot date. Simply gunzip(1) the file and copy the resulting file to /usr/libexec/ld.so.dll.

The core DLLs archives can also be retrieved from http://sourceforge.net/project/showfiles.php?group_id=21711 as peace-i386-sysdll-*.tgz and peace-i386-dll-*.tgz. The dynamic loader searches for required DLLs from the following directories:

1. directories listed in the environment variable DLLPATH (separated by colons)
2. /usr/lib
3. the directory where the executable is located

The core DLLs are required to be installed into /usr/lib, in order to use CMD.EXE (or another Win32 application) as the login shell.

According to the development phase, some other PEACE-specific DLLs might be distributed separately. Please check the announcements on the Web or the mailing list.

Other DLLs can be stored in arbitrary directories specified by the environment variable DLLPATH. To use Windows NT/2000 DLLs installed on a separate partition of the local disk directly for NetBSD, type:

```
mount -t ntfs -o ro /dev/wd0h /nthd
setenv DLLPATH /nthd/WINNT/SYSTEM32:/nthd/WINNT
```

(assuming csh(1)).

SEE ALSO

config(1), gunzip(1), ld.so(1), execve(2), options(4), modload(8), mount_ntfs(8), http://chiharu.hauN.org/peace/

HISTORY

Kernel support for PE/COFF appeared in NetBSD 1.5.

AUTHORS

Implementation of Win32 binary compatibility support for NetBSD was started by Masaru OKI. The PEACE Project is founded by him to implement the enormous number of functions in the Win32/WinCE API.

BUGS

- Currently only the i386 platform is supported.
- Most functions in Win32 are missing.
- The dynamic loader and core DLLs are not provided in the standard distribution of NetBSD. This is because a cross-compiler is required to build them.

NAME

compat_sunos — setup procedure for m68k, sparc and sparc64 architectures

DESCRIPTION

NetBSD/sparc64, NetBSD/sparc and some of the NetBSD/m68k architectures can run SunOS executables. Most executables will work.

The exceptions include programs that use the SunOS kvm library, and various system calls, **ioctl**()'s, or kernel semantics that are difficult to emulate. The number of reasons why a program might fail to work is (thankfully) longer than the number of programs that fail to run.

Static executables will normally run without any extra setup. This procedure details the directories and files that must be set up to allow dynamically linked executables to work.

The files you need are on your SunOS machine. You need to worry about the legal issues of ensuring that you have a right to use the required files on your machine. On your NetBSD machine, do the following:

1. `mkdir -p /emul/sunos/usr/lib /emul/sunos/usr/5lib`
2. `cp SunOS:/usr/lib/lib*.so.*.* NetBSD:/emul/sunos/usr/lib`
3. `cp SunOS:/usr/5lib/lib*.so.*.* NetBSD:/emul/sunos/usr/5lib`
4. `cp SunOS:/usr/lib/ld.so NetBSD:/emul/sunos/usr/lib/ld.so`
5. If you ever expect to use YP, you will want to create a link:
   `ln -s /var/run/ypbind.lock /etc/ypbind.lock`

Alternatively, you can use an NFS mount to accomplish the same effect. On your NetBSD machine, do the following:

1. `mkdir -p /emul/sunos/usr`
2. `mount SunOS:/usr /emul/sunos/usr`

This will place the SunOS libraries on your NetBSD machine in a location where the SunOS compatibility code will look for first, where they do not conflict with the standard libraries.

NOTES

When using **compat_sunos** on NetBSD/sparc64, the `COMPAT_NETBSD32` option must also be used.

BUGS

A list of things which fail to work in compatibility mode should be here.

SunOS executables can not handle directory offset cookies > 32 bits. Should such an offset occur, you will see the message "sunos_getdents: dir offset too large for emulated program". Currently, this can only happen on NFS mounted filesystems, mounted from servers that return offsets with information in the upper 32 bits. These errors should rarely happen, but can be avoided by mounting this filesystem with offset translation enabled. See the **−X** option to `mount_nfs`(8). The **−2** option to `mount_nfs`(8) will also have the desired effect, but is less preferable.

The NetBSD/sparc64 support is less complete than the other ports.

NAME

compat_svr4 — setup procedure for running SVR4/iBCS2 binaries **compat_svr4_32** — setup procedure for running 32-bit SVR4/iBCS2 binaries

DESCRIPTION

NetBSD supports running SVR4/iBCS2 binaries. This code has been tested on i386 (with binaries from SCO OpenServer and XENIX), m68k (with binaries from AMIX) and sparc (with binaries from Solaris) systems. Most programs should work, but not ones that use or depend on:

kernel internal data structures
the `/proc` filesystem
the ticotsord loopback RPC mechanism (NIS uses this)
sound and video interfaces
threads (ttsession uses threads)
the streams administrative driver

The SVR4 compatibility feature is active for kernels compiled with the `COMPAT_SVR4` option enabled. Since support for ELF executables is included only if the kernel is compiled with the `EXEC_ELF32` or `EXEC_ELF64` options enabled, kernels which include `COMPAT_SVR4` should also typically include `EXEC_ELF32` (for 32-bit ELF support) and/or `EXEC_ELF64` (for 64-bit ELF support).

Another compatibility feature is `COMPAT_SVR4_32`, which allows the execution of 32-bit SVR4 binaries on a machine with a 64-bit kernel. This requires `EXEC_ELF32` and `COMPAT_NETBSD32` options as well as `COMPAT_SVR4`. It is configured the same way as `COMPAT_SVR4` but uses the `/emul/svr4_32` directory instead of `/emul/svr4`. But typically, `/emul/svr4_32` can be made to point to `/emul/svr4` if the operating system donating the libraries has support for both 32-bit and 64-bit binaries.

Execution of 32-bit SVR4 binaries on a machine with a 32-bit kernel uses `COMPAT_SVR4`, not `COMPAT_SVR4_32`.

Most SVR4 programs are dynamically linked. This means that you will also need the shared libraries that the program depends on and the runtime linker. Also, you will need to create a "shadow root" directory for SVR4 binaries on your NetBSD system. This directory is named `/emul/svr4`. Any file operations done by SVR4 programs run under NetBSD will look in this directory first. So, if a SVR4 program opens, for example, `/etc/passwd`, NetBSD will first try to open `/emul/svr4/etc/passwd`, and if that does not exist open the 'real' `/etc/passwd` file. It is recommended that you install SVR4 packages that include configuration files, etc under `/emul/svr4`, to avoid naming conflicts with possible NetBSD counterparts. Shared libraries should also be installed in the shadow tree.

The simplest way to set up your system for SVR4 binaries is:

1. Make the necessary directories:

```
(me@netbsd) mkdir -p /emul/svr4/{dev,etc}
(me@netbsd) mkdir -p /emul/svr4/usr/{bin,lib,ucblib}
(me@netbsd) mkdir -p /emul/svr4/usr/openwin/{bin,lib}
(me@netbsd) mkdir -p /emul/svr4/usr/dt/{bin,lib}
```

2. Copy files from an svr4 system:

```
(me@svr4) cd /usr/lib
(me@svr4) tar -cf - . | \
            rsh netbsd 'cd /emul/svr4/usr/lib && tar -xpf -'
```

```
(me@svr4) cd /usr/ucblib
(me@svr4) tar -cf - . | \
          rsh netbsd 'cd /emul/svr4/usr/ucblib && tar -xpf -'
```

If you are running openwindows:

```
(me@svr4) cd /usr/openwin/lib
(me@svr4) tar -cf - . | \
          rsh netbsd 'cd /emul/svr4/usr/openwin/lib && tar -xpf -'
(me@svr4) cd /usr/dt/lib
(me@svr4) tar -cf - . | \
          rsh netbsd 'cd /emul/svr4/usr/dt/lib && tar -xpf -'
```

3. You will also probably need the timezone files from your Solaris system, otherwise emulated binaries will run on UTC time.

```
(me@netbsd) mkdir -p /emul/svr4/usr/share/lib/zoneinfo
(me@netbsd) mkdir -p /emul/svr4/etc/default
(me@svr4) cd /usr/share/lib/zoneinfo
(me@solaris) tar -cf -. | \
          rsh netbsd 'cd /emul/svr4/usr/share/lib/zoneinfo &&
          tar -xpf -'
(me@netbsd) echo TZ=US/Pacific > /emul/svr4/etc/default/init
```

4. Set up the configuration files and devices:

```
(me@netbsd) cd /usr/share/examples/emul/svr4/etc
(me@netbsd) cp netconfig nsswitch.conf /emul/svr4/etc
(me@netbsd) cp SVR4_MAKEDEV /emul/svr4/dev
(me@netbsd) cd /emul/svr4/dev && sh SVR4_MAKEDEV all
```

As the major number allocated for emulation of SVR4 devices may vary between NetBSD platforms, the SVR4_MAKEDEV script uses the `uname`(1) command to determine the architecture the devices nodes are being created for; this can be overridden by setting the `MACHINE` environment variable accordingly.

An alternative method is to mount a whole SVR4 partition in `/emul/svr4` and then override with other mounts `/emul/svr4/etc` and `/emul/svr4/dev`.

BUGS

Many system calls are still not emulated. The streams emulation is incomplete (socketpair does not work yet).

Most SVR4 executables can not handle directory offset cookies > 32 bits. More recent ones, compiled for large file support (Solaris 2.6 and up) can. With older programs, you will see the message "svr4_getdents: dir offset too large for emulated program"" when this happens. Currently, this can only happen on NFS mounted filesystems, mounted from servers that return offsets with information in the upper 32 bits. These errors should rarely happen, but can be avoided by mounting this filesystem with offset translation enabled. See the **-X** option to `mount_nfs`(8). The **-2** option to `mount_nfs`(8) will also have the desired effect, but is less preferable.

NAME

compat_ultrix — setup procedure for ULTRIX compatibility on MIPS and VAX architectures

DESCRIPTION

NetBSD/mips and NetBSD/vax architectures can run Risc ULTRIX and VAX ULTRIX executables, respectively. However, you have to worry about the legal issues of ensuring that you have a right to use any ULTRIX binaries on your machine.

Most executables will work. The exceptions include programs that use proprietary, ULTRIX-specific features (LAT, CI support, DECnet support) and various system calls, **ioctl**()'s, or ULTRIX kernel semantics that are difficult to emulate (e.g. ULTRIX packetfilter) or buggy (e.g. ULTRIX NIS).

All ULTRIX executables are static, so no shared libraries are required for ULTRIX compatibility. However, ULTRIX is based on a 4.3BSD alpha release. ULTRIX commands and libraries are often much older than their NetBSD or even SunOS 4.x equivalents, and may require incompatible configuration files.

SYSTEM CONFIGURATION FILES

Set up `resolv.conf` and `svc.conf` as below:

```
# mkdir -p /emul/ultrix/etc
# cd /emul/ultrix/etc
# egrep 'domain|nameserver' /etc/resolv.conf > ./resolv.conf
# cp -p /usr/share/examples/emul/ultrix/etc/* ./
```

/etc/resolv.conf

The ULTRIX resolver library only understands **domain** and **nameserver** lines in `resolv.conf`(5). You should create a copy of `/etc/resolv.conf` containing only those commands and put it in `/emul/ultrix/etc/resolv.conf`. Note that the domain search order used by ULTRIX executables may not be the same as native binaries; there is no good way around this.

/etc/svc.conf

ULTRIX uses `/etc/svc.conf` to select an ordered search of NIS, Hesiod, or local flat-file mappings. You should create an `/emul/ultrix/etc/svc.conf` specifying either local files or bind (DNS) lookups for all ULTRIX name services.

SEE ALSO

`resolv.conf`(5)

BUGS

RISC ULTRIX NIS (YP) is known to not work. The ULTRIX NIS libraries have a consistent endian-ness bug. ULTRIX NIS client will not inter-operate with the NetBSD `ypbind`(8) process. The only workaround is to use `/etc/svc.conf` to disable NIS (YP).

The ndbm hashed-password file used by ULTRIX are incompatible with the db hashed-password file used by NetBSD. There is no good solution for this. NIS would be a good one, if ULTRIX NIS worked.

The API used by Xservers to talk to the kernel is currently compatible with ULTRIX 4.1. An implementation of the ULTRIX 4.2 Xws interface (used by X11R6) is in progress.

A complete list of things which fail to work in ULTRIX compatibility mode should be added here.

NAME

comsat — biff server

SYNOPSIS

comsat [**-l**]

DESCRIPTION

comsat is the server process which receives reports of incoming mail and notifies users if they have requested this service. **comsat** receives messages on a datagram port associated with the "biff" service specification (see `services`(5) and `inetd`(8)). The one line messages are of the form:

```
user@mailbox-offset
```

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a "`biff y`"), the *offset* is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the "From", "To", "Date", or "Subject" lines are not included in the displayed message.

OPTIONS

The **comsat** program supports this option:

-l The **-l** option turns on `syslogd`(8) log messages.

FILES

To find out who's logged on and on what terminals **comsat** consults:

```
/var/run/utmp
/var/run/utmpx
```

SEE ALSO

`biff`(1), `inetd`(8), `syslogd`(8)

HISTORY

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

BUGS

The message header filtering is prone to error. The density of the information presented is near the theoretical minimum.

Users should be notified of mail which arrives on other machines than the one to which they are currently logged in.

The notification should appear in a separate window so it does not mess up the screen.

comsat runs as root so that it can open the users maildrop.

NAME

cpuctl — a program to control CPUs

SYNOPSIS

cpuctl *command* [*arguments*]

DESCRIPTION

The **cpuctl** command can be used to control and inspect the state of CPUs in the system.

The first argument, *command*, specifies the action to take. Valid commands are:

identify *cpu*

Output information on the specified CPU's features and capabilities. Not available on all architectures.

list

For each CPU in the system, display the current state and time of the last state change.

offline *cpuno*

Set the specified CPU off line.

Unbound LWPs (lightweight processes) will not be executed on the CPU while it is off line. Bound LWPs will continue to be executed on the CPU, and device interrupts routed to the CPU will continue to be handled. A future release of the system may allow device interrupts to be re-routed away from individual CPUs.

At least one CPU in the system must remain on line.

online *cpuno*

Set the specified CPU on line, making it available to run unbound LWPs.

FILES

`/dev/cpuctl` control device

HISTORY

The **cpuctl** command first appeared in NetBSD 5.0.

NAME

crash — UNIX system failures

DESCRIPTION

This section explains a bit about system crashes and (very briefly) how to analyze crash dumps.

When the system crashes voluntarily it prints a message of the form

> panic: why i gave up the ghost

on the console, takes a dump on a mass storage peripheral, and then invokes an automatic reboot procedure as described in `reboot`(8). Unless some unexpected inconsistency is encountered in the state of the file systems due to hardware or software failure, the system will then resume multi-user operations.

The system has a large number of internal consistency checks; if one of these fails, then it will panic with a very short message indicating which one failed. In many instances, this will be the name of the routine which detected the error, or a two-word description of the inconsistency. A full understanding of most panic messages requires perusal of the source code for the system.

The most common cause of system failures is hardware failure, which can reflect itself in different ways. Here are the messages which are most likely, with some hints as to causes. Left unstated in all cases is the possibility that hardware or software error produced the message in some unexpected way.

iinit This cryptic panic message results from a failure to mount the root filesystem during the bootstrap process. Either the root filesystem has been corrupted, or the system is attempting to use the wrong device as root filesystem. Usually, an alternative copy of the system binary or an alternative root filesystem can be used to bring up the system to investigate.

Can't exec /sbin/init
This is not a panic message, as reboots are likely to be futile. Late in the bootstrap procedure, the system was unable to locate and execute the initialization process, `init`(8). The root filesystem is incorrect or has been corrupted, or the mode or type of `/sbin/init` forbids execution.

IO err in push
hard IO err in swap
The system encountered an error trying to write to the paging device or an error in reading critical information from a disk drive. The offending disk should be fixed if it is broken or unreliable.

realloccg: bad optim
ialloc: dup alloc
alloccgblk:cyl groups corrupted
ialloccg: map corrupted
free: freeing free block
free: freeing free frag
ifree: freeing free inode
alloccg: map corrupted
These panic messages are among those that may be produced when filesystem inconsistencies are detected. The problem generally results from a failure to repair damaged filesystems after a crash, hardware failures, or other condition that should not normally occur. A filesystem check will normally correct the problem.

timeout table overflow
This really shouldn't be a panic, but until the data structure involved is made to be extensible, running out of entries causes a crash. If this happens, make the timeout table bigger.

trap type %d, code = %x, v = %x

An unexpected trap has occurred within the system; the trap types are:

0	bus error
1	address error
2	illegal instruction
3	divide by zero
4	*chk* instruction
5	*trapv* instruction
6	privileged instruction
7	trace trap
8	MMU fault
9	simulated software interrupt
10	format error
11	FP coprocessor fault
12	coprocessor fault
13	simulated AST

The favorite trap type in system crashes is trap type 8, indicating a wild reference. ‘‘code’’ (hex) is the concatenation of the MMU status register (see <hp300/cpu.h>) in the high 16 bits and the 68020 special status word (see the 68020 manual, page 6-17) in the low 16. ‘‘v’’ (hex) is the virtual address which caused the fault. Additionally, the kernel will dump about a screenful of semi-useful information. ‘‘pid’’ (decimal) is the process id of the process running at the time of the exception. Note that if we panic in an interrupt routine, this process may not be related to the panic. ‘‘ps’’ (hex) is the 68020 processor status register ‘‘ps’’. ‘‘pc’’ (hex) is the value of the program counter saved on the hardware exception frame. It may *not* be the PC of the instruction causing the fault. ‘‘sfc’’ and ‘‘dfc’’ (hex) are the 68020 source/destination function codes. They should always be one. ‘‘p0’’ and ‘‘p1’’ are the VAX-like region registers. They are of the form:

<length> ’@’ <kernel VA>

where both are in hex. Following these values are a dump of the processor registers (hex). Finally, is a dump of the stack (user/kernel) at the time of the offense.

init died

The system initialization process has exited. This is bad news, as no new users will then be able to log in. Rebooting is the only fix, so the system just does it right away.

out of mbufs: map full

The network has exhausted its private page map for network buffers. This usually indicates that buffers are being lost, and rather than allow the system to slowly degrade, it reboots immediately. The map may be made larger if necessary.

That completes the list of panic types you are likely to see.

When the system crashes it writes (or at least attempts to write) an image of memory into the back end of the dump device, usually the same as the primary swap area. After the system is rebooted, the program `savecore`(8) runs and preserves a copy of this core image and the current system in a specified directory for later perusal. See `savecore`(8) for details.

To analyze a dump you should begin by running `adb`(1) with the **`-k`** flag on the system load image and core dump. If the core image is the result of a panic, the panic message is printed. Normally the command ‘‘$c’’ will provide a stack trace from the point of the crash and this will provide a clue as to what went wrong. For more details consult *Using ADB to Debug the UNIX Kernel*.

SEE ALSO

adb(1), reboot(8)

MC68020 32-bit Microprocessor User's Manual.

Using ADB to Debug the UNIX Kernel.

4.3BSD for the HP300.

HISTORY

A **crash** man page appeared in Version 6 AT&T UNIX.

NAME

crash — UNIX system failures

DESCRIPTION

This section explains what happens when the system crashes and (very briefly) how to analyze crash dumps.

When the system crashes voluntarily it prints a message of the form

```
panic: why i gave up the ghost
```

on the console, takes a dump on a mass storage peripheral, and then invokes an automatic reboot procedure as described in `reboot`(8). (If auto-reboot is disabled on the front panel of the machine the system will simply halt at this point.) Unless some unexpected inconsistency is encountered in the state of the file systems due to hardware or software failure, the system will then resume multi-user operations.

The system has a large number of internal consistency checks; if one of these fails, then it will panic with a very short message indicating which one failed. In many instances, this will be the name of the routine which detected the error, or a two-word description of the inconsistency. A full understanding of most panic messages requires perusal of the source code for the system.

The most common cause of system failures is hardware failure, which can reflect itself in different ways. Here are the messages which are most likely, with some hints as to causes. Left unstated in all cases is the possibility that hardware or software error produced the message in some unexpected way.

iinit
: This cryptic panic message results from a failure to mount the root filesystem during the bootstrap process. Either the root filesystem has been corrupted, or the system is attempting to use the wrong device as root filesystem. Usually, an alternative copy of the system binary or an alternative root filesystem can be used to bring up the system to investigate.

Can't exec /sbin/init
: This is not a panic message, as reboots are likely to be futile. Late in the bootstrap procedure, the system was unable to locate and execute the initialization process, `init`(8). The root filesystem is incorrect or has been corrupted, or the mode or type of /sbin/init forbids execution.

IO err in push
hard IO err in swap
: The system encountered an error trying to write to the paging device or an error in reading critical information from a disk drive. The offending disk should be fixed if it is broken or unreliable.

realloccg: bad optim
ialloc: dup alloc
alloccgblk: cyl groups corrupted
ialloccg: map corrupted
free: freeing free block
free: freeing free frag
ifree: freeing free inode
alloccg: map corrupted
: These panic messages are among those that may be produced when filesystem inconsistencies are detected. The problem generally results from a failure to repair damaged filesystems after a crash, hardware failures, or other condition that should not normally occur. A filesystem check will normally correct the problem.

timeout table overflow
: This really shouldn't be a panic, but until the data structure involved is made to be extensible, running out of entries causes a crash. If this happens, make the timeout table bigger.

KSP not valid
SBI fault

CHM? in kernel

These indicate either a serious bug in the system or, more often, a glitch or failing hardware. If SBI faults recur, check out the hardware or call field service. If the other faults recur, there is likely a bug somewhere in the system, although these can be caused by a flakey processor. Run processor microdiagnostics.

machine check %x: *description*

machine dependent machine-check information

Machine checks are different on each type of CPU. Most of the internal processor registers are saved at the time of the fault and are printed on the console. For most processors, there is one line that summarizes the type of machine check. Often, the nature of the problem is apparent from this message and/or the contents of key registers. The VAX Hardware Handbook should be consulted, and, if necessary, your friendly field service people should be informed of the problem.

trap type %d, code=%x, pc=%x

A unexpected trap has occurred within the system; the trap types are:

```
0       reserved addressing fault
1       privileged instruction fault
2       reserved operand fault
3       bpt instruction fault
4       xfc instruction fault
5       system call trap
6       arithmetic trap
7       ast delivery trap
8       segmentation fault
9       protection fault
10      trace trap
11      compatibility mode fault
12      page fault
13      page table fault
```

The favorite trap types in system crashes are trap types 8 and 9, indicating a wild reference. The code is the referenced address, and the pc at the time of the fault is printed. These problems tend to be easy to track down if they are kernel bugs since the processor stops cold, but random flakiness seems to cause this sometimes. The debugger can be used to locate the instruction and subroutine corresponding to the PC value. If that is insufficient to suggest the nature of the problem, more detailed examination of the system status at the time of the trap usually can produce an explanation.

init died The system initialization process has exited. This is bad news, as no new users will then be able to log in. Rebooting is the only fix, so the system just does it right away.

out of mbufs: map full

The network has exhausted its private page map for network buffers. This usually indicates that buffers are being lost, and rather than allow the system to slowly degrade, it reboots immediately. The map may be made larger if necessary.

That completes the list of panic types you are likely to see.

When the system crashes it writes (or at least attempts to write) an image of memory into the back end of the dump device, usually the same as the primary swap area. After the system is rebooted, the program `savecore`(8) runs and preserves a copy of this core image and the current system in a specified directory for later perusal. See `savecore`(8) for details.

To analyze a dump you should begin by running **adb** with the **-k** flag on the system load image and core dump. If the core image is the result of a panic, the panic message is printed. Normally the command "$c" will provide a stack trace from the point of the crash and this will provide a clue as to what went wrong. For more detail see “Using ADB to Debug the UNIX Kernel”.

SEE ALSO

`gdb`(1), `reboot`(8)
“VAX 11/780 System Maintenance Guide” and “VAX Hardware Handbook” for more information about machine checks.
“Using ADB to Debug the UNIX Kernel”

NAME

cron — daemon to execute scheduled commands (Vixie Cron)

SYNOPSIS

cron [**-x** *debugflags*]

DESCRIPTION

cron is normally started during system boot by `rc.d`(8) framework, if cron is switched on in `rc.conf`(5).

cron searches `/var/cron/tabs` for crontab files which are named after accounts in `/etc/passwd`. Crontabs found are loaded into memory. **cron** also searches for `/etc/crontab` which is in a different format (see `crontab`(5)). **cron** then wakes up every minute, examining all stored crontabs, checking each command to see if it should be run in the current minute. When executing commands, any output is mailed to the owner of the crontab (or to the user named in the `MAILTO` environment variable in the crontab, if such exists).

Events such as `START` and `FINISH` are recorded in the `/var/log/cron` log file with date and time details. This information is useful for a number of reasons, such as determining the amount of time required to run a particular job. By default, root has an hourly job that rotates these log files with compression to preserve disk space.

Additionally, **cron** checks each minute to see if its spool directory's modtime (or the modtime on `/etc/crontab`) has changed, and if it has, **cron** will then examine the modtime on all crontabs and reload those which have changed. Thus **cron** need not be restarted whenever a crontab file is modified. Note that the `crontab`(1) command updates the modtime of the spool directory whenever it changes a crontab.

The **-x** flag turns on some debugging flags. *debugflags* is comma-separated list of debugging flags to turn on. If a flag is turned on, **cron** writes some additional debugging information to system log during its work. Available debugging flags are:

sch	scheduling
proc	process control
pars	parsing
load	database loading
misc	miscellaneous
test	test mode - do not actually execute any commands
bit	show how various bits are set (long)
ext	print extended debugging information

FILES

`/var/cron/tabs`	**cron** spool directory
`/etc/crontab`	system crontab
`/var/log/cron`	log file for cron events

SEE ALSO

`crontab`(1), `crontab`(5)

AUTHORS

Paul Vixie ⟨paul@vix.com⟩

NAME

cvsbug − send problem report (PR) about CVS to a central support site

SYNOPSIS

cvsbug [*site*] [**−f** *problem-report*] [**−t** *mail-address*]
[**−P**] [**−L**] [**−−request-id**] [**−v**]

DESCRIPTION

cvsbug is a tool used to submit *problem reports* (PRs) to a central support site. In most cases the correct *site* will be the default. This argument indicates the support site which is responsible for the category of problem involved. Some sites may use a local address as a default. *site* values are defined by using the **aliases**(5).

cvsbug invokes an editor on a problem report template (after trying to fill in some fields with reasonable default values). When you exit the editor, **cvsbug** sends the completed form to the *Problem Report Management System* (**GNATS**) at a central support site. At the support site, the PR is assigned a unique number and is stored in the **GNATS** database according to its category and submitter-id. **GNATS** automatically replies with an acknowledgement, citing the category and the PR number.

To ensure that a PR is handled promptly, it should contain your (unique) *submitter-id* and one of the available *categories* to identify the problem area. (Use **'cvsbug -L'** to see a list of categories.)

The **cvsbug** template at your site should already be customized with your submitter-id (running ' **install-sid** *submitter-id* ' to accomplish this is part of the installation procedures for **cvsbug**). If this hasn't been done, see your system administrator for your submitter-id, or request one from your support site by invoking **'cvsbug −−request−id'.** If your site does not distinguish between different user sites, or if you are not affiliated with the support site, use **'net'** for this field.

The more precise your problem description and the more complete your information, the faster your support team can solve your problems.

OPTIONS

−f *problem-report*
specify a file (*problem-report*) which already contains a complete problem report. **cvsbug** sends the contents of the file without invoking the editor. If the value for *problem-report* is **'−'**, then **cvsbug** reads from standard input.

−t *mail-address*
Change mail address at the support site for problem reports. The default *mail-address* is the address used for the default *site*. Use the *site* argument rather than this option in nearly all cases.

−P print the form specified by the environment variable **PR_FORM** on standard output. If **PR_FORM** is not set, print the standard blank PR template. No mail is sent.

-L print the list of available categories. No mail is sent.

−−request−id
sends mail to the default support site, or *site* if specified, with a request for your *submitter-id*. If you are not affiliated with *site*, use a *submitter-id* of **net** '.

−v Display the **cvsbug** version number.

Note: use **cvsbug** to submit problem reports rather than mailing them directly. Using both the template and **cvsbug** itself will help ensure all necessary information will reach the support site.

ENVIRONMENT

The environment variable **EDITOR** specifies the editor to invoke on the template.
default: **vi**

If the environment variable **PR_FORM** is set, then its value is used as the file name of the template for your problem-report editing session. You can use this to start with a partially completed form (for example, a form with the identification fields already completed).

HOW TO FILL OUT A PROBLEM REPORT

Problem reports have to be in a particular form so that a program can easily manage them. Please remember the following guidelines:

- describe only **one problem** with each problem report.
- For follow-up mail, use the same subject line as the one in the automatic acknowledgement. It consists of category, PR number and the original synopsis line. This allows the support site to relate several mail messages to a particular PR and to record them automatically.
- Please try to be as accurate as possible in the subject and/or synopsis line.
- The subject and the synopsis line are not confidential. This is because open-bugs lists are compiled from them. Avoid confidential information there.

See the GNU **Info** file **cvsbug.info** or the document *Reporting Problems With cvsbug* for detailed information on reporting problems

HOW TO SUBMIT TEST CASES, CODE, ETC.

Submit small code samples with the PR. Contact the support site for instructions on submitting larger test cases and problematic source code.

FILES

/tmp/p$$	copy of PR used in editing session
/tmp/pf$$	copy of empty PR form, for testing purposes
/tmp/pbad$$	file for rejected PRs

INSTALLATION AND CONFIGURATION

See **INSTALL** for installation instructions.

SEE ALSO

gnats(l), **query-pr**(1), **edit-pr**(1), **gnats**(8), **queue-pr**(8), **at-pr**(8), **mkcat**(8), **mkdist**(8).

AUTHORS

Jeffrey Osier, Brendan Kehoe, Jason Merrill, Heinz G. Seidl (Cygnus Support)

COPYING

NAME

dbsym — copy kernel symbol table into db_symtab space

SYNOPSIS

dbsym [**−v**] [**−b** *bfdname*] *kernel*

DESCRIPTION

dbsym is used to copy the symbol table in a newly linked kernel into the *db_symtab* array (in the data section) so that the ddb(4) kernel debugger can find the symbols. This program is only used on systems for which the boot program does not load the symbol table into memory with the kernel. The space for these symbols is reserved in the data segment using a config option like:

```
options         SYMTAB_SPACE=72000
```

The size of the db_symtab array (the value of SYMTAB_SPACE) must be at least as large as the kernel symbol table. If insufficient space is reserved, dbsym will refuse to copy the symbol table.

To recognize kernel executable format, the **−b** flag specifies BFD name of kernel.

If the **−v** flag is given, **dbsym** will print out status information as it is copying the symbol table.

Note that debugging symbols are not useful to the ddb(4) kernel debugger, so to minimize the size of the kernel, one should either compile the kernel without debugging symbols (no **−g** flag) or use the strip(1) command to strip debugging symbols from the kernel before **dbsym** is used to copy the symbol table. The command

```
strip -d netbsd
```

will strip out debugging symbols.

SEE ALSO

strip(1), ddb(4)

NAME

dev_mkdb — create /dev database

SYNOPSIS

dev_mkdb [**-o** *database*] [directory]

DESCRIPTION

The **dev_mkdb** command creates a db(3) hash access method database in "/var/run/dev.db" which contains the names of all of the character and block special files in the specified directory, using the file type and the *st_rdev* field as the key. If no directory is specified, the "/dev" directory is used.

Keys are a structure containing a mode_t followed by a dev_t, with any padding zero'd out. The former is the type of the file (st_mode & S_IFMT), the latter is the st_rdev field.

The options are as follows:

-o *database*
Put the output databases in the named file.

FILES

/dev	Device directory.
/var/run/dev.db	Database file.

SEE ALSO

ps(1), stat(2), db(3), devname(3), kvm_nlist(3), ttyname(3), kvm_mkdb(8)

HISTORY

The **dev_mkdb** command appeared in 4.4BSD.

NAME

dhclient-script - DHCP client network configuration script

DESCRIPTION

The DHCP client network configuration script is invoked from time to time by **dhclient(8)**. This script is used by the dhcp client to set each interface's initial configuration prior to requesting an address, to test the address once it has been offered, and to set the interface's final configuration once a lease has been acquired. If no lease is acquired, the script is used to test predefined leases, if any, and also called once if no valid lease can be identified.

This script is not meant to be customized by the end user. If local customizations are needed, they should be possible using the enter and exit hooks provided (see HOOKS for details). These hooks will allow the user to override the default behaviour of the client in creating a **/etc/resolv.conf** file.

No standard client script exists for some operating systems, even though the actual client may work, so a pioneering user may well need to create a new script or modify an existing one. In general, customizations specific to a particular computer should be done in the **/etc/dhclient.conf** file. If you find that you can't make such a customization without customizing **/etc/dhclient.conf** or using the enter and exit hooks, please submit a bug report.

HOOKS

When it starts, the client script first defines a shell function, **make_resolv_conf ,** which is later used to create the **/etc/resolv.conf** file. To override the default behaviour, redefine this function in the enter hook script.

On after defining the make_resolv_conf function, the client script checks for the presence of an executable **/etc/dhclient-enter-hooks** script, and if present, it invokes the script inline, using the Bourne shell '.' command. The entire environment documented under OPERATION is available to this script, which may modify the environment if needed to change the behaviour of the script. If an error occurs during the execution of the script, it can set the exit_status variable to a nonzero value, and **/sbin/dhclient-script** will exit with that error code immediately after the client script exits.

After all processing has completed, **/sbin/dhclient-script** checks for the presence of an executable **/etc/dhclient-exit-hooks** script, which if present is invoked using the '.' command. The exit status of dhclient-script will be passed to dhclient-exit-hooks in the exit_status shell variable, and will always be zero if the script succeeded at the task for which it was invoked. The rest of the environment as described previously for dhclient-enter-hooks is also present. The **/etc/dhclient-exit-hooks** script can modify the valid of exit_status to change the exit status of dhclient-script.

OPERATION

When dhclient needs to invoke the client configuration script, it defines a set of variables in the environment, and then invokes **CLIENTBINDIR/dhclient-script.** In all cases, $reason is set to the name of the reason why the script has been invoked. The following reasons are currently defined: MEDIUM, PREINIT, BOUND, RENEW, REBIND, REBOOT, EXPIRE, FAIL and TIMEOUT.

MEDIUM

The DHCP client is requesting that an interface's media type be set. The interface name is passed in $interface, and the media type is passed in $medium.

PREINIT

The DHCP client is requesting that an interface be configured as required in order to send packets prior to receiving an actual address. For clients which use the BSD socket library, this means configuring the interface with an IP address of 0.0.0.0 and a broadcast address of 255.255.255.255. For other clients, it may be possible to simply configure the interface up without actually giving it an IP address at all. The interface name is passed in $interface, and the media type in $medium.

If an IP alias has been declared in dhclient.conf, its address will be passed in $alias_ip_address, and that ip alias should be deleted from the interface, along with any routes to it.

BOUND

The DHCP client has done an initial binding to a new address. The new ip address is passed in $new_ip_address, and the interface name is passed in $interface. The media type is passed in $medium. Any options acquired from the server are passed using the option name described in **dhcp-options**, except that dashes ('-') are replaced by underscores ('_') in order to make valid shell variables, and the variable names start with new_. So for example, the new subnet mask would be passed in $new_subnet_mask.

Before actually configuring the address, dhclient-script should somehow ARP for it and exit with a nonzero status if it receives a reply. In this case, the client will send a DHCPDECLINE message to the server and acquire a different address. This may also be done in the RENEW, REBIND, or REBOOT states, but is not required, and indeed may not be desirable.

When a binding has been completed, a lot of network parameters are likely to need to be set up. A new /etc/resolv.conf needs to be created, using the values of $new_domain_name and $new_domain_name_servers (which may list more than one server, separated by spaces). A default route should be set using $new_routers, and static routes may need to be set up using $new_static_routes.

If an IP alias has been declared, it must be set up here. The alias IP address will be written as $alias_ip_address, and other DHCP options that are set for the alias (e.g., subnet mask) will be passed in variables named as described previously except starting with $alias_ instead of $new_. Care should be taken that the alias IP address not be used if it is identical to the bound IP address ($new_ip_address), since the other alias parameters may be incorrect in this case.

RENEW

When a binding has been renewed, the script is called as in BOUND, except that in addition to all the variables starting with $new_, there is another set of variables starting with $old_. Persistent settings that may have changed need to be deleted - for example, if a local route to the bound address is being configured, the old local route should be deleted. If the default route has changed, the old default route should be deleted. If the static routes have changed, the old ones should be deleted. Otherwise, processing can be done as with BOUND.

REBIND

The DHCP client has rebound to a new DHCP server. This can be handled as with RENEW, except that if the IP address has changed, the ARP table should be cleared.

REBOOT

The DHCP client has successfully reacquired its old address after a reboot. This can be processed as with BOUND.

EXPIRE

The DHCP client has failed to renew its lease or acquire a new one, and the lease has expired. The IP address must be relinquished, and all related parameters should be deleted, as in RENEW and REBIND.

FAIL

The DHCP client has been unable to contact any DHCP servers, and any leases that have been tested have not proved to be valid. The parameters from the last lease tested should be deconfigured. This can be handled in the same way as EXPIRE.

TIMEOUT

The DHCP client has been unable to contact any DHCP servers. However, an old lease has been identified, and its parameters have been passed in as with BOUND. The client configuration script should test these parameters and, if it has reason to believe they are valid, should exit with a value of zero. If not, it should exit with a nonzero value.

The usual way to test a lease is to set up the network as with REBIND (since this may be called to test more than one lease) and then ping the first router defined in $routers. If a response is received, the lease must be valid for the network to which the interface is currently connected. It would be more complete to try to ping all of the routers listed in $new_routers, as well as those listed in $new_static_routes, but current scripts do not do this.

FILES

Each operating system should generally have its own script file, although the script files for similar operating systems may be similar or even identical. The script files included in Internet Systems Consortium DHCP distribution appear in the distribution tree under client/scripts, and bear the names of the operating systems on which they are intended to work.

BUGS

If more than one interface is being used, there's no obvious way to avoid clashes between server-supplied configuration parameters - for example, the stock dhclient-script rewrites /etc/resolv.conf. If more than one interface is being configured, /etc/resolv.conf will be repeatedly initialized to the values provided by one server, and then the other. Assuming the information provided by both servers is valid, this shouldn't cause any real problems, but it could be confusing.

SEE ALSO

dhclient(8), dhcpd(8), dhcrelay(8), dhclient.conf(5) and dhclient.leases(5).

AUTHOR

dhclient-script(8) has been written for Internet Systems Consortium by Ted Lemon in cooperation with Vixie Enterprises. To learn more about Internet Systems Consortium, see **http://www.isc.org.** To learn more about Vixie Enterprises, see **http://www.vix.com.**

Ì¾¾Î

dhclient-script - DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Î¥Í¥Ã¥È¥ï¡¼¥¯ÀßÄê¥¹¥¯¥ê¥×¥È

²òÀâ

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Î¥Í¥Ã¥È¥ï¡¼¥¯ÀßÄê¥¹¥¯¥ê¥×¥È¤Ï¡¢ »þ¤¢¤ë´¤È¤Ë **dhclient(8)** ¤¬¸Æ¤Ó½Ð¤·¤Þ¤¹¡£ DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ËÜ¥¹¥¯¥ê¥×¥È¤ò»ÈÍÑ¤¹¤ë¤³¤È¤Ë¤è¤ê¡¢ ¥¢¥É¥ì¥¹Í×µá¤ËÀèÎ©¤Ä³Æ¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤Î½é´üÀßÄê¤È¡¢ ÉÕÍ¿¤µ¤ì¤¿¥¢¥É¥ì¥¹¤Î¸¡¾Ú¤È¡¢ ¥ê¡¼¥¹³ÍÆÀ»þ¤Î¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ÎºÇ½ªÀßÄê¤ò¹Ô¤¤¤Þ¤¹¡£ ¥ê¡¼¥¹¤¬³ÍÆÀ¤µ¤ì¤Ê¤«¤Ã¤¿¾ì¹ç¡¢ ÄêµÁºÑ¤ß¤Î¥ê¡¼¥¹¤¬Â¸ºß¤¹¤ë¤Ê¤é¤Ð¤³¤ì¤ò¸¡¾Ú¤¹¤ë¤¿¤á¤ËËÜ¥¹¥¯¥ê¥×¥È¤Ï»ÈÍÑ¤µ¤ì¡¢ Í¸ú¤Ê¥ê¡¼¥¹¤¬È½ÌÀ¤·¤Ê¤«¤Ã¤¿¾ì¹ç¤Ë¤â¤¦ 1 ²ó¤³¤Î¥¹¥¯¥ê¥×¥È¤¬¸Æ¤Ð¤ì¤Þ¤¹¡£

ËÜ¥¹¥¯¥ê¥×¥È¤Ï¡¢¥¨¥ó¥É¥æ¡¼¥¶¤Ë¥«¥¹¥¿¥Þ¥¤¥º¤µ¤ì¤ë¤³¤È¤ò°Õ¿Þ¤·¤Æ¤¤¤Þ¤»¤ó¡£ ¥í¡¼¥«¥ë¤Ê¥«¥¹¥¿¥Þ¥¤¥º¤¬É¬Í×¤Ê¾ì¹ç¡¢ ¤³¤ì¤Ï¥Æ¥Þ (enter) ¤È½Ð (exit) ¤È¤¤¤¦¥Õ¥Ã¥¯¤ò»ÈÍÑ¤¹¤ë¤³¤È¤Ç²ÄÇ½¤È¤Ê¤ê¤Þ¤¹ (¾Ü°Ù¤Ï¥Õ¥Ã¥¯»²¾È)¡£ ¤³¤ì¤é¤Î¥Õ¥Ã¥¯¤Ï¡¢ **/etc/resolv.conf** ºîÀ®»þ¤Ë¡¢ ¥¯¥é¥¤¥¢¥ó¥È¤Î¥Ç¥Õ¥©¥ë¥ÈÆ°ºî¤ò¥æ¡¼¥¶¤¬¥ª¡¼¥Ð¥é¥¤¥É¤Ç¤¤ë¤è¤¦¤Ë¤·¤Þ¤¹¡£

ÆÃÄê¤Î¥ª¥Ú¥ì¡¼¥Æ¥£¥ó¥°¥·¥¹¥Æ¥à¤Ç¤Ï¡¢ ¥¯¥é¥¤¥¢¥ó¥È¤Î¼ÂÁõ¤ÏÆ°ºî¤¹¤ë¤È¤·¤Æ¤â¡¢ É¸½à¤Î¥¹¥¯¥ê¥×¥È¤¬Æ°ºî¤·¤Ê¤¤¤«¤â¤·¤ì¤Þ¤»¤ó¡£ Àè¶îÅª¤Ê¥æ¡¼¥¶¤¬¿·µ¬¥¹¥¯¥ê¥×¥È¤òºîÀ®¤·¤¿¤ê´ûÂ¸¤Î¤â¤Î¤ò½¤Àµ¤·¤¿¤ê¤¹¤ëÉ¬Í×¤¬¤¢¤ë ¤³¤È¤Ï¤â¤Ã¤È¤â¤Ê¤³¤È¤Ç¤¹¡£ °ìÈÌÅª¤Ë¤Ï¡¢¤½¤Î¤è¤¦¤Ê¤Î¥³¥ó¥Ô¥å¡¼¥¿¤Ë¸ÇÍ¤Î¥«¥¹¥¿¥Þ¥¤¥º¤Ï **ETCDIR/dhclient.conf** ¥¹¥¯¥ê¥×¥È¤Ç¹Ô¤¦¤Ù¤¤¤Ç¤¹¡£ **ETCDIR/dhclient.conf** ¤Î¥«¥¹¥¿¥Þ¥¤¥º¤Ìµ¤·¤Ë¤Ç¤¤Ê¤¤¥«¥¹¥¿¥Þ¥¤¥º¤ä¡¢ Æþ¤È½Ð¤Î¥Õ¥Ã¥¯¤Î»ÈÍÑ¤Ç¤Ï¤Ç¤¤Ê¤¤¥«¥¹¥¿¥Þ¥¤¥º¤ËµÚ¤Ó¤¿¤¤¾ì¹ç¤Ë¤Ï¡¢ ¥Ð°¥ì¥Ý¡¼¥È¤òÁ÷¤Ã¤Æ¤¯¤À¤µ¤¤¡£

¥Õ¥Ã¥¯

³«»Ï»þ¤Ë¡¢¥¯¥é¥¤¥¢¥ó¥È¥¹¥¯¥ê¥×¥È¤Ï¤Þ¤º¥·¥§¥ë´Ø¿ô¤òÄêµÁ¤·¤Þ¤¹¡£¤½¤Î´Ø¿ô¤Ï **make_resolv_conf** ¤Ç¤¢¤ê¡¢¸å¤Ë **/etc/resolv.conf** ¥Õ¥¡¥¤¥ë¤òºîÀ®¤¹¤ë¤¿¤á¤Ë»ÈÍÑ¤µ¤ì¤Þ¤¹¡£ ¥Ç¥Õ¥©¥ë¥ÈÆ°ºî¤ò¥ª¡¼¥Ð¥é¥¤¥É¤¹¤ë¤Ë¤Ï¡¢ ¤³¤Î´Ø¿ô¤òÆþ¤Î¥Õ¥Ã¥¯¥¹¥¯¥ê¥×¥È¤ÇºÆÄêµÁ¤·¤Æ¤¯¤À¤µ¤¤¡£

make_resolv_conf ´Ø¿ô¤ÎÄêµÁ¤Î¸å¡¢¥¯¥é¥¤¥¢¥ó¥È¥¹¥¯¥ê¥×¥È¤Ï ¼Â¹Ô²ÄÇ½¤Ê **ETCDIR/dhclient-enter-hooks** ¥¹¥¯¥ê¥×¥È¤Î¸ºß¤ò¸¡¤º¤·¡¢ Â¸ºß¤¹¤ë¾ì¹ç¤Ï Bourne ¥·¥§¥ë¤Î '. ' ¥³¥Þ¥ó¥É¤ò»ÈÍÑ¤·¤Æ ËÜ¥¹¥¯¥ê¥×¥È¤ò¥¤¥ó¥é¥¤¥ó¤Çµ¯Æ°¤·¤Þ¤¹¡£ Áà°î¤Çµ½Ò¤µ¤ì¤Æ¤¤¤ë¤¹¤Ù¤Æ¤Î´Ä¶¤¬ËÜ¥¹¥¯¥ê¥×¥È¤Ç»ÈÍÑ²ÄÇ½¤Ç¤¢¤ê¡¢ ¥¹¥¯¥ê¥×¥È¤ÎÆ°ºî¤ÎÊÑ¹¹¤¬É¬Í×¤Ê¾ì¹ç¤Ë¤Ï´Ä¶¤Î½¤Àµ¤¬µö¤µ¤ì¤Æ¤¤¤Þ¤¹¡£ ¥¹¥¯¥ê¥×¥È¼Â¹ÔÃæ¤Ë¥¨¥é¡¼¤¬È¯À¸¤¹¤ë¾ì¹ç¡¢ exit_status ÊÑ¿ô¤òÈó 0 ÃÍ¤ËÀßÄê¤¹¤ë¤³¤È¤¬²ÄÇ½¤Ç¤¢¤ê¡¢ ¥¯¥é¥¤¥¢¥ó¥È¥¹¥¯¥ê¥×¥È½ªÎ»Ä¾¸å¤Ë **CLIENTBINDIR/dhclient-script** ¤Ï¤½¤Î¥¨¥é¡¼¥³¡¼¥É¤Ç½ªÎ»¤·¤Þ¤¹¡£

¤¹¤Ù¤Æ¤Î½èÍý¤Î´°Î»¸å¤Ë¡¢ **CLIENTBINDIR/dhclient-script** ¤Ï¼Â¹Ô²ÄÇ½¤Ê **ETCDIR/dhclient-exit-hooks** ¥¹¥¯¥ê¥×¥È¤Î¸ºß¤ò¸¡¤º¤·¡¢Â¸ºß¤¹¤ë¾ì¹ç¤Ï '. ' ¥³¥Þ¥ó¥É¤Ç¤³¤ì¤òµ¯Æ°¤·¤Þ¤¹¡£ dhclient-script ¤Î ½ªÎ»¾õÂÖ¤Ï dhclient-exit-hooks ¤Î exit_status ¥·¥§¥ëÊÑ¿ô¤ËÅÏ¤µ¤ì¡¢ µ¯Æ°¤µ¤ì¤¿»Å»ö¤Ë¥¹¥¯¥ê¥×¥È¤¬À®¸ù¤·¤¿¾ì¹ç¤ËÃÍ¤Ï¾ï¤Ë 0 ¤Ë¤Ê¤ê¤Þ¤¹¡£ dhclient-enter-hooks ¤Î¹àÇÄ°½Ò¤·¤¿¤½¤ÎÂ¾¤Î´Ä¶¤â°ú¤·Ñ¤¬¤ì¤Þ¤¹¡£ **ETCDIR/dhclient-exit-hooks** ¤Ï exit_status ¤Ë¼ê¤ò²Ã¤¨¤Æ dhclient-script ¤ÎÌá¤êÃÍ¤òÊÑ¹¹¤Ç¤¤Þ¤¹¡£

Áà°î

dhclient ¤¬¥¯¥é¥¤¥¢¥ó¥ÈÀßÄê¥¹¥¯¥ê¥×¥È¤òµ¯Æ°¤¹¤ëÉ¬Í×¤¬¤¢¤ë¤È¤¡¢ Í¡¹¤ÊÊÑ¿ô¤ò´Ä¶¤ËÄêµÁ¤·¤Æ¤«¤é **CLIENTBINDIR/dhclient-script** ¤òµ¯Æ°¤·¤Þ¤¹¡£ ¤¹¤Ù¤Æ¤Î¾ì¹ç¤Ë¤ª¤¤¤Æ¡¢$reason ¤Ë¤Ï¥¹¥¯¥ê¥×¥È¤¬µ¯Æ°¤µ¤ì¤ëÍýÍ³Ì¾¤¬ÀßÄê¤µ¤ì¤Þ¤¹¡£ ¼¡¤ÎÍýÍ³¤¬¸½ºßÄêµÁ¤µ¤ì¤Æ¤¤¤Þ¤¹: MEDIUM, PREINIT, BOUND, RENEW, REBIND, REBOOT, EXPIRE, FAIL, TIMEOUT¡£

MEDIUM

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤Î¥á¥Ç¥£¥¢¥¿¥¤¥×¤ÎÀßÄê¤òµá¤á¤Æ¤¤¤Þ¤¹¡£ ¥¤¥ó¥¿¥Õ¥§¡¼¥¹Ì¾¤Ï $interface ¤ÇÅÏ¤µ¤ì¡¢¥á¥Ç¥£¥¢¥¿¥¤¥×¤Ï $medium ¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£

PREINIT

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ ¼ÂºÝ¤Î¥¢¥É¥ì¥¹¤ò¼õ¤±¼è¤ëÁ°¤Ë¥Ñ¥±¥Ã¥È¤òÁ÷¿®¤¹¤ë¤ÈÜÅª¤Ç¡¢ Í×µáÄÌ¤ê¤Ë¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤¬ÀßÄê¤µ¤ì¤ë¤³¤È¤òµá¤á¤Æ¤¤¤Þ¤¹¡£ BSD

¤Î¥½¥±¥Ã¥È¥é¥¤¥Ö¥é¥ê¤ò»ÈÍÑ¤¹¤ë¥¯¥é¥¤¥¢¥ó¥È¤Ç¤Ï¡¢ IP ¥¢¥É¥ì¥¹ 0.0.0.0 ¤«¤Ä¥Ö¥í¡¼¥É¥¥ã¥¹¥È¥¢¥É¥ì¥¹ 255.255.255.255 ¤Ç¡¢ ¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òÀßÄê¤¹¤ë¤³¤È¤ò°ÕÌ£¤·¤Þ¤¹¡£ Â¾¤Î¥¯¥é¥¤¥¢¥ó¥È¤Ç¤Ï¡¢ ¼Â°Ý¤Ë IP ¥¢¥É¥ì¥¹¤òÍ¿¤¨¤ë¤³¤È¤Ê¤¯Ã±¤Ë¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òÀßÄê¤¹¤ë¤³¤È¤Ç ¼Â¸½¤µ¤ì¤ë¤Ç¤·¤ç¤¦¡£ ¥¤¥ó¥¿¥Õ¥§¡¼¥¹Ì¾¤Ï $interface ¤ÇÅÏ¤µ¤ì¡¢¥á¥Ç¥£¥¢¥¿¥¤¥×¤Ï $medium ¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£

IP ¥¨¥¤¥ê¥¢¥¹¤¬ dhclient.conf ¤ÇÀë¸À¤µ¤ì¤Æ¤¤¤ë¾ì¹ç¡¢ ¤³¤Î¥¢¥É¥ì¥¹¤¬ $alias_ip_address ¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£ ËÜ IP ¥¢¥É¥ì¥¹¤Ø¤Î·Ð¥©¤È¤È¤â¤Ë¡¢ ËÜ IP ¥¢¥É¥ì¥¹¤òÂÐ¾Ý¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤«¤é´ï½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£

BOUND

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢¿·¥¢¥É¥ì¥¹¤Ø¤Î½é´ü¤Î·ë¹ç¤ò´°Î»¤·¤Þ¤·¤¿¡£ ¿·¤·¤¤ IP ¥¢¥É¥ì¥¹¤Ï $new_ip_address ¤ÇÅÏ¤µ¤ì¡¢ ¥¤¥ó¥¿¥Õ¥§¡¼¥¹Ì¾¤Ï $interface ¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£ ¥á¥Ç¥£¥¢¥¿¥¤¥×¤Ï $medium ¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£ ¥µ¡¼¥Ð¤«¤é³ÍÆÀ¤·¤¿¥ª¥×¥·¥ç¥ó¤Ï¡¢**dhcp-options** ¤ÇÀë¸À¤µ¤ì¤Æ¤¤¤ë ¥ª¥×¥·¥ç¥óÌ¾¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£ Îã³°¤È¤·¤Æ¡¢ Í¸ú¤Ê¥·¥§¥ëÊÑ¿ô¤È¤¹¤ë¤¿¤á¤Ë ¥À¥Ã¥·¥å ('-') ¤Ï¥¢¥ó¥À¥¹¥³¥¢('_')¤ÇÃÖ¤´¹¤¨¤é¤ì¡¢ ÊÑ¿ôÌ¾¤Ï new_ ¤Ç³«»Ï¤·¤Þ¤¹¡£ Îã¤¨¤Ð¡¢¿·¤·¤¤¥µ¥Ö¥Í¥Ã¥È¥Þ¥¹¥¯¤Ï $new_subnet_mask ¤ÇÅÏ¤µ¤ì¤Þ¤¹¡£

¥¢¥É¥ì¥¹¤ò¼Â°Ý¤ËÀßÄê¤¹¤ëÁ°¤Ë¡¢dhclient-script ¤Ï²¿¤é¤«¤ÎÊýË¡¤Ç ¤½¤Î¥¢¥É¥ì¥¹¤ËÂÐ¤·¤Æ ARP ¤ò¹Ô¤¤¡¢ÊÖ»ö¤ò¼õ¤±¼è¤Ã¤¿¾ì¹ç¤Ë¤ÏÈó 0 ¤ÎÃÍ¤Ç ½ªÎ»¤¹¤ë¤Ù¤¤Ç¤¹¡£¤³¤Î¾ì¹ç¥¯¥é¥¤¥¢¥ó¥È¤Ï DHCPDECLINE ¥á¥Ã¥»¡¼¥¸¤ò¥µ¡¼¥Ð ¤ËÁ÷¿®¤·¡¢°ã¤¦¥¢¥É¥ì¥¹¤ò¼èÆÀ¤·¤Þ¤¹¡£ ¤³¤Î°î¶¤È¤Ï RENEW, REBIND, REBOOT ¾õÂÖ¤Ç¤â±ÍÍ¤Ë¹Ô¤¤¤Þ¤¹¤¬¡¢ É¬¤º¤·¤âÉ¬Í×¤Ç¤Ï¤Ê¤¯¡¢¼Â°Ý¹¥Þ¤·¤¯¤Ê¤¤¤Ç¤·¤ç¤¦¡£

·ë¹ç¤¬´°Î»¤¹¤ë¤È¡¢ ¥Í¥Ã¥È¥ï¡¼¥¯¤Ë´Ø¤¹¤ëÂ¿¤¯¤Î¥Ñ¥é¥á¡¼¥¿¤òÀßÄê¤¹¤ëÉ¬Í×¤¬¤¢¤ë¤Ç¤·¤ç¤¦¡£ $new_domain_name ¤ª¤è¤Ó $new_domain_name_servers (¤³¤ì¤ÏÊ£¿ô¤Î¥µ¡¼¥Ð¤ò¶õÇò¤Ç¶èÀÚ¤Ã¤ÆÎóµó¤·¤Æ¤¢¤ë¤«¤â¤·¤ì¤Þ¤»¤ó) ¤ò»ÈÍÑ¤·¤Æ¡¢ ¿·¤·¤¤ /etc/resolv.conf ¤òºîÀ®¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ¥Ç¥Õ¥©¥ë¥È·Ð¥©¤Ï¡¢$new_routers ¤ò»ÈÍÑ¤·¤ÆÀßÄê¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ÀÅÅª·Ð¥©¤Ï¡¢$new_static_routes ¤ò»ÈÍÑ¤·¤ÆÀßÄê¤¹¤ëÉ¬Í×¤¬¤¢¤ë¤«¤â¤·¤ì¤Þ¤»¤ó¡£

IP ¥¨¥¤¥ê¥¢¥¹¤¬Àë¸À¤µ¤ì¤Æ¤¤¤ë¾ì¹ç¡¢¤³¤³¤ÇÀßÄê¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ¥¨¥¤¥ê¥¢¥¹¤Î IP ¥¢¥É¥ì¥¹¤Ï $alias_ip_address ¤È¤·¤Æµ½Ò¤µ¤ì¡¢ ¥¨¥¤¥ê¥¢¥¹ÍÑ¤ËÀßÄê¤µ¤ì¤ëÂ¾¤Î DHCP ¥ª¥×¥·¥ç¥ó (Îã¤¨¤Ð¥µ¥Ö¥Í¥Ã¥È¥Þ¥¹¥¯) ¤Ï Á°½Ò¤Î¤è¤¦¤ËÊÑ¿ô¤ÇÅÏ¤µ¤ì¤Þ¤¹¤¬¡¢ $new_ ¤Ç³«»Ï¤¹¤ë¤Î¤Ç¤Ï¤Ê¤¯ $alias_ ¤Ç³«»Ï¤·¤Þ¤¹¡£ ¥¨¥¤¥ê¥¢¥¹¤Î IP ¥¢¥É¥ì¥¹¤¬·ë¹ç¤µ¤ì¤¿ IP ¥¢¥É¥ì¥¹ ($new_ip_address) ¤È Æ±¤¸¾ì¹ç¡¢¤³¤ì¤ò»ÈÍÑ¤·¤Æ¤Ï¤Ê¤é¤Ê¤¤¤³¤È¤ËÃí°Õ¤·¤Æ¤¯¤À¤µ¤¤¡£ ¤Ê¤¼¤Ê¤é¡¢¤³¤Î¾ì¹ç¤Ë¤ÏÂ¾¤Î¥¨¥¤¥ê¥¢¥¹¤Î¥Ñ¥é¥á¡¼¥¿¤¬Àµ¤·¤¯¤Ê¤¤²ÄÇ½À¤¬¤¢¤ë ¤«¤é¤Ç¤¹¡£

RENEW

·ë¹ç¤¬¹¹¿·¤µ¤ì¤ë¤È¡¢¥¹¥¯¥ê¥×¥È¤Ï BOUND ¤È¤±Í¤Ë¸Æ¤Ð¤ì¤Þ¤¹¤¬¡¢ $new_ ¤Ç³«»Ï¤¹¤ëÁ´ÊÑ¿ô¤Ë²Ã¤¨¤Æ $old ¤Ç³«»Ï¤¹¤ëÊÌ¤ÎÊÑ¿ô¤ÎÁÈ¤¬¤¢¤ë¤È¤¤¤¦ Îã³°¤¬¤¢¤ê¤Þ¤¹¡£ ÊÑ¹¹¤µ¤ì¤¿²ÄÇ½À¤¬¤¢¤ë±Ê³ª¤ÊÀßÄê¤Ï¡¢º¹½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ Îã¤¨¤Ð¡¢·ë¹ç¤µ¤ì¤¿¥¢¥É¥ì¥¹¤ËÂÐ¤¹¤ë¥í¡¼¥«¥ë·Ð¥©¤¬ÀßÄê¤µ¤ì¤¿¾ì¹ç¡¢ ¸Å¤¤¥í¡¼¥«¥ë·Ð¥©¤òº¹½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ¥Ç¥Õ¥©¥ë¥È·Ð¥©¤¬ÊÑ¹¹¤µ¤ì¤¿¾ì¹ç¡¢¸Å¤¤¥Ç¥Õ¥©¥ë¥È·Ð¥©¤òº¹½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ÀÅÅª·Ð¥©¤¬ÊÑ¹¹¤µ¤ì¤¿¾ì¹ç¡¢¸Å¤¤¤â¤Î¤òº¹½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ¤½¤ÎÂ¾¤Ë¤Ä¤¤¤Æ¤Ï¡¢BOUND ¤ÈÆ±ÍÍ¤Ë½èÍý²ÄÇ½¤Ç¤¹¡£

REBIND

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤¬¡¢¿·¤¬ DHCP ¥µ¡¼¥Ð¤ËºÆ·ë¹ç¤µ¤ì¤Þ¤·¤¿¡£ ¤³¤ì¤Ï RENEW ¤ÈÆ±ÍÍ¤Ë°·¤¨¤Þ¤¹¤¬¡¢IP ¥¢¥É¥ì¥¹¤¬ÊÑ¤ï¤Ã¤¿¾ì¹ç¤Ë¤Ï ARP É½¤ò¥¯¥ê¥¢¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£

REBOOT

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢¥ê¥Ö¡¼¥È¸å¤Ë¸µ¤Î¥¢¥É¥ì¥¹¤ò°Æ³ÍÆÀ¤¹¤ë¤³¤È¤ËÀ®¸ù¤·¤Þ¤·¤¿¡£ ¤³¤ì¤Ï BOUND ¤ÈÆ±ÍÍ¤Ë½èÍý²ÄÇ½¤Ç¤¹¡£

EXPIRE

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¥ê¡¼¥¹¤¹¹¿·¤¤Ë¼ºÇÔ¤·¡¢¥ê¡¼¥¹¤¬´ü¸Â¤¬ÀÚ¤ì¤Þ¤·¤¿¡£ ÂÐ¾Ý IP ¥¢¥É¥ì¥¹¤ò²òÊü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¡¢ RENEW ¤ª¤è¤Ó REBIND ¤ÈÆ±ÍÍ¤Ë¡¢´Ø¢¤¹¤ë¥Ñ¥é¥á¡¼¥¿¤òº¹½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£

FAIL

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï DHCP ¥µ¡¼¥Ð¤ËÀÜÂ³¤Ç¤¤º¡¢ ¤Þ¤¿¸¡ºº¤·¤¿ IP ¥¢¥É¥ì¥¹¤ËÏÍ¸ú¤Ê¤â¤Î¤Ï¤¢¤ê¤Þ¤»¤ó¤Ç¤·¤¿¡£ ºÇ¸å¤Ë¸¡ºº¤·¤¿¥ê¡¼¥¹¤Î¥Ñ¥é¥á¡¼¥¿¤Ï¡¢ÀßÄê²ò½ü¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£ ¤³¤ì¤Ï¡¢EXPIRE ¤ÈÆ±ÍÍ¤Ë°·¤¨¤Þ¤¹¡£

TIMEOUT

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¤É¤Î DHCP ¥µ¡¼¥Ð¤Ë¤âÀÜÂ³¤Ç¤¤Þ¤»¤ó¤Ç¤·¤¿¡£ ¤·¤«¤·¤Ê¤¬¤é¡¢¸Å¤¤¥ê¡¼¥¹¤¬¼±ÊÌ¤µ¤ì¡¢ BOUND ¤ÈÆ±ÍÍ¤Ë¡¢¤³¤Î¸Å¤¤¥ê¡¼¥¹¤Î¥Ñ¥é¥á¡¼¥¿¤¬ÅÏ¤µ¤ì¤Þ¤·¤¿¡£ ¥¯¥é¥¤¥¢¥ó¥È¤ÎÀßÄê¥¹¥¯¥ê¥×¥È¤Ï¡¢¤³¤Î¥Ñ¥é¥á¡¼¥¿¤ò¸¡ºº¤·¡¢ ¤³¤ì¤¬Í¸ú¤Ç¤¢¤ë¤È¿®¤¸¤ëÍýÍ³¤¬¤¢¤ë¤Ê¤é¤Ð¡¢ÃÍ 0 ¤Ç½ªÎ»¤¹¤Ù¤¤Ç¤¹¡£ ¤½¤¦¤Ç¤Ê¤¤¤Ê¤é¤Ð¡¢Èó 0 ¤ÎÃÍ¤Ç½ªÎ»¤¹¤Ù¤¤Ç¤¹¡£

¥ê¡¼¥¹¤ò¸¡ºº¤¹¤ëÄÌ¾ï¤ÎÊýË¡¤Ï¡¢REBIND ¤ÈÆ±ÍÍ¤Ë¥Í¥Ã¥È¥ï¡¼¥¯¤òÀßÄê¤·¤Æ (Ê£¿ô¤Î¥ê¡¼¥¹¤ò¸¡ºº¤¹¤ë¤¿¤á¤Ë¸Æ¤Ð¤ì¤ë¤³¤È¤¬¤¢¤ë¤«¤é¤Ç¤¹)¡¢ $routers ¤ÇÄêµÁ¤µ¤ì¤ëºÇ½é¤Î¥ë¡¼¥¿¤Ë ping ¤¹¤ë¤³¤È¤Ç¤¹¡£ ±þÅú¤ò¼õ¿®¤·¤¿¾ì¹ç¡¢¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤¬¸½ºßÀÜÂ³¤µ¤ì¤Æ¤¤¤ë¥Í¥Ã¥È¥ï¡¼¥¯¤ËÂÐ¤·¤Æ¡¢¥ê¡¼¥¹¤¬Í¸ú¤Ç¤¹¡£ $new_static_routers ¤Ë²Ã¤¨¤Æ $new_routers ¤ËÎóµó¤µ¤ì¤Æ¤¤¤ëÁ´¥ë¡¼¥¿¤Ë ping ¤ò»î¤¹¤è¤¦¤Ë¤Ê¤ì¤Ð¡¢ ´°Á´À¤¬Áý¤¹¤Ç¤·¤ç¤¦¡£¤·¤«¤·¡¢¸½ºß¤Î¥¹¥¯¥ê¥×¥È¤Ï¤½¤¦¤Ê¤Ã¤Æ¤¤¤Þ¤»¤ó¡£

´ØÏ¢¥Õ¥¡¥¤¥ë

Îà»÷¤·¤¿¥ª¥Ú¥ì¡¼¥Æ¥£¥ó¥°¥·¥¹¥Æ¥à¤ËÂÐ¤¹¤ë¥¹¥¯¥ê¥×¥È¥Õ¥¡¥¤¥ë¤Ï »÷¤Æ¤¤¤¿¤êÁ´¤¯Æ±¤¸¤«¤â¤·¤ì¤Þ¤»¤ó¤¬¡¢°ìÈÌ¤Ë¤Ï¡¢ ³Æ¥ª¥Ú¥ì¡¼¥Æ¥£¥ó¥°¥·¥¹¥Æ¥àÍÑ¤Ë³Æ¡¹¤Î¥¹¥¯¥ê¥×¥È¥Õ¥¡¥¤¥ë¤¬¤¢¤ë¤Ù¤¤Ç¤¹¡£ Internet Systems Consortium ¤Î DHCP ÇÛÉÛ¤Ë´Þ¤Þ¤ì¤ë¥¹¥¯¥ê¥×¥È¥Õ¥¡¥¤¥ë¤Ï¡¢ client/scripts °Ê²¼¤ÎÇÛÉÛ¥Ä¥ê¡¼¤Ë¤¢¤ê¡¢ Æ°ºîÂÐ¾Ý¥ª¥Ú¥ì¡¼¥Æ¥£¥ó¥°¥·¥¹¥Æ¥àÌ¾¤Ë¤Ê¤Ã¤Æ¤¤¤Þ¤¹¡£

¥Ð¥°

Ê£¿ô¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ò»ÈÍÑ¤¹¤ë¾ì¹ç¡¢ ¥µ¡¼¥Ð¤¬Äó¶¡¤¹¤ëÀßÄê¥Ñ¥é¥á¡¼¥¿¤ÆÆ±»Î¤¬ ¾×ÆÍ¤·¤Ê¤¤¤è¤¦¤Ë¤¹¤ëÌÀ³Î¤ÊÊýË¡¤Ï¤Ê¤¤¤¢¤ê¤Þ¤»¤ó¡£ Îã¤¨¤Ð¡¢ É¸½à¤Î dhclient-script ¤Ï /etc/resolv.conf ¤òºÆÅÙ½ñ¤¹¤¨¤Æ¤·¤Þ¤¤¤Þ¤¹¡£ ¤¹¤Ê¤ï¤Á¡¢Ê£¿ô¤Î¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤¬ÀßÄê¤µ¤ì¤Æ¤¤¤ë¾ì¹ç¡¢ ¤¢¤ë¥µ¡¼¥Ð¤«¤éÄó¶¡¤µ¤ì¤ëÃÍ¤Ë /etc/resolv.conf ¤¬½é´ü²½¤µ¤ì¤¿¸å¤Ë¡¢ ÊÌ¤Î¥µ¡¼¥Ð¤«¤éÄó¶¡¤µ¤ì¤ëÃÍ¤Ë½é´ü²½¤µ¤ì¤ë¤È¤¤¤¦¦Æ°ºî¤ò·«¤êÊÖ¤·¤Þ¤¹¡£ ¤É¤Á¤é¤Î¥µ¡¼¥Ð¤«¤éÄó¶¡¤µ¤ì¤ë¾ðÊó¤âÍ¸ú¤Ç¤¢¤ë¾ì¹ç¤Ë¤Ï¡¢ ¼Â²Ý¾å¤ÏÌäÂê¤È¤Ê¤é¤Ê¤¤¤Ç¤·¤ç¤¦¤¬¡¢º®Íð¤Î¸¶°ø¤È¤Ê¤ê¤¨¤Þ¤¹¡£

´ØÏ¢¹àÌÜ

dhclient.conf(5), dhclient.leases(5), dhclient(8)

ºî¼Ô

dhclient-script(8) ¤Ï Ted Lemon ¤¬ Vixie Enterprises ¤È¶¨ÎÏ¤·¤Æ Internet Systems Consortium ¤Î¤¿¤á¤Ë ½ñ¤¤Þ¤·¤¿¡£ Internet Systems Consortium ¤Ë¤Ä¤¤¤Æ¤è¤ê¾Ü¤·¤¯¤Ï¡¢ **http://www.isc.org** ¤ò¸÷¤¯¤À¤µ¤¤¡£ Vixie Enterprises ¤Ë¤Ä¤¤¤Æ¤è¤ê¾Ü¤·¤¯¤Ï¡¢**http://www.vix.com** ¤ò¸÷¤¯¤À¤µ¤¤¡£

NAME

dhclient - Dynamic Host Configuration Protocol (DHCP) Client

SYNOPSIS

dhclient [**-p** *port*] [**-d**] [**-q**] [**-1**] [**-o**] [**-r**] [**-lf** *lease-file*] [**-pf** *pid-file*] [**-cf** *config-file*] [**-sf** *script-file*] [**-s** server] [**-g** relay] [**-n**] [**-nw**] [**-w**] [*if0* [*...ifN*]]

DESCRIPTION

The Internet Systems Consortium DHCP Client, dhclient, provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol, BOOTP protocol, or if these protocols fail, by statically assigning an address.

SYSTEM REQUIREMENTS

You must have the Berkeley Packet Filter (bpf) configured in your NetBSD kernel.

OPERATION

The DHCP protocol allows a host to contact a central server which maintains a list of IP addresses which may be assigned on one or more subnets. A DHCP client may request an address from this pool, and then use it on a temporary basis for communication on network. The DHCP protocol also provides a mechanism whereby a client can learn important details about the network to which it is attached, such as the location of a default router, the location of a name server, and so on.

On startup, dhclient reads the *dhclient.conf* for configuration instructions. It then gets a list of all the network interfaces that are configured in the current system. For each interface, it attempts to configure the interface using the DHCP protocol.

In order to keep track of leases across system reboots and server restarts, dhclient keeps a list of leases it has been assigned in the dhclient.leases(5) file. On startup, after reading the dhclient.conf file, dhclient reads the dhclient.leases file to refresh its memory about what leases it has been assigned.

When a new lease is acquired, it is appended to the end of the dhclient.leases file. In order to prevent the file from becoming arbitrarily large, from time to time dhclient creates a new dhclient.leases file from its in-core lease database. The old version of the dhclient.leases file is retained under the name *dhclient.leases˜* until the next time dhclient rewrites the database.

Old leases are kept around in case the DHCP server is unavailable when dhclient is first invoked (generally during the initial system boot process). In that event, old leases from the dhclient.leases file which have not yet expired are tested, and if they are determined to be valid, they are used until either they expire or the DHCP server becomes available.

A mobile host which may sometimes need to access a network on which no DHCP server exists may be preloaded with a lease for a fixed address on that network. When all attempts to contact a DHCP server have failed, dhclient will try to validate the static lease, and if it succeeds, will use that lease until it is restarted.

A mobile host may also travel to some networks on which DHCP is not available but BOOTP is. In that case, it may be advantageous to arrange with the network administrator for an entry on the BOOTP database, so that the host can boot quickly on that network rather than cycling through the list of old leases.

COMMAND LINE

The names of the network interfaces that dhclient should attempt to configure may be specified on the command line. If no interface names are specified on the command line dhclient will normally identify all network interfaces, eliminating non-broadcast interfaces if possible, and attempt to configure each interface.

It is also possible to specify interfaces by name in the **dhclient.conf(5)** file. If interfaces are specified in this way, then the client will only configure interfaces that are either specified in the configuration file or on the command line, and will ignore all other interfaces.

If the DHCP client should listen and transmit on a port other than the standard (port 68), the **-p** flag may used. It should be followed by the udp port number that dhclient should use. This is mostly useful for debugging purposes. If a different port is specified for the client to listen on and transmit on, the client will also use a different destination port - one greater than the specified destination port.

The DHCP client normally transmits any protocol messages it sends before acquiring an IP address to, 255.255.255.255, the IP limited broadcast address. For debugging purposes, it may be useful to have the server transmit these messages to some other address. This can be specified with the **-s** flag, followed by the IP address or domain name of the destination.

For testing purposes, the giaddr field of all packets that the client sends can be set using the **-g** flag, followed by the IP address to send. This is only useful for testing, and should not be expected to work in any consistent or useful way.

The DHCP client will normally run in the foreground until it has configured an interface, and then will revert to running in the background. To run force dhclient to always run as a foreground process, the **-d** flag should be specified. This is useful when running the client under a debugger, or when running it out of inittab on System V systems.

The client normally prints a startup message and displays the protocol sequence to the standard error descriptor until it has acquired an address, and then only logs messages using the **syslog (3)** facility. The **-q** flag prevents any messages other than errors from being printed to the standard error descriptor.

The client normally doesn't release the current lease as it is not required by the DHCP protocol. Some cable ISPs require their clients to notify the server if they wish to release an assigned IP address. The **-r** flag explicitly releases the current lease, and once the lease has been released, the client exits.

The **-1** flag cause dhclient to try once to get a lease. If it fails, dhclient exits with exit code two.

The **-o** flag cause dhclient to assume that it's been given a fixed lease, so once it installs the lease, it exits. This is really only useful on very small systems, and only works on a single interface at a time - if you want it to support multiple interfaces, run dhclient on each interface in succession.

The DHCP client normally gets its configuration information from **/etc/dhclient.conf,** its lease database from **/var/db/dhclient.leases,** stores its process ID in a file called **/var/run/dhclient.pid,** and configures the network interface using **/sbin/dhclient-script** To specify different names and/or locations for these files, use the **-cf, -lf, -pf** and **-sf** flags, respectively, followed by the name of the file. This can be particularly useful if, for example, **/var/db** or **/var/run** has not yet been mounted when the DHCP client is started.

The DHCP client normally exits if it isn't able to identify any network interfaces to configure. On laptop computers and other computers with hot-swappable I/O buses, it is possible that a broadcast interface may be added after system startup. The **-w** flag can be used to cause the client not to exit when it doesn't find any such interfaces. The **omshell (1)** program can then be used to notify the client when a network interface has been added or removed, so that the client can attempt to configure an IP address on that interface.

The DHCP client can be directed not to attempt to configure any interfaces using the **-n** flag. This is most likely to be useful in combination with the **-w** flag.

The client can also be instructed to become a daemon immediately, rather than waiting until it has acquired an IP address. This can be done by supplying the **-nw** flag.

CONFIGURATION

The syntax of the dhclient.conf(5) file is discussed separately.

OMAPI

The DHCP client provides some ability to control it while it is running, without stopping it. This capability is provided using OMAPI, an API for manipulating remote objects. OMAPI clients connect to the client using TCP/IP, authenticate, and can then examine the client's current status and make changes to it.

Rather than implementing the underlying OMAPI protocol directly, user programs should use the dhcpctl API or OMAPI itself. Dhcpctl is a wrapper that handles some of the housekeeping chores that OMAPI does not do automatically. Dhcpctl and OMAPI are documented in **dhcpctl(3)** and **omapi(3)**. Most things you'd want to do with the client can be done directly using the **omshell(1)** command, rather than having to write a special program.

THE CONTROL OBJECT

The control object allows you to shut the client down, releasing all leases that it holds and deleting any DNS records it may have added. It also allows you to pause the client - this unconfigures any interfaces the client is using. You can then restart it, which causes it to reconfigure those interfaces. You would normally pause the client prior to going into hibernation or sleep on a laptop computer. You would then resume it after the power comes back. This allows PC cards to be shut down while the computer is hibernating or sleeping, and then reinitialized to their previous state once the computer comes out of hibernation or sleep.

The control object has one attribute - the state attribute. To shut the client down, set its state attribute to 2. It will automatically do a DHCPRELEASE. To pause it, set its state attribute to 3. To resume it, set its state attribute to 4.

FILES

/sbin/dhclient-script, /etc/dhclient.conf, /var/db/dhclient.leases, /var/run/dhclient.pid, /var/db/dhclient.leases˜.

SEE ALSO

dhcpd(8), dhcrelay(8), dhclient-script(8), dhclient.conf(5), dhclient.leases(5).

AUTHOR

dhclient(8) has been written for Internet Systems Consortium by Ted Lemon in cooperation with Vixie Enterprises. To learn more about Internet Systems Consortium, see **http://www.isc.org** To learn more about Vixie Enterprises, see **http://www.vix.com.**

This client was substantially modified and enhanced by Elliot Poger for use on Linux while he was working on the MosquitoNet project at Stanford.

The current version owes much to Elliot's Linux enhancements, but was substantially reorganized and partially rewritten by Ted Lemon so as to use the same networking framework that the Internet Systems Consortium DHCP server uses. Much system-specific configuration code was moved into a shell script so that as support for more operating systems is added, it will not be necessary to port and maintain system-specific configuration code to these operating systems - instead, the shell script can invoke the native tools to accomplish the same purpose.

Ì¾¾Î

dhclient - Æ°Åª¥Û¥¹¥ÈÀßÄê¥×¥í¥È¥³¥ë¤Î¥¯¥é¥¤¥¢¥ó¥È

½ñ¼°

dhclient [**-p** *port*] [**-D**] [**-d**] [**-q**] [**-1**] [**-r**] [**-lf lease-file**] [**-pf** *pid-file*] [**-cf** *config-file*] [**-sf** *script-file*] [**-s** server] [**-g** relay] [**-n**] [**-nw**] [**-w**] [*if0* [*...ifN*]]

²òÀâ

Internet Systems Consortium ¤Î DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ç¤¢¤ë dhclient ¤ÏÆ°Åª¥Û¥¹¥ÈÀßÄê¥×¥í¥È¥³¥ë (DHCP: Dynamic Host Configuration Protocol) ¤Þ¤¿¤Ï BOOTP ¥×¥í¥È¥³¥ë¤òÍÑ¤¤¤Æ¡¢¤¢¤ë¤¤¤Ï ¤³¤ì¤é¤Î¥×¥í¥È¥³¥ë¤¬¼ºÇÔ¤·¤¿¾ì¹ç¤Ë¤Ï¥¢¥É¥ì¥¹¤òÀÅÅª¤Ë³ä¤êÅö¤Æ¤Æ¡¢ 1 ¤Ä°Ê¾å¤Î¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òÀßÄê¤¹¤ë¼êÃÊ¤òÄó¶¡¤·¤Þ¤¹¡£

Áàºî

DHCP ¥×¥í¥È¥³¥ë¤Ç¤Ï¡¢1 ¤Ä°Ê¾å¤Î¥µ¥Ö¥Í¥Ã¥È¤Ë³ä¤êÅö¤Æ¤ë¤³¤È¤Î¤Ç¤¤ë IP ¥¢¥É¥ì¥¹¤Î¥ê¥¹¥È¤ò´ÉÍý¤¹¤ëÃæ±û¥µ¡¼¥Ð¤Ë¡¢¥Û¥¹¥È¤¬¥¢¥¯¥»¥¹¤Ç¤¤Þ¤¹¡£ DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¤³¤Î¥ê¥¹¥È¤«¤é¥¢¥É¥ì¥¹¤òÍ×µá¤·¤Æ¡¢ ¤½¤ì¤ò¥Í¥Ã¥È¥ï¡¼¥¯ÄÌ¿®¤Î°ì»þÅª¤ÊÚÂæ¤ËÍÑ¤¤¤ë¤³¤È¤¬¤Ç¤¤Þ¤¹¡£ ¤Þ¤¿ DHCP ¥×¥í¥È¥³¥ë¤Ï¡¢¥Ç¥Õ¥©¥ë¥È¥ë¡¼¥¿¤Î¾ì½ê¤ä¥Í¡¼¥à¥µ¡¼¥Ð¤Î¾ì½ê¤Ê¤É¡¢ ¥¯¥é¥¤¥¢¥ó¥È¤¬ÀÜÂ³¤·¤Æ¤¤¤ë¥Í¥Ã¥È¥ï¡¼¥¯¤Ë´Ø¤¹¤ë½ÅÍ×¤Ê¾ðÊó¤ò ¥¯¥é¥¤¥¢¥ó¥È¤Ë¾ÜºÙ¤ËÃÎ¤é¤»¤ë»ëµ¡¹½¤âÄó¶¡¤·¤Þ¤¹¡£

µ¯Æ°»þ¤Ë dhclient ¤Ï *dhclient.conf* ¤«¤éÀßÄê»Ø¼¨¤òÆÉ¤ß¹þ¤ê¤Þ¤¹¡£ ¤½¤ì¤«¤é¸½ºß¤Î¥·¥¹¥Æ¥à¤ËÁÈ¤ß¹þ¤Þ¤ì¤Æ¤¤¤ë ¤¹¤Ù¤Æ¤Î¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤Î¥ê¥¹¥È¤ò¼èÆÀ¤·¤Þ¤¹¡£ ³Æ¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ËÂÐ¤· dhclient ¤Ï DHCP ¥×¥í¥È¥³¥ë¤òÍÑ¤¤¤ÆÀßÄê¤ò»î¤ß¤Þ¤¹¡£

¥·¥¹¥Æ¥à¥ê¥Ö¡¼¥È¤ä¥µ¡¼¥Ð°Æµ¯Æ°¤Î°Ý¤Ë¥ê¡¼¥¹¤ò¼º¤ï¤Ê¤¤¤¿¤á¤¦¤Ë¡¢ dhclient ¤Ï³ä¤êÅö¤Æ¤é¤ì¤¿¥ê¡¼¥¹¤Î¥ê¥¹¥È¤ò dhclient.leases(5) ¥Õ¥¡¥¤¥ë¤ËÊÝÂ¸¤·¤Þ¤¹¡£ µ¯Æ°»þ¡¢dhclient.conf ¥Õ¥¡¥¤¥ë¤òÆÉ¤ß¹þ¤ó¤À¸å¡¢ dhclient ¤Ï dhclient.leases ¥Õ¥¡¥¤¥ë¤òÆÉ¤ß¹þ¤ó¤Ç¡¢ ³ä¤êÅö¤Æ¤é¤ì¤¿¥ê¡¼¥¹¤Ë´Ø¤¹¤ë¥á¥â¥ê¤ò¹¿¿·¤·¤Þ¤¹¡£

¿·¤·¤¤¥ê¡¼¥¹¤ò¼èÆÀ¤¹¤ë¤È¡¢dhclient.leases ¥Õ¥¡¥¤¥ë¤ÎËöÈø¤ËÉÕ¤±²Ã¤¨¤é¤ì¤Þ¤¹¡£ ¥Õ¥¡¥¤¥ë¤¬¶ËÃ¼¤ËÂç¤¤¯¤Ê¤ë¤Î¤òËÉ¤°¤¿¤á¤Ë¡¢ dhclient ¤Ï»þ¤ª¤ê¥³¥¢¥Æ¡¼¥Ö¥ë¥Ç¡¼¥¿¥Ù¡¼¥¹¤«¤é ¿·µ¬¤Ë dhclient.leases ¥Õ¥¡¥¤¥ë¤òºÎ®¤·¤Þ¤¹¡£ ¸Å¤¤ dhclient.leases ¥Õ¥¡¥¤¥ë¤Ï¡¢ dhclient ¤¬¼¡¤Ë¥Ç¡¼¥¿¥Ù¡¼¥¹¤òºîÀ®¤¹¤ë¤Þ¤Ç¡¢ *dhclient.leases~* ¤È¤¤¤¦Ì¾Á°¤ÇÊÝÂ¸¤µ¤ì¤Þ¤¹¡£

dhclient ¤¬ºÇ½é¤Ëµ¯Æ°¤µ¤ì¤¿¤È¤ (°ìÈÌÅª¤Ë¤Ï¥·¥¹¥Æ¥à¥Ö¡¼¥È½é´ü²áÄø¤Î´Ö) ¤Ë DHCP ¥µ¡¼¥Ð¤¬ÍøÍÑ¤Ç¤¤Ê¤±¤ì¤Ð¡¢ ¸Å¤¤¥ê¡¼¥¹¤Ï»Ä¤µ¤ì¤Þ¤¹¡£ ¤½¤Î¾ì¹ç¡¢dhclient.leases ¥Õ¥¡¥¤¥ë¤«¤é ¤Þ¤À´ü¸Â¤ÎÀÚ¤ì¤Æ¤¤¤Ê¤¤¸Å¤¤¥ê¡¼¥¹¤ò¸¡ºº¤·¡¢ Í¸ú¤Ç¤¢¤ë¤ÈÈ½ÃÇ¤µ¤ì¤ì¤Ð¡¢¤½¤ì¤é¤Î´ü¸Â¤¬ÀÚ¤ì¤ë¤« ¤Þ¤¿¤Ï DHCP ¥µ¡¼¥Ð¤¬ÍøÍÑ¤Ç¤¤ë¤è¤¦¤Ë¤Ê¤ë¤Þ¤Ç¡¢¤½¤Î¥ê¡¼¥¹¤ò»È¤¤¤Þ¤¹¡£

DHCP ¥µ¡¼¥Ð¤¬Â¸ºß¤·¤Ê¤¤¥Í¥Ã¥È¥ï¡¼¥¯¤Ë»þ¤ª¤ê¥¢¥¯¥»¥¹¤¹¤ëÉ¬Í×¤¬ ¤¢¤ë¤è¤¦¤Ê°ÜÆ°¥Û¥¹¥È¤Ï¡¢¤½¤Î¥Í¥Ã¥È¥ï¡¼¥¯¾å¤Î¸ÇÄê¥¢¥É¥ì¥¹¤Î¥ê¡¼¥¹¤ò ¤¢¤é¤«¤¸¤áÆÉ¤ß¹þ¤ó¤Ç¤ª¤¯¤³¤È¤¬¤Ç¤¤Þ¤¹¡£ DHCP ¥µ¡¼¥Ð¤Ø¤Î¥¢¥¯¥»¥¹¤¬¤É¤ì¤âÀ®¸ù¤·¤Ê¤«¤Ã¤¿¾ì¹ç¡¢ dhclient ¤Ï¤½¤ÎÀÅÅª¤Ê¥ê¡¼¥¹¤¬Í¸ú¤Ç¤¢¤ë¤«¸¡¾Ú¤·¡¢ Í¸ú¤Ç¤¢¤ì¤Ð¼¡¤Ë°Æµ¯Æ°¤µ¤ì¤ë¤Þ¤Ç¤½¤Î¥ê¡¼¥¹¤ò»È¤¤¤Þ¤¹¡£

¤Þ¤¿°ÜÆ°¥Û¥¹¥È¤Ï¡¢DHCP ¤ÏÍøÍÑ¤Ç¤¤Ê¤¤¤¬ BOOTP ¤Ê¤éÍøÍÑ¤Ç¤¤ë¤è¤¦¤Ê ¥Í¥Ã¥È¥ï¡¼¥¯¤Ø°ÜÆ°¤¹¤ë¤³¤È¤â¤¢¤ë¤Ç¤·¤ç¤¦¡£ ¤½¤Î¤è¤¦¤Ê¾ì¹ç¤Ï¡¢¸Å¤¤¥ê¡¼¥¹¤ò½ç¼¡»î¤¹¤è¤ê¤â¡¢ ¤½¤Î¥Í¥Ã¥È¥ï¡¼¥¯¤Î´ÉÍý¼Ô¤ÈÁêÃÌ¤·¤Æ BOOTP ¥Ç¡¼¥¿¥Ù¡¼¥¹¤Ë¥¨¥ó¥È¥ê¤òºîÀ®¤·¤Æ¤â¤é¤¤¡¢ ¤½¤Î¥Í¥Ã¥È¥ï¡¼¥¯¾å¤ÇÁÇÁá¤¯¥Ö¡¼¥È¤Ç¤¤ë¤è¤¦¤Ë¤¹¤ë¤È¤¤¤¤¤Ç¤·¤ç¤¦¡£

¥³¥Þ¥ó¥É¥é¥¤¥ó

dhclient ¤¬ÀßÄê¤·¤è¤¦¤È¤¹¤ë¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤Î¤Î¾Á°¤ò ¥³¥Þ¥ó¥É¥é¥¤¥ó¤Ç»ØÄê¤Ç¤¤Þ¤¹¡£ ¥³¥Þ¥ó¥É¥é¥¤¥ó¤Ç¥¤¥ó¥¿¥Õ¥§¡¼¥¹Ì¾¤¬»ØÄê¤µ¤ì¤Ê¤±¤ì¤Ð¡¢ dhclient ¤Ï¤¹¤Ù¤Æ¤Î¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ò¼±ÊÌ¤·¡¢ ²ÄÇ½¤Ê¤é¤ÐÈó¥Ö¥í¡¼¥É¥¥ã¥¹¥È¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤Ï½ü¤¤¤Æ¡¢ ¤½¤ì¤¾¤ì¤Î¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òÀßÄê¤·¤è¤¦¤È¤·¤Þ¤¹¡£

dhclient.conf(5) ¥Õ¥¡¥¤¥ëÃæ¤ÎÌ¾Á°¤Ç¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ò»ØÄê¤¹¤ë¤³¤È¤â²ÄÇ½¤Ç¤¹¡£ ¤³¤ÎÊýË¡¤Ç¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ò»ØÄê¤·¤¿¾ì¹ç¡¢¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ ÀßÄê¥Õ¥¡¥¤¥ëÃæ¤Ç»ØÄê¤·¤¿¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤â·¤¯¤Ï¥³¥Þ¥ó¥É¹Ô¤Ç »ØÄê¤·¤¿¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ÎÉ¤Á¤é¤«¤À¤±¤òÀßÄê¤¹¤ë¤Ç¤·¤ç¤¦¡£

-D ¥Õ¥é¥°¤ò»ØÄê¤¹¤ë¤È¡¢ **dhclient** ¤¬ **dhclient-script** ¤ÈÁÈ¤ß¹ç¤ï¤»¤Æ»ÈÍÑ¤¹¤ë¤¿¤á¤ËºîÀ®¤·¤¿¥¹¥¯¥ê¥×¥È¤ò¡¢*/tmp* ¤ËÊÝÂ¸¤µ¤»¤Þ¤¹¡£

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤¬É¸½à¥Ý¡¼¥È (¥Ý¡¼¥ÈÈÖ¹æ 68) °Ê³°¤Î¥Ý¡¼¥È¤Ç ÂÔµ¡¤ª¤è¤ÓÁ÷¿®¤¹¤ëÉ¬Í×¤¬¤¢¤ë¾ì¹ç¤Ë¤Ï **-p** ¥Õ¥é¥°¤¬»È¤¨¤Þ¤¹¡£ ¤³¤Î¥Õ¥é¥°¤ËÂ³¤±¤Æ¡¢dhclient ¤¬»È¤¦ udp ¥Ý¡¼¥ÈÈÖ¹æ¤ò»ØÄê¤·¤Þ¤¹¡£ ¤³¤ì¤Ï¼ç¤È¤·¤Æ¥Ç¥Ð¥Ã¥°ÌÜÅª¤ÇÍÍÑ¤Ç¤¹¡£ ¥¯¥é¥¤¥¢¥ó¥È¤¬ÂÔµ¡¤ª¤è¤ÓÁ÷¿®¤¹¤ë¤¿¤á¤Ë»ÈÍÑ¤¹¤ë¥Ý¡¼¥È¤Ë ¥Ç¥Õ¥©¥ë¥È¤È¤Ï°ã¤¦¥Ý¡¼¥È¤ò»ØÄê¤¹¤ë¾ì¹ç¡¢¥¯¥é¥¤¥¢¥ó¥È¤Ï ¤â¤¦ 1 ¤ÄÊÌ¤ÎÁ÷¿®Àè¥Ý¡¼¥È¤â»ÈÍÑ¤·¤Þ¤¹¡£¤½¤ÎÁ÷¿®Àè¥Ý¡¼¥È¤Ï¡¢ »ØÄê¤·¤¿Á÷¿®Àè¥Ý¡¼¥È¤è¤ê¤âÂç¤¤ÊÈÖ¹æ¤ò»ý¤Ã¤¿¤â¤Î¤Ç¤¹¡£

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ÄÌ¾ï IP ¥¢¥É¥ì¥¹¤ò³ÍÆÀ¤·¤Æ¤¤¤Ê¤¤´Ö Ç¤°Õ¤Î¥×¥í¥È¥³¥ë¥á¥Ã¥»¡¼¥¸¤ò¥Ö¥í¡¼¥É¥¥ã¥¹¥È ¥¢¥É¥ì¥¹¤Ç¤¢¤ë 255.255.255.255 ¤Ø¤ÈÁ÷¿®¤·¤Þ¤¹¡£ ¥Ç¥Ð¥Ã¥°ÌÜÅª¤Ç¡¢¥µ¡¼¥Ð¤¬¤³¤ì¤é¤Î¥á¥Ã¥»¡¼¥¸¤ò¤É¤³¤«Â¾¤Î¥¢¥É¥ì¥¹¤Ø Á÷¿®¤·¤¿Êý¤¬ÊØÍø¤Ê¾ì¹ç¤¬¤¢¤ê¤Þ¤¹¡£ **-s** ¥Õ¥é¥°¤Î¸å¤ËÁ÷¿®Àè¤Î IP ¥¢¥É¥ì¥¹¤Þ¤¿¤Ï¥É¥á¥¤¥óÌ¾¤ò¤Ä¤±¤Æ»ØÄê ¤Ç¤¤Þ¤¹¡£ ¥Æ¥¹¥ÈÌÜÅª¤Ç¡¢DHCP ¥¯¥é¥¤¥¢¥ó¥È¤¬Á÷¿®¤¹¤ëÁ´¤Æ¤Î¥Ñ¥±¥Ã¥È¤Î giaddr ¥Õ¥£¡¼¥ë¥É¤ò **-g** ¥Õ¥é¥°¤ËÁ÷¿®Àè¤Î IP ¥¢¥É¥ì¥¹¤òÂ³¤±¤¿·Á¤ò»ÈÍÑ¤¹¤ë¤³¤È¤ÇÀßÄê¤¹¤ë ¤³¤È¤¬¤Ç¤¤Þ¤¹¡£¤³¤ì¤Ï¥Æ¥¹¥ÈÌÜÅª¤Î»þ¤Î¤ßÍÍÑ¤Ê¤â¤Î¤Ç¤¢¤ê¡¢ ·ø¼Â¤µ¤ä»È¤¤¤ä¤¹¤µ¤òµá¤á¤ë¾õ¶·¤ÇÆ°ºî¤¹¤ë¤³¤È¤òÁÛÄê¤·¤Æ¤Ï ¤¤¤±¤Þ¤»¤ó¡£

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ÄÌ¾ï¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òÀßÄê¤¹¤ë¤Þ¤Ç¤Ï ¥Õ¥©¥¢¥°¥é¥¦¥ó¥É¤ÇÆ°ºî¤·¡¢¤½¤Î¸å¥Ð¥Ã¥¯¥°¥é¥¦¥ó¥É¤ÇÆ°ºî ¤¹¤ë¤è¤¦¤Ë¤Ê¤ê¤Þ¤¹¡£dhclient ¤ò¾ï¤Ë¥Õ¥©¥¢¥°¥é¥¦¥ó¥É¤Î ¥×¥í¥»¥¹¤È¤·¤ÆÆ°ºî¤µ¤»¤ë¤¿¤á¤Ë¤Ï¡¢ **-d** ¥Õ¥é¥°¤ò»ØÄê¤¹¤ëÉ¬Í×¤¬¤¢¤ê¤Þ¤¹¡£¤³¤ì¤Ï¡¢DHCP ¥¯¥é¥¤¥¢¥ó¥È¤¬ ¥Ç¥Ð¥Ã¥¬¤Î²¼¤ÇÆ°ºî¤·¤Æ¤¤¤ë¾ì¹ç¤ä¡¢System V ¥·¥¹¥Æ¥à¤Î inittab ¤Î³°Â¦¤ÇÆ°ºî¤·¤Æ¤¤¤ë¾ì¹ç¤Ë¤ÏÍ¸ú¤Ê¤â¤Î¤Ç¤¹¡£

¤³¤Î¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ÄÌ¾ï¤Ïµ¯Æ°¥á¥Ã¥»¡¼¥¸¤òÉ½¼¨¤·¡¢¥¢¥É¥ì¥¹¤ò ³ÍÆÀ¤¹¤ë¤Þ¤Ç¤ÏÉ¸½à¥¨¥é¡¼½ÐÎÏ¤Ë¥×¥í¥È¥³¥ë¥·¡¼¥±¥ó¥¹¤ò ½ñ¤½Ð¤·¤Þ¤¹¡£¥¢¥É¥ì¥¹¤ò³ÍÆÀ¤·¤¿¸å¤Ï **syslog (3)** ¥Õ¥¡¥·¥ê¥Æ¥£¤ò»ÈÍÑ¤·¤Æ¥á¥Ã¥»¡¼¥¸¤Î¥í¥°¤ò¼è¤ë¤è¤¦¤Ë¤Ê¤ê¤Þ¤¹¡£ **-q** ¥Õ¥é¥°¤ò»ÈÍÑ¤¹¤ë¤È¡¢¥¨¥é¡¼°Ê³°¤Î¥á¥Ã¥»¡¼¥¸¤òÉ¸½à¥¨¥é¡¼½ÐÎÏ¤Ë ½ñ¤½Ð¤µ¤Ê¤¤¤è¤¦¤Ë¤Ê¤ê¤Þ¤¹¡£

¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢DHCP ¥×¥í¥È¥³¥ë¤ÇµÁÌ³¤Å¤±¤é¤ì¤Æ¤¤¤Ê¤¤¤¿¤á¡¢ ÄÌ¾ï¤Ï¸½ºß¼èÆÀ¤·¤Æ¤¤¤ë¥ê¡¼¥¹¤ò³«Êü¤·¤Þ¤»¤ó¡£ ¤¿¤À¡¢¥±¡¼¥Ö¥ë ISP ¤Ê¤É¤Ç¤Ï¡¢¥¯¥é¥¤¥¢¥ó¥È¤¬ ³äÅö¤Æ¤é¤ì¤¿IP ¥¢¥É¥ì¥¹¤ò³«Êü¤·¤¿¤¤¾ì¹ç¤Ë¤Ï¡¢¥µ¡¼¥Ð¤Ë ÄÌÃÎ¤¹¤ë¤³¤È¤òµÁÌ³¤Å¤±¤Æ¤¤¤ë¤È¤³¤í¤â¤¢¤ê¤Þ¤¹¡£ **-r** ¥Õ¥é¥°¤òÍÑ¤¤¤ë¤È¡¢ÌÀ¼¨Åª¤Ë¸½ºß¤Î¥ê¡¼¥¹¤ò³«Êü¤·¡¢¤¤¤Ã¤¿¤ó ¥ê¡¼¥¹¤ò³«Êü¤¹¤ë¤È¥¯¥é¥¤¥¢¥ó¥È¤Ï½ªÎ»¤·¤Þ¤¹¡£

-1 ¥Õ¥é¥°¤ò»ØÄê¤¹¤ë¤È¡¢ dhclient ¤Ï¤Ò¤È¤Ä¤Î¥ê¡¼¥¹¤ËÂÐ¤· 1 ÅÙ¤À¤±¤·¤«¼èÆÀ¤ò»î¤ß¤Þ¤»¤ó¡£ ¤â¤·¼èÆÀ¤Ë¼ºÇÔ¤¹¤ì¤Ð dhclient ¤Ï½ªÎ»¥³¡¼¥É 2 ¤Ç½ªÎ»¤·¤Þ¤¹¡£

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ÄÌ¾ï¤ÏÀßÄê¾ðÊó¤ò **ETCDIR/dhclient.conf** ¤«¤é¡¢¥ê¡¼¥¹¥Ç¡¼¥¿¥Ù¡¼¥¹¤ò **DBDIR/dhclient.leases** ¤«¤é¼èÆÀ¤·¡¢¼«Ê¬¤Î¥×¥í¥»¥¹ ID ¤ò **RUNDIR/dhclient.pid** ¤È¤¤¤¦Ì¾Á°¤Î¥Õ¥¡¥¤¥ë¤ËÊÝÂ¸¤·¡¢ ¤½¤·¤Æ¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ò **CLIENT-BINDIR/dhclient-script** ¤ò»ÈÍÑ¤·¤ÆÀßÄê¤·¤Þ¤¹¡£ ¤³¤ì¤é¤Î¥Õ¥¡¥¤¥ë¤ËÊÌ¤ÎÌ¾Á°¤ò»ØÄê¤·¤¿¤ê¡¢ÊÌ¤Î¾ì½ê¤ò »ØÄê¤·¤¿¤ê¤¹¤ë¤Ë¤Ï¡¢¤½¤ì¤¾¤ì **-cf, -lf, -pf** ¤ª¤è¤Ó **-sf** ¥Õ¥é¥°¤ò¡¢¸å¤í¤Ë¥Õ¥¡¥¤¥ëÌ¾¤òÂ³¤±¤ë·Á¤Ç»ÈÍÑ¤·¤Æ¤¯¤À¤µ¤¤¡£ ¤³¤ÎÊýË¡¤Ï¡¢Îã¤¨¤Ð DHCP ¥¯¥é¥¤¥¢¥ó¥È¤¬µ¯Æ°¤·¤¿¤È¤¤Ë **DBDIR** ¤â¤·¤¯¤Ï **RUNDIR** ¤¬¤Þ¤À¥Þ¥¦¥ó¥È¤µ¤ì¤Æ¤¤¤Ê¤¤¾ì¹ç¤Ë¤ÏÆÃ¤ËÍÍÑ¤Ê¤â¤Î¤Ë ¤Ê¤ê¤Þ¤¹¡£

DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢ÀßÄê¤¹¤Ù¤¥Í¥Ã¥È¥ï¡¼¥¯ ¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òÆ±Äê¤Ç¤¤Ê¤¤¾ì¹ç¡¢ÄÌ¾ï¤Ï½ªÎ»¤·¤Þ¤¹¡£ ¥é¥Ã¥×¥È¥Ã¥×¥³¥ó¥Ô¥å¡¼¥¿¤ä¥Û¥Ã¥È¥¹¥ï¥Ã¥×²ÄÇ½¤Ê I/O ¥Ð¥¹¤ò »ý¤Ã¤¿¥³¥ó¥Ô¥å¡¼¥¿¤Ç¤Ï¡¢¥Ö¥í¡¼¥É¥¥ã¥¹¥È¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤¬

¥·¥¹¥Æ¥àµ¯Æ°¸å¤ËÄÉ²Ã¤µ¤ì¤ë¤³¤È¤¬¤¢¤êÆÀ¤Þ¤¹¡£ **-w** ¥Õ¥é¥°¤òÍÑ¤¤¤ë¤È¡¢¤½¤Î¤è¤¦¤Ê¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤¬ 1 ¤Ä¤â ¸«¤Ä¤«¤é¤Ê¤¤¤È¤-¤Ë¤â¥¯¥é¥¤¥¢¥ó¥È¤¬½ªÎ»¤·¤Ê¤¤¤è¤¦¤Ë¤Ç¤¤Þ¤¹¡£ ¸å¤Ç **omshell (8)** ¥×¥í¥°¥é¥à¤»ÈÍÑ¤·¤Æ¡¢¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤¬ÄÉ²Ã¤µ¤ì¤¿¤ê ºï½ü¤µ¤ì¤¿¤ê¤·¤¿¤³¤È¤ò¥¯¥é¥¤¥¢¥ó¥È¤ËÄÌÃÎ¤¹¤ë¤³¤È¤¬¤Ç¤¡¢ ¤³¤ì¤Ë¤è¤Ã¤Æ¥¯¥é¥¤¥¢¥ó¥È¤¬¤³¤Î¥¤¥ó¥¿¥Õ¥§¡¼¥¹¾å¤Î IP ¥¢¥É¥ì¥¹¤òÀßÄê¤¹¤ë¤è¤¦»î¤ß¤ë¤³¤È¤¬¤Ç¤¤Þ¤¹¡£

-n ¥Õ¥é¥°¤òÍÑ¤¤¤ë¤³¤È¤Ç¡¢¤É¤Î¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤âÀßÄê¤·¤è¤¦¤È ¤·¤Ê¤¤¤è¤¦¤Ë DHCP ¥¯¥é¥¤¥¢¥ó¥È¤ò»Ø¼¨¤¹¤ë¤³¤È¤¬¤Ç¤¤Þ¤¹¡£ ¤³¤Î¥Õ¥é¥°¤Ï¡¢¤¤Ã¤È **-w** ¥Õ¥é¥°¤È¶¦¤Ë»ÈÍÑ¤¹¤ë¤ÈÍÍÑ¤Ç¤·¤ç¤¦¡£

IP ¥¢¥É¥ì¥¹¤ò³ÍÆÀ¤¹¤ë¤Þ¤ÇÂÔ¤Ä¤Î¤Ç¤Ï¤Ê¤¯¡¢Â¨ºÂ¤Ë¥Ç¡¼¥â¥ó¤È ¤Ê¤ë¤è¤¦¤Ë¥¯¥é¥¤¥¢¥ó¥È¤ò»Ø¼¨¤¹¤ë¤³¤È¤â¤Ç¤¤Þ¤¹¡£**-nw** ¥Õ¥é¥°¤òÍ¿¤¨¤ë¤È²Äǽ¤Ç¤¹¡£

ÀßÄê

dhclient.conf(5) ¥Õ¥¡¥¤¥ë¤Î½ñ¼°¤ÏÊÌ¤Ë²òÀâ¤µ¤ì¤Æ¤¤¤Þ¤¹¡£

OMAPI

¤³¤Î DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢Æ°ºîÃæ¤Ë¤½¤ÎÆ°ºî¤òÄä»ß¤µ¤»¤ë ¤³¤È¤Ê¤¯¼«Ê¬¼«¿È¤òÀ©¸æ¤Ç¤-¤ë¤è¤¦¤Ë¤¹¤ë¤¿¤á¤Îµ¡Ç½¤òÄó¶¡¤·¤Æ¤¤¤Þ¤¹¡£ ¤³¤Îµ¡Ç½¤Ï¡¢¥ê¥â¡¼¥È¥ª¥Ö¥¸¥§¥¯¥ÈÁàºî API ¤Ç¤¢¤ë OMAPI ¤ò ÍÑ¤¤¤ÆÄó¶¡¤µ¤ì¤Æ¤¤¤Þ¤¹¡£OMAPI ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢TCP/IP ¤ò »ÈÍÑ¤·¤Æ¤³¤Î DHCP ¥¯¥é¥¤¥¢¥ó¥È¤ËÀÜÂ³¤·¤Þ¤¹¡£¤½¤·¤Æ¡¢ DHCP ¥¯¥é¥¤¥¢¥ó¥È¤Î¸½ºß¤Î¾õÂÖ¤ò¸¡º¤Ç¤-¡¢¤½¤Î¾õÂÖ¤òÊÑ¹¹¤¹¤ë¤³¤È¤¬ ¤Ç¤¤Þ¤¹¡£

¥æ¡¼¥¶¥×¥í¥°¥é¥à¤Ç¤Ï¡¢´ðÁÃ¤Ë¤¢¤ë OMAPI ¥×¥í¥È¥³¥ë¤òÄ¾ÀÜ¼ÂÁõ¤¹¤ë ¤Î¤Ç¤Ï¤Ê¤¯¡¢dhcpctl API ¤â¤·¤¯¤Ï OMAPI ¤½¤Î¤â¤Î¤ò»ÈÍÑ¤¹¤Ù¤¤Ç¤¹¡£ dhcpctl ¤Ï¡¢OMAPI ¤¬¼«Æ°Ç¹Ô¤Ã¤Æ¤Ï¤¯¤ì¤Ê¤¤»¨»ö¤Î¤¤¤¯¤Ä¤«¤ò°·¤¦ ¥é¥Ã¥Ñ¤Ç¤¹¡£dhcpctl ¤ª¤è¤Ó OMAPI ¤Ë¤Ä¤¤¤Æ¤Ï **dhcpctl(3)** ¤ª¤è¤Ó **omapi(3)** ¤Ëµ½Ò¤µ¤ì¤Æ¤¤¤Þ¤¹¡£ ¥¯¥é¥¤¥¢¥ó¥È¤òÍÑ¤¤¤Æ¤ä¤ê¤¿¤¤¤³¤È¤Î¤Û¤È¤ó¤É¤Ï¡¢ÆÃÊÌ¤Ê¥×¥í¥°¥é¥à¤ò ½ñ¤«¤Ê¤¯¤È¤â **omshell(1)** ¥³¥Þ¥ó¥É¤ò»ÈÍÑ¤·¤ÆÄ¾ÀÜ¼Â¸½¤Ç¤¤ë ¤â¤Î¤Ç¤¹¡£

À©¸æ¥ª¥Ö¥¸¥§¥¯¥È

À©¸æ¥ª¥Ö¥¸¥§¥¯¥È¤ò»ÈÍÑ¤¹¤ë¤È¡¢DHCP ¥¯¥é¥¤¥¢¥ó¥È¤ò½ªÎ»¤µ¤»¡¢ ÊÝ»ý¤·¤Æ¤¤¤ë¥ê¡¼¥¹¤ò¤¹¤Ù¤Æ³«Êü¤·¡¢¥¯¥é¥¤¥¢¥ó¥È¤¬ÄÉ²Ã¤·¤¿ DNS ¥ì¥³¡¼¥É¤ò¤¹¤Ù¤Æ¾Ãµî¤¹¤ë¤³¤È¤¬¤Ç¤¤ë¤è¤¦¤Ë¤Ê¤ê¤Þ¤¹¡£ ¤Þ¤¿¡¢¥¯¥é¥¤¥¢¥ó¥È¤ò°ì»þÄä»ß¤µ¤»¡¢¥¯¥é¥¤¥¢¥ó¥È¤¬»ÈÍÑ¤·¤Æ¤¤¤ë ¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤ÎÀßÄê¤ò½ü¤¯¤³¤È¤¬¤Ç¤¤ë¤è¤¦¤Ë¤â¤Ê¤ê¤Þ¤¹¡£ ¤½¤Î¸å¤Ç¡¢DHCP ¥¯¥é¥¤¥¢¥ó¥È¤òºÆµ¯Æ°¤µ¤»¤ë¤³¤È¤¬¤Ç¤¡¢ ¥¤¥ó¥¿¥Õ¥§¡¼¥¹¤òºÆÀßÄê¤¹¤ë¤³¤È¤¬¤Ç¤-¤Þ¤¹¡£ÄÌ¾ï¡¢¥Ï¥¤¥Ð¥Í¡¼¥·¥ç¥ó¤Ë Æþ¤ëÁ°¤ä¥é¥Ã¥×¥È¥Ã¥×¥³¥ó¥Ô¥å¡¼¥¿¤Ç¥Ï¥ê¥¹¥ê¡¼¥×¤¹¤ëÁ°¤Ë DHCP ¥¯¥é¥¤¥¢¥ó¥È¤ò°ì»þÄä»ß¤µ¤»¤ë¤Ç¤·¤ç¤¦¡£ ¤½¤·¤Æ¡¢ÅÅ¸»¤¬Ìá¤Ã¤Æ¤¤¿¸å¤Ç DHCP ¥¯¥é¥¤¥¢¥ó¥È¤ò²óÉü¤µ¤»¤ë ¤Ç¤·¤ç¤¦¡£¤³¤¦¤¹¤ë¤³¤È¤Ç¡¢¥³¥ó¥Ô¥å¡¼¥¿¤¬¥Ï¥¤¥Ð¥Í¡¼¥·¥ç¥ó¤ä ¥¹¥ê¡¼¥×Ãæ¤Ë¤Ï PC ¥«¡¼¥É¤òÄä»ß¤µ¤»¤Æ¤ª¤¡¢¥³¥ó¥Ô¥å¡¼¥¿¤¬ ¥Ï¥¤¥Ð¥Í¡¼¥·¥ç¥ó¤ä¥¹¥ê¡¼¥×¤«¤éÉüµ¢¤·¤¿¤é¤é°ÊÁ°¤Î¾õÂÖ¤Ë ºÆÅÙ½é´ü²½¤¹¤ë¤³¤È¤¬¤Ç¤-¤ë¤è¤¦¤Ë¤Ê¤ë¤Î¤Ç¤¹¡£

À©¸æ¥ª¥Ö¥¸¥§¥¯¥È¤Ë¤Ï°À¤¬ 1 ¤Ä¤¢¤ê¤Þ¤¹¡£¤½¤ì¤Ï¾õÂÖ°À¤Ç¤¹¡£ ¥¯¥é¥¤¥¢¥ó¥È¤ò½ªÎ»¤µ¤»¤ë¤Ë¤Ï¡¢¥¯¥é¥¤¥¢¥ó¥È¤Î¾õÂÖ°À¤ò 2 ¤Ë ÀßÄê¤·¤Þ¤¹¡£¥¯¥é¥¤¥¢¥ó¥È¤Ï¼«Æ°Åª¤Ë DHCPRELEASE ¤ò¹Ô¤¦¤Ç¤·¤ç¤¦¡£ ¥¯¥é¥¤¥¢¥ó¥È¤ò°ì»þÄä»ß¤µ¤»¤ë¤Ë¤Ï¡¢¥¯¥é¥¤¥¢¥ó¥È¤Î¾õÂÖ°À¤ò 3 ¤ËÀßÄê¤·¤Þ¤¹¡£¥¯¥é¥¤¥¢¥ó¥È¤òÉüµ¢¤µ¤»¤ë¤Ë¤Ï¡¢¥¯¥é¥¤¥¢¥ó¥È¤Î ¾õÂÖ°À¤ò 4 ¤ËÀßÄê¤·¤Þ¤¹¡£

´ØÏ¢¥Õ¥¡¥¤¥ë

CLIENTBINDIR/dhclient-script, ETCDIR/dhclient.conf, DBDIR/dhclient.leases, RUNDIR/dhclient.pid, DBDIR/dhclient.leases˜

´ØÏ¢¹àÌÜ

dhclient.conf(5), dhclient.leases(5), dhclient-script(8)

º¹¼Ô

dhclient(8) ¤Ï Ted Lemon ¤¬ Vixie Enterprises ¤È¶¨ÎÏ¤·¤Æ Internet Systems Consortium ¤Î¤¿¤á¤Ë ½ñ¤¤Þ¤·¤¿¡£ Internet Systems Consortium ¤Ë¤Ä¤¤¤Æ¤è¤ê¾Ü¤·¤¯¤Ï¡¢**http://www.isc.org** ¤ò¤´Í÷¤¯¤À¤µ¤¤¡£ Vixie Enterprises ¤Ë¤Ä¤¤¤Æ¤è¤ê¾Ü¤·¤¯¤Ï¡¢**http://www.vix.com** ¤ò¤´Í÷¤¯¤À¤µ¤¤¡£

ËÜ¥¯¥é¥¤¥¢¥ó¥È¤Ï¡¢Elliot Poger ¤¬ Stanford Âç³Ø¤Î MosquitoNet ¥×¥í¥¸¥§¥¯¥È¤Ë»²²Ã¤·¤Æ¤¤¤ë´Ö¤Ë¡¢ Linux ¤Ç¤ÎÍøÍÑ¤ËºÝ¤·ÂçÉý¤Ë½¤Àµ¡¢²þÎÉ¤ò¹Ô¤¤¤Þ¤·¤¿¡£

¸½ºß¤Î¥Ð¡¼¥¸¥ç¥ó¤Ï¡¢Elliot ¤Ë¤è¤ë Linux ¤Ç¤Î²þÎÉ¤ËÉé¤¦¤È¤³¤í¤¬Âç¤¤¤¤Ç¤¹¤¬¡¢ Internet Systems Consortium ¤Î DHCP ¥µ¡¼¥Ð¤¬»È¤¦¤â¤Î¤ÈÆ±¤¸ ¥Í¥Ã¥È¥ï¡¼¥¯¥¤¥ó¥¿¥Õ¥§¡¼¥¹µ¡¹½¤ò»ÈÍÑ¤¹¤ë¤è¤¦¤Ë¡¢Ted Lemon ¤¬ÂçÉý¤ËºÆÊÔÀ®¤äÉôÊ¬Åª¤Ê½ñ¤´¹¤¨¤ò¹Ô¤¤¤Þ¤·¤¿¡£ ¥·¥¹¥Æ¥àÆÃÍ¤ÎÀßÄê¥³¡¼¥É¤ÎÂçÉôÊ¬¤Ï¥·¥§¥ë¥¹¥¯¥ê¥×¥È¤Ë°Ü¤µ¤ì¤¿¤Î¤Ç¡¢ ¤è¤êÂ¿¤¯¤Î¥ª¥Ú¥ì¡¼¥Æ¥£¥ó¥°¥·¥¹¥Æ¥à¤Î¥µ¥Ý¡¼¥È¤¬²Ã¤¨¤é¤ì¤ë¤Ë¤Ä¤ì¤¢ ¥·¥¹¥Æ¥àÆÃÍ¤ÎÀßÄê¥³¡¼¥É¤ò¤½¤Î¥ª¥Ú¥ì¡¼¥Æ¥£¥ó¥°¥·¥¹¥Æ¥à¤Ë °Ü¿¢¤·¤¿¤ê´ÉÍý¤·¤¿¤ê¤¹¤ëÉ¬Í×¤Ï¤Ê¤¯¤Ê¤ë¤Ç¤·¤ç¤¦¡£ Âå¤ï¤ê¤Ë¡¢¥·¥§¥ë¥¹¥¯¥ê¥×¥È¤¬´Ä¶¤Ë¹ç¤Ã¤¿¥Ä¡¼¥ë¤ò¸Æ¤Ó½Ð¤·¤Æ ¤½¤ÎÌÜÅª¤ò²Ì¤¿¤·¤Æ¤¯¤ì¤Þ¤¹¡£

NAME

dhcpd - Dynamic Host Configuration Protocol Server

SYNOPSIS

dhcpd [**-p** *port*] [**-f**] [**-d**] [**-q**] [**-t** | **-T**] [**-cf** *config-file*] [**-lf** *lease-file*] [**-tf** *trace-output-file*] [**-play** *trace-playback-file*] [*if0* [*...ifN*]]

DESCRIPTION

The Internet Systems Consortium DHCP Server, dhcpd, implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP). DHCP allows hosts on a TCP/IP network to request and be assigned IP addresses, and also to discover information about the network to which they are attached. BOOTP provides similar functionality, with certain restrictions.

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.

OPERATION

The DHCP protocol allows a host which is unknown to the network administrator to be automatically assigned a new IP address out of a pool of IP addresses for its network. In order for this to work, the network administrator allocates address pools in each subnet and enters them into the dhcpd.conf(5) file.

On startup, dhcpd reads the *dhcpd.conf* file and stores a list of available addresses on each subnet in memory. When a client requests an address using the DHCP protocol, dhcpd allocates an address for it. Each client is assigned a lease, which expires after an amount of time chosen by the administrator (by default, one day). Before leases expire, the clients to which leases are assigned are expected to renew them in order to continue to use the addresses. Once a lease has expired, the client to which that lease was assigned is no longer permitted to use the leased IP address.

In order to keep track of leases across system reboots and server restarts, dhcpd keeps a list of leases it has assigned in the dhcpd.leases(5) file. Before dhcpd grants a lease to a host, it records the lease in this file and makes sure that the contents of the file are flushed to disk. This ensures that even in the event of a system crash, dhcpd will not forget about a lease that it has assigned. On startup, after reading the dhcpd.conf file, dhcpd reads the dhcpd.leases file to refresh its memory about what leases have been assigned.

New leases are appended to the end of the dhcpd.leases file. In order to prevent the file from becoming arbitrarily large, from time to time dhcpd creates a new dhcpd.leases file from its in-core lease database. Once this file has been written to disk, the old file is renamed *dhcpd.leases˜*, and the new file is renamed dhcpd.leases. If the system crashes in the middle of this process, whichever dhcpd.leases file remains will contain all the lease information, so there is no need for a special crash recovery process.

BOOTP support is also provided by this server. Unlike DHCP, the BOOTP protocol does not provide a protocol for recovering dynamically-assigned addresses once they are no longer needed. It is still possible to dynamically assign addresses to BOOTP clients, but some administrative process for reclaiming addresses is required. By default, leases are granted to BOOTP clients in perpetuity, although the network administrator may set an earlier cutoff date or a shorter lease length for BOOTP leases if that makes sense.

BOOTP clients may also be served in the old standard way, which is to simply provide a declaration in the dhcpd.conf file for each BOOTP client, permanently assigning an address to each client.

Whenever changes are made to the dhcpd.conf file, dhcpd must be restarted. To restart dhcpd, send a SIGTERM (signal 15) to the process ID contained in */var/run/dhcpd.pid*, and then re-invoke dhcpd. Because the DHCP server database is not as lightweight as a BOOTP database, dhcpd does not automatically restart itself when it sees a change to the dhcpd.conf file.

Note: We get a lot of complaints about this. We realize that it would be nice if one could send a SIGHUP to the server and have it reload the database. This is not technically impossible, but it would require a great deal of work, our resources are extremely limited, and they can be better spent elsewhere. So please don't complain about this on the mailing list unless you're prepared to fund a project to implement this feature, or prepared to do it yourself.

COMMAND LINE

The names of the network interfaces on which dhcpd should listen for broadcasts may be specified on the command line. This should be done on systems where dhcpd is unable to identify non-broadcast interfaces, but should not be required on other systems. If no interface names are specified on the command line dhcpd will identify all network interfaces which are up, eliminating non-broadcast interfaces if possible, and listen for DHCP broadcasts on each interface.

If dhcpd should listen on a port other than the standard (port 67), the **-p** flag may used. It should be followed by the udp port number on which dhcpd should listen. This is mostly useful for debugging purposes.

To run dhcpd as a foreground process, rather than allowing it to run as a daemon in the background, the **-f** flag should be specified. This is useful when running dhcpd under a debugger, or when running it out of inittab on System V systems.

To have dhcpd log to the standard error descriptor, specify the **-d** flag. This can be useful for debugging, and also at sites where a complete log of all dhcp activity must be kept but syslogd is not reliable or otherwise cannot be used. Normally, dhcpd will log all output using the syslog(3) function with the log facility set to LOG_DAEMON.

Dhcpd can be made to use an alternate configuration file with the **-cf** flag, or an alternate lease file with the **-lf** flag. Because of the importance of using the same lease database at all times when running dhcpd in production, these options should be used **only** for testing lease files or database files in a non-production environment.

When starting dhcpd up from a system startup script (e.g., /etc/rc), it may not be desirable to print out the entire copyright message on startup. To avoid printing this message, the **-q** flag may be specified.

The DHCP server reads two files on startup: a configuration file, and a lease database. If the **-t** flag is specified, the server will simply test the configuration file for correct syntax, but will not attempt to perform any network operations. This can be used to test the a new configuration file automatically before installing it.

The **-T** flag can be used to test the lease database file in a similar way.

The **-tf** and **-play** options allow you to specify a file into which the entire startup state of the server and all the transactions it processes are either logged or played back from. This can be useful in submitting bug reports - if you are getting a core dump every so often, you can start the server with the **-tf** option and then, when the server dumps core, the trace file will contain all the transactions that led up to it dumping core, so that the problem can be easily debugged with **-play**.

The **-play** option must be specified with an alternate lease file, using the **-lf** switch, so that the DHCP server doesn't wipe out your existing lease file with its test data. The DHCP server will refuse to operate in playback mode unless you specify an alternate lease file.

CONFIGURATION

The syntax of the dhcpd.conf(5) file is discussed separately. This section should be used as an overview of the configuration process, and the dhcpd.conf(5) documentation should be consulted for detailed reference information.

Subnets

dhcpd needs to know the subnet numbers and netmasks of all subnets for which it will be providing service. In addition, in order to dynamically allocate addresses, it must be assigned one or more ranges of addresses on each subnet which it can in turn assign to client hosts as they boot. Thus, a very simple configuration providing DHCP support might look like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.250;
}
```

Multiple address ranges may be specified like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.107;
  range 239.252.197.113 239.252.197.250;
}
```

If a subnet will only be provided with BOOTP service and no dynamic address assignment, the range clause can be left out entirely, but the subnet statement must appear.

Lease Lengths

DHCP leases can be assigned almost any length from zero seconds to infinity. What lease length makes sense for any given subnet, or for any given installation, will vary depending on the kinds of hosts being served.

For example, in an office environment where systems are added from time to time and removed from time to time, but move relatively infrequently, it might make sense to allow lease times of a month of more. In a final test environment on a manufacturing floor, it may make more sense to assign a maximum lease length of 30 minutes - enough time to go through a simple test procedure on a network appliance before packaging it up for delivery.

It is possible to specify two lease lengths: the default length that will be assigned if a client doesn't ask for any particular lease length, and a maximum lease length. These are specified as clauses to the subnet command:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.107;
  default-lease-time 600;
  max-lease-time 7200;
}
```

This particular subnet declaration specifies a default lease time of 600 seconds (ten minutes), and a maximum lease time of 7200 seconds (two hours). Other common values would be 86400 (one day), 604800 (one week) and 2592000 (30 days).

Each subnet need not have the same lease—in the case of an office environment and a manufacturing environment served by the same DHCP server, it might make sense to have widely disparate values for default and maximum lease times on each subnet.

BOOTP Support

Each BOOTP client must be explicitly declared in the dhcpd.conf file. A very basic client declaration will specify the client network interface's hardware address and the IP address to assign to that client. If the client needs to be able to load a boot file from the server, that file's name must be specified. A simple bootp client declaration might look like this:

```
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 239.252.197.9;
  filename "/tftpboot/haagen.boot";
}
```

Options

DHCP (and also BOOTP with Vendor Extensions) provide a mechanism whereby the server can provide the client with information about how to configure its network interface (e.g., subnet mask), and also how the client can access various network services (e.g., DNS, IP routers, and so on).

These options can be specified on a per-subnet basis, and, for BOOTP clients, also on a per-client basis. In the event that a BOOTP client declaration specifies options that are also specified in its subnet declaration, the options specified in the client declaration take precedence. A reasonably complete DHCP configuration might look something like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.250;
  default-lease-time 600 max-lease-time 7200;
  option subnet-mask 255.255.255.0;
  option broadcast-address 239.252.197.255;
  option routers 239.252.197.1;
  option domain-name-servers 239.252.197.2, 239.252.197.3;
  option domain-name "isc.org";
}
```

A bootp host on that subnet that needs to be in a different domain and use a different name server might be declared as follows:

```
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 239.252.197.9;
  filename "/tftpboot/haagen.boot";
  option domain-name-servers 192.5.5.1;
  option domain-name "vix.com";
}
```

A more complete description of the dhcpd.conf file syntax is provided in dhcpd.conf(5).

OMAPI

The DHCP server provides the capability to modify some of its configuration while it is running, without stopping it, modifying its database files, and restarting it. This capability is currently provided using OMAPI - an API for manipulating remote objects. OMAPI clients connect to the server using TCP/IP, authenticate, and can then examine the server's current status and make changes to it.

Rather than implementing the underlying OMAPI protocol directly, user programs should use the dhcpctl API or OMAPI itself. Dhcpctl is a wrapper that handles some of the housekeeping chores that OMAPI does not do automatically. Dhcpctl and OMAPI are documented in **dhcpctl(3)** and **omapi(3)**.

OMAPI exports objects, which can then be examined and modified. The DHCP server exports the following objects: lease, host, failover-state and group. Each object has a number of methods that are provided: lookup, create, and destroy. In addition, it is possible to look at attributes that are stored on objects, and in some cases to modify those attributes.

THE LEASE OBJECT

Leases can't currently be created or destroyed, but they can be looked up to examine and modify their state.

Leases have the following attributes:

state *integer* lookup, examine

1 = free
2 = active
3 = expired
4 = released
5 = abandoned
6 = reset
7 = backup
8 = reserved
9 = bootp

ip-address *data* lookup, examine

The IP address of the lease.

dhcp-client-identifier *data* lookup, examine, update

The client identifier that the client used when it acquired the lease. Not all clients send client identifiers, so this may be empty.

client-hostname *data* examine, update
The value the client sent in the host-name option.

host *handle* examine
the host declaration associated with this lease, if any.

subnet *handle* examine
the subnet object associated with this lease (the subnet object is not currently supported).

pool *handle* examine
the pool object associated with this lease (the pool object is not currently supported).

billing-class *handle* examine
the handle to the class to which this lease is currently billed, if any (the class object is not currently supported).

hardware-address *data* examine, update
the hardware address (chaddr) field sent by the client when it acquired its lease.

hardware-type *integer* examine, update
the type of the network interface that the client reported when it acquired its lease.

ends *time* examine
the time when the lease's current state ends, as understood by the client.

tstp *time* examine
the time when the lease's current state ends, as understood by the server.

tsfp *time* examine
the time when the lease's current state ends, as understood by the failover peer (if there is no failover peer, this value is undefined).

cltt *time* examine
The time of the last transaction with the client on this lease.

THE HOST OBJECT

Hosts can be created, destroyed, looked up, examined and modified. If a host declaration is created or deleted using OMAPI, that information will be recorded in the dhcpd.leases file. It is permissible to delete host declarations that are declared in the dhcpd.conf file.

Hosts have the following attributes:

name *data* lookup, examine, modify
the name of the host declaration. This name must be unique among all host declarations.

group *handle* examine, modify
the named group associated with the host declaration, if there is one.

hardware-address *data* lookup, examine, modify
the link-layer address that will be used to match the client, if any. Only valid if hardware-type is also present.

hardware-type *integer* lookup, examine, modify
the type of the network interface that will be used to match the client, if any. Only valid if hardware-address is also present.

dhcp-client-identifier *data* lookup, examine, modify
the dhcp-client-identifier option that will be used to match the client, if any.

ip-address *data* examine, modify
a fixed IP address which is reserved for a DHCP client that matches this host declaration. The IP address will only be assigned to the client if it is valid for the network segment to which the client is connected.

statements *data* modify
a list of statements in the format of the dhcpd.conf file that will be executed whenever a message

from the client is being processed.

known *integer* examine, modify
: if nonzero, indicates that a client matching this host declaration will be treated as *known* in pool permit lists. If zero, the client will not be treated as known.

THE GROUP OBJECT

Named groups can be created, destroyed, looked up, examined and modified. If a group declaration is created or deleted using OMAPI, that information will be recorded in the dhcpd.leases file. It is permissible to delete group declarations that are declared in the dhcpd.conf file.

Named groups currently can only be associated with hosts - this allows one set of statements to be efficiently attached to more than one host declaration.

Groups have the following attributes:

name *data*
: the name of the group. All groups that are created using OMAPI must have names, and the names must be unique among all groups.

statements *data*
: a list of statements in the format of the dhcpd.conf file that will be executed whenever a message from a client whose host declaration references this group is processed.

THE CONTROL OBJECT

The control object allows you to shut the server down. If the server is doing failover with another peer, it will make a clean transition into the shutdown state and notify its peer, so that the peer can go into partner down, and then record the "recover" state in the lease file so that when the server is restarted, it will automatically resynchronize with its peer.

On shutdown the server will also attempt to cleanly shut down all OMAPI connections. If these connections do not go down cleanly after five seconds, they are shut down pre-emptively. It can take as much as 25 seconds from the beginning of the shutdown process to the time that the server actually exits.

To shut the server down, open its control object and set the state attribute to 2.

THE FAILOVER-STATE OBJECT

The failover-state object is the object that tracks the state of the failover protocol as it is being managed for a given failover peer. The failover object has the following attributes (please see **dhcpd.conf (5)** for explanations about what these attributes mean):

name *data* examine
: Indicates the name of the failover peer relationship, as described in the server's **dhcpd.conf** file.

partner-address *data* examine
: Indicates the failover partner's IP address.

local-address *data* examine
: Indicates the IP address that is being used by the DHCP server for this failover pair.

partner-port *data* examine
: Indicates the TCP port on which the failover partner is listening for failover protocol connections.

local-port *data* examine
: Indicates the TCP port on which the DHCP server is listening for failover protocol connections for this failover pair.

max-outstanding-updates *integer* examine
: Indicates the number of updates that can be outstanding and unacknowledged at any given time, in this failover relationship.

mclt *integer* examine
: Indicates the maximum client lead time in this failover relationship.

load-balance-max-secs *integer* examine

Indicates the maximum value for the secs field in a client request before load balancing is bypassed.

load-balance-hba *data* examine
Indicates the load balancing hash bucket array for this failover relationship.

local-state *integer* examine, modify
Indicates the present state of the DHCP server in this failover relationship. Possible values for state are:

1 - partner down
2 - normal
3 - communications interrupted
4 - resolution interrupted
5 - potential conflict
6 - recover
7 - recover done
8 - shutdown
9 - paused
10 - startup
11 - recover wait

In general it is not a good idea to make changes to this state. However, in the case that the failover partner is known to be down, it can be useful to set the DHCP server's failover state to partner down. At this point the DHCP server will take over service of the failover partner's leases as soon as possible, and will give out normal leases, not leases that are restricted by MCLT. If you do put the DHCP server into the partner-down when the other DHCP server is not in the partner-down state, but is not reachable, IP address assignment conflicts are possible, even likely. Once a server has been put into partner-down mode, its failover partner must not be brought back online until communication is possible between the two servers.

partner-state *integer* examine
Indicates the present state of the failover partner.

local-stos *integer* examine
Indicates the time at which the DHCP server entered its present state in this failover relationship.

partner-stos *integer* examine
Indicates the time at which the failover partner entered its present state.

hierarchy *integer* examine
Indicates whether the DHCP server is primary (0) or secondary (1) in this failover relationship.

last-packet-sent *integer* examine
Indicates the time at which the most recent failover packet was sent by this DHCP server to its failover partner.

last-timestamp-received *integer* examine
Indicates the timestamp that was on the failover message most recently received from the failover partner.

skew *integer* examine
Indicates the skew between the failover partner's clock and this DHCP server's clock

max-response-delay *integer* examine
Indicates the time in seconds after which, if no message is received from the failover partner, the partner is assumed to be out of communication.

cur-unacked-updates *integer* examine
Indicates the number of update messages that have been received from the failover partner but not yet processed.

FILES

/etc/dhcpd.conf, /var/db/dhcpd.leases, /var/run/dhcpd.pid, /var/db/dhcpd.leases~.

SEE ALSO

dhclient(8), dhcrelay(8), dhcpd.conf(5), dhcpd.leases(5)

AUTHOR

dhcpd(8) was originally written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by Internet Systems Consortium. Version 3 of the DHCP server was funded by Nominum, Inc. Information about Internet Systems Consortium is available at **http://www.isc.org/**. Information about Nominum can be found at **http://www.nominum.com/**.

NAME

dhcrelay - Dynamic Host Configuration Protocol Relay Agent

SYNOPSIS

dhcrelay [**-p** *port*] [**-d**] [**-q**] [**-i** *if0* [**...** **-i** *ifN*]] [**-a**] [**-c** *count*] [**-A** *length*] [**-D**] [**-m** *append* | *replace* | *forward* | *discard*] *server0* [*...serverN*]

DESCRIPTION

The Internet Systems Consortium DHCP Relay Agent, dhcrelay, provides a means for relaying DHCP and BOOTP requests from a subnet to which no DHCP server is directly connected to one or more DHCP servers on other subnets.

SYSTEM REQUIREMENTS

You must have the Berkeley Packet Filter (bpf) configured in your NetBSD kernel.

OPERATION

The DHCP Relay Agent listens for DHCP and BOOTP queries and responses. When a query is received from a client, dhcrelay forwards it to the list of DHCP servers specified on the command line. When a reply is received from a server, it is broadcast or unicast (according to the relay agent's ability or the client's request) on the network from which the original request came.

COMMAND LINE

The names of the network interfaces that dhcrelay should attempt to configure may be specified on the command line using the **-i** option. If no interface names are specified on the command line dhcrelay will identify all network interfaces, elimininating non-broadcast interfaces if possible, and attempt to configure each interface.

The **-i** flag can be used to specify the network interfaces on which the relay agent should listen. In general, it must listen not only on those network interfaces to which clients are attached, but also on those network interfaces to which the server (or the router that reaches the server) is attached. However, in some cases it may be necessary to exclude some networks; in this case, you must list all those network interfaces that should *not* be excluded using the **-i** flag.

In some cases it *is* helpful for the relay agent to forward requests from networks on which a DHCP server is running to other DHCP servers. This would be the case if two DHCP servers on different networks were being used to provide backup service for each other's networks.

If dhcrelay should listen and transmit on a port other than the standard (port 67), the **-p** flag may used. It should be followed by the udp port number that dhcrelay should use. This is mostly useful for debugging purposes.

Dhcrelay will normally run in the foreground until it has configured an interface, and then will revert to running in the background. To force dhcrelay to always run as a foreground process, the **-d** flag should be specified. This is useful when running dhcrelay under a debugger, or when running it out of inittab on System V systems.

Dhcrelay will normally print its network configuration on startup. This can be unhelpful in a system startup script - to disable this behaviour, specify the **-q** flag.

RELAY AGENT INFORMATION OPTIONS

If the **-a** flag is set the relay agent will append an agent option field to each request before forwarding it to the server. Agent option fields in responses sent from servers to clients will be stripped before forwarding such responses back to the client.

The agent option field will contain two agent options: the Circuit ID suboption and the Remote ID suboption. Currently, the Circuit ID will be the printable name of the interface on which the client request was received. The client supports inclusion of a Remote ID suboption as well, but this is not used by default.

When forwarding packets, dhcrelay discards packets which have reached a hop count of 10. If a lower or higher threshold (up to 255) is desired, depending on your environment, you can specify the max hop count threshold as a number following the **-c** option.

Relay Agent options are added to a DHCP packet without the knowledge of the DHCP client. The client may have filled the DHCP packet option buffer completely, in which case there theoretically isn't any space to add Agent options. However, the DHCP server may be able to handle a much larger packet than most DHCP clients would send. The current Agent Options draft requires that the relay agent use a maximum packet size of 576 bytes.

It is recommended that with the Internet Systems Consortium DHCP server, the maximum packet size be set to about 1400, allowing plenty of extra space in which the relay agent can put the agent option field, while still fitting into the Ethernet MTU size. This can be done by specifying the **-A** flag, followed by the desired maximum packet size (e.g., 1400).

Note that this is reasonably safe to do even if the MTU between the server and the client is less than 1500, as long as the hosts on which the server and client are running support IP fragmentation (and they should). With some knowledge as to how large the agent options might get in a particular configuration, this parameter can be tuned as finely as necessary.

It is possible for a relay agent to receive a packet which already contains an agent option field. If this packet does not have a giaddr set, the standard requires that the packet be discarded.

If giaddr is set, the server may handle the situation in one of four ways: it may *append* its own set of relay options to the packet, leaving the supplied option field intact. It may *replace* the existing agent option field. It may *forward* the packet unchanged. Or, it may *discard* it.

Which of these behaviours is followed by the Internet Systems Consortium DHCP Relay Agent may be configured with the **-m** flag, followed by one of the four keywords specified in *italics* above.

When the relay agent receives a reply from a server that it's supposed to forward to a client, and Relay Agent Information option processing is enabled, the relay agent scans the packet for Relay Agent Information options and removes them. As it's scanning, if it finds a Relay Agent Information option field containing an Agent ID suboption that matches one of its IP addresses, that option is recognized as its own. If no such option is found, the relay agent can either drop the packet, or relay it anyway. If the **-D** option is specified, all packets that don't contain a match will be dropped.

SPECIFYING DHCP SERVERS

The name or IP address of at least one DHCP server to which DHCP and BOOTP requests should be relayed must be specified on the command line.

SEE ALSO

dhclient(8), dhcpd(8), RFC2132, RFC2131, draft-ietf-dhc-agent-options-03.txt.

BUGS

It should be possible for the user to define the Circuit ID and Remote ID values on a per-interface basis.

The relay agent should not relay packets received on a physical network to DHCP servers on the same physical network - if they do, the server will receive duplicate packets. In order to fix this, however, the relay agent needs to be able to learn about the network topology, which requires that it have a configuration file.

AUTHOR

dhcrelay(8) has been written for Internet Systems Consortium by Ted Lemon in cooperation with Vixie Enterprises. To learn more about Internet Systems Consortium, see **http://www.isc.org/isc.** To learn more about Vixie Enterprises, see **http://www.vix.com.**

NAME

discard − Postfix discard mail delivery agent

SYNOPSIS

discard [generic Postfix daemon options]

DESCRIPTION

The Postfix **discard**(8) delivery agent processes delivery requests from the queue manager. Each request specifies a queue file, a sender address, a domain or host name that is treated as the reason for discarding the mail, and recipient information. The reason may be prefixed with an RFC 3463-compatible detail code. This program expects to be run from the **master**(8) process manager.

The **discard**(8) delivery agent pretends to deliver all recipients in the delivery request, logs the "next-hop" domain or host information as the reason for discarding the mail, updates the queue file and marks recipients as finished or informs the queue manager that delivery should be tried again at a later time.

Delivery status reports are sent to the **trace**(8) daemon as appropriate.

SECURITY

The **discard**(8) mailer is not security-sensitive. It does not talk to the network, and can be run chrooted at fixed low privilege.

STANDARDS

None.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces and of other trouble.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **discard**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

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.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

double_bounce_sender (double-bounce)
The sender address of postmaster notifications that are generated by the mail system.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

qmgr(8), queue manager
bounce(8), delivery status reports
error(8), Postfix error delivery agent
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

LICENSE

HISTORY

This service was introduced with Postfix version 2.2.

AUTHOR(S)

Victor Duchovni
Morgan Stanley

Based on code by:
Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

disklabel — read and write disk pack label

SYNOPSIS

disklabel [**−ACDFrtv**] *disk*
disklabel −e [**−CDFIrv**] *disk*
disklabel −i [**−DFIrv**] *disk*
disklabel −R [**−DFrv**] *disk protofile*
disklabel −w [**−DFrv**] [**−f** *disktab*] *disk disktype* [*packid*]
disklabel [**−NW**] *disk*
disklabel −l

DESCRIPTION

disklabel can be used to install, examine, or modify the label on a disk drive or pack. When writing the label, it can be used to change the drive identification, the disk partitions on the drive, or to replace a damaged label.

The **−e**, **−i**, **−l**, **−R**, **−w**, **−N**, and **−W** options determine the basic operation. If none are specified the label is displayed.

−e Edit the existing label (using EDITOR) and write it back to the disk. If EDITOR is undefined, then vi(1) is used.

−i Interactively update the existing label and write it back to the disk.

−l Show all known file system types (those that can be specified along a partition within the label) and exit.

−R Write (restore) a label by reading it from *protofile*. The file should be in the same format as the default output.

−w Write a standard label for the specified *disktype*. See disktab(5).

−N Disallow writes to the disk sector that contains the label. This is the default state.

−W Allow writes to the disk sector that contains the label. This state may not persist if no programs have the disk open.

The majority of the rest of the options affect more than one form of the command:

−A Read all labels from the disk, including ones deleted with **disklabel −D**. Implies **−r**.

−C Output the partition offset and size values in ⟨cylinder/head/sector⟩ format. Note this format is always accepted on input with either the **−e** or **−R** flags.

−D Delete all existing labels (by 1's complementing the magic number) before writing any labels to their default location. Implies **−r**. If **−D** is specified without a request to write the label, then existing labels are just deleted.

−F Treat *disk* as a regular file. This suppresses all ioctl(2) calls, and is the default if *disk* is a regular file. *disk* is always opened using opendisk(3) even if **−F** is specified. Implies **−r**.

−I If a label cannot be read from *disk* request the default one from the kernel. Implies **−r**.

−f *disktab*
Specify the name of a file to use instead of /etc/disktab.

−r Read/write the disk directly rather than using ioctl(2) requests on the kernel. When writing a label, the kernel will be told about the label before the label is written and asked to write afterwards. This is the historic behaviour and can be supressed by specifying **−F**.

-t Format the output as a `disktab`(5) entry.

-v Be verbose about the operations being done, in particular the disk sectors being read and written. Specifying **-v** more than once will increase the verbosity.

On systems that expect to have disks with MBR partitions (see `fdisk`(8)) **disklabel** will find, and update if requested, labels in the first 8k of type 169 (NetBSD) MBR labels and within the first 8k of the physical disk. On other systems **disklabel** will only look at the start of the disk. The offset at which the labels are written is also system dependent.

disklabel will detect byteswapped labels, but currently cannot display them.

Previous versions of **disklabel** could update the bootstrap code on some architectures. This functionality has been subsumed by `installboot`(8).

EXIT STATUS

The exit status of **disklabel** is set to indicate any errors or warnings. The values used are:

0	The **disklabel** utility has completed successfully.
1	A fatal error has occurred, such as unknown options passed on the command line, or writing the disklabel failed.
4	An I/O error of some sort occurred.
101..n	One or more warnings occured while reading the disklabel. Subtract 100 to get the number of warnings detected.

FILES

`/etc/disktab`

EXAMPLES

```
disklabel sd0
```

Display the in-core label for sd0 as obtained via `/dev/rsd0c`.

```
disklabel -w -r /dev/rsd0c sd2212 foo
```

Create a label for sd0 based on information for “sd2212” found in `/etc/disktab`, using `foo` as the disk pack label. If you do not have an entry for your disk in `/etc/disktab`, you can use this style to put an initial label onto a new disk. Then dump the label to a file (using **disklabel sd0 > protofile**), editing the file, and replacing the label with **disklabel -R sd0 protofile**.

```
disklabel -e -r sd0
```

Read the on-disk label for sd0, edit it and reinstall in-core as well as on-disk.

```
disklabel -e -I sd0
```

As previous, but don't fail if there was no label on the disk yet; provide some default values instead.

```
disklabel -i -I sd0
```

As previous, only use the built-in interactive editor.

```
disklabel -R sd0 mylabel
```

Restore the on-disk and in-core label for sd0 from information in `mylabel`.

DIAGNOSTICS

The kernel device drivers will not allow the size of a disk partition to be decreased or the offset of a partition to be changed while it is open. Some device drivers create a label containing only a single large partition if a disk is unlabeled; thus, the label must be written to the “a” partition of the disk while it is open. This sometimes requires the desired label to be set in two steps, the first one creating at least one other partition, and the second setting the label on the new partition while shrinking the “a” partition.

SEE ALSO

`opendisk`(3), `disklabel`(5), `disktab`(5), `dkctl`(8), `fdisk`(8), `installboot`(8), `mbrlabel`(8), `mscdlabel`(8)

BUGS

If the disk partition is not specified in the disk name (i.e., *xy0* instead of */dev/rxy0c*), **disklabel** will construct the full pathname of the disk and use the “d” partition on i386, hpcmips, or arc, and the “c” partition on all others.

On the sparc, sparc64, sun2, and sun3 NetBSD systems, the size of each partition must be a multiple of the number of sectors per cylinder (i.e., each partition must be an integer number of cylinders), or the boot ROMs will declare the label invalid and fail to boot the system.

In addition, the **−r** option should never be used on a sparc, sparc64, sun2, or sun3 system boot disk - the NetBSD kernel translates the NetBSD disk label into a SunOS compatible format (which is required by the boot PROMs) when it writes the label. Using the **−r** flag causes **disklabel** to write directly to disk, and bypass the format translation. This will result in a disk label that the PROMs will not recognize, and that therefore cannot be booted from.

NAME

diskless — booting a system over the network

DESCRIPTION

The ability to boot a system over the network is useful for two kinds of systems:

diskless	a system with no attached mass storage media to boot or run from (e.g. a network computer).
dataless	a system with a hard drive that only contains system and application software, and user data is mounted over the network from a central server.

It can also be done as a temporary measure while repairing or re-installing file systems on a local disk. This capability is necessarily platform dependent because of its dependence on system firmware support; not all platforms supported by NetBSD are capable of being network booted.

The protocols used to obtain a network address (e.g. an IP host address), include, but are not limited to:

RARP	Reverse Address Resolution Protocol (ARP)
DHCP	Dynamic Host Configuration Protocol
BOOTP	Bootstrap Protocol

This information can also be derived from non-volatile RAM or by a transform of a network interface (e.g. Ethernet) MAC address.

The protocols used to load a NetBSD kernel over a network include, but are not limited to:

TFTP	Trivial File Transfer Protocol
NFS	Sun Network File System
RMP	HP Remote Maintenance Protocol
MOP	DEC Maintenance Operations Protocol

Derivation of the filename of the secondary bootstrap program can be done by a transform of a network interface MAC address (or other protocol address), or provided by a server as with BOOTP, and DHCP. How this is done is platform dependent; see boot(8).

The NetBSD kernel doesn't care how it gets loaded and started. The protocols used to boot NetBSD can be completely different than the ones that NetBSD uses operationally, i.e. you can netboot the system using HP RMP and the NetBSD kernel can use IP to communicate after bootstrap.

There is no standard way to pass all the required information from a boot loader to an operating system kernel, so the NetBSD kernel usually has to recapitulate the same (or similar) protocol exchanges over the network to obtain a network address, determine which servers to use, and so on. NetBSD supports obtaining this information from RARP, BOOTP, DHCP, and Sun RPC "bootparams". See options(4) for a list of methods that can be compiled into a NetBSD kernel.

NetBSD only supports the Sun Network File System (NFS) for mounting its root file system over a network. NetBSD can use any local mass storage device for which it has a driver, after bootstrap, even if that device is not supported by the system's firmware for booting.

N.B. DHCP is essentially a series of extensions to BOOTP; the NetBSD dhcpd(8) is capable of responding to both kinds of protocol requests.

In the majority of configurations, network boot servers and clients are attached to the same LAN so that broadcast queries from the clients can be heard by the servers. Unless specially configured, routers block broadcasts from propagating from LAN to LAN; some routers can be configured to "forward" broadcast BOOTP packets to another LAN attached to that router, which permits a server on that remote LAN to respond to the client's broadcast query.

OPERATION

When booting a system over the network, there are three phases of interaction between client and server:

1. The system firmware (or stage-1 bootstrap) loads a boot program.
2. The boot program loads a NetBSD kernel.
3. The NetBSD kernel performs an NFS mount of the root file system.

Each of these phases are described in further detail below.

1. loading a boot program

In phase 1, the system firmware loads a boot program. Firmware designs vary widely, so this phase is inherently machine-specific. Some examples:

DEC Alpha systems use BOOTP to determine the client's IP address and then use TFTP load a secondary bootstrap program from the server and filename specified in the BOOTP reply. DEC Alpha systems can also use MOP to load a program to run the system.

Sun systems use RARP to determine the client's IP address, transform that address to a hexadecimal string to form the filename of the secondary boot program, and then use TFTP to download the boot program from the server that sent the RARP reply.

HP 300-series systems use the HP RMP to download a boot program.

Typical personal computers may load a network boot program either from diskette or from a PROM on a Network Interface Card (NIC). Some BIOSes support booting from a network interface.

2. loading a kernel

In phase 2, the secondary boot program loads a kernel. Operation in this phase depends on the design of the boot program (the design described here is the one used by Sun and NetBSD/hp300). The boot program:

1. gets the client IP address using RARP.
2. gets the client name and server IP address by broadcasting an RPC / BOOTPARAMS / WHOAMI request with the client IP address.
3. gets the server path for this client's root using an RPC / BOOTPARAMS / GETFILE request with the client name.
4. gets the root file handle by calling `mountd`(8) with the server path for the client root file system.
5. gets the kernel file handle by calling NFS **`lookup`**() on the root file handle.
6. loads the kernel using NFS read calls on the kernel file handle.
7. transfers control to the kernel entry point.

A BOOTP and/or DHCP secondary bootstrap program will do the following:

1. query for the client's bootstrap parameters. The response must include the client's IP address, and a TFTP server to load the NetBSD kernel from.
2. loads the NetBSD kernel from the TFTP server.
3. transfers control to the kernel entry point.

3. NFS mounting the root file system

In phase 3, the kernel performs an NFS mount of the root file system. The kernel repeats much of the work done by the boot program because there is no standard way for the boot program to pass the information it gathered on to the kernel.

In general, the GENERIC kernel `config`(1) file for any particular architecture will specify compile-time options to use the same protocol used by the secondary boot program for that architecture. A NetBSD kernel can be compiled to use any of BOOTP, DHCP, or Sun RPC BOOTPARAMS; see `options`(4).

The procedure typically used by the kernel is as follows:

1. The kernel finds a boot server using the same procedures as described above to determine the client's IP address, an NFS server, etc.
2. The kernel gets the NFS file handle for root using the same procedure as described above.
3. The kernel calls the NFS **getattr**() function to get the last-modified time of the root directory, and uses it to check the system clock.

SERVER CONFIGURATION

Before a client can bootstrap over the network, its server must be configured. Each daemon that implements these protocols must be set up so that it can answer queries from the clients. Some of these daemons are invoked as packets come in, by `inetd`(8), and some must run independently, started from `/etc/rc`; see `rc.conf`(5).

Protocol	**Program**	**Startup**
RARP	rarpd	`rc.conf`(5)
DHCP	dhcpd	`rc.conf`(5)
BOOTP	bootpd	`inetd.conf`(5)
TFTP	tfptd	`inetd.conf`(5)
Sun RPC	rpcbind	`rc.conf`(5)
Sun RPC	rpc.bootparamd	`rc.conf`(5)
Sun NFS	mountd	`rc.conf`(5)
Sun NFS	nfsiod	`rc.conf`(5)
HP RMP	rbootd	`rc.conf`(5)

N.B. DHCP is essentially a series of extensions to BOOTP; the NetBSD `dhcpd`(8) is capable of responding to both kinds of protocol requests. Since they both bind to the same UDP port, only one may be run on a given server.

In the following examples, the client's hostname is **myclient**; the server is **myserver**, and the addresses are all fictional. In these examples the hostnames may be Fully Qualified Domain Names (FQDN, e.g. "myclient.mydomain.com") provided that they are used consistently.

RARP

For clients that use RARP to obtain their IP address, an entry must be added for each client to `/etc/ethers` with the client's Ethernet MAC address and Internet hostname:

```
8:0:20:7:c5:c7          myclient
```

This will be used by `rarpd`(8) to reply to queries from the clients. There must be one entry per client system.

A client system's Ethernet MAC address is often printed on the system case, or on a chip on its motherboard, or on the NIC. If not, "sniffing" the network with `tcpdump`(8) when the client is powered-on should reveal its Ethernet MAC address.

Each client system that uses RARP must have its own, unique IP address assigned to it. Assign an IP address for myclient in your `/etc/hosts` file, or in the master file for your DNS zone. For `/etc/hosts` the entry should look like:

```
192.197.96.12           myclient
```

DHCP/BOOTP

The NetBSD DHCP server `dhcpd`(8) was developed by the Internet Software Consortium (ISC); `http://www.isc.org/`

DHCP can provide a wide range of information to a requesting client; the key data for bootstrapping a diskless client are:

1. an IP address
2. a subnet mask
3. a TFTP server address for loading the secondary bootstrap and the NetBSD kernel
4. a filename of the secondary bootstrap
5. an NFS server address for the client's file system
6. the client's root file system path, to be NFS mounted.

An example for `/etc/dhcpd.conf`

```
host myclient {
        hardware ethernet 8:0:20:7:c5:c7;
        fixed-address myclient;               # client’s assigned IP address
        filename "myclient.netboot";  # secondary bootstrap
        next-server myserver;         # TFTP server for secondary bootstrap
        option swap-server myserver;  # NFS server for root filesystem
        option root-path "/export/myclient/root";
}
```

That **host** declaration goes inside a **subnet** declaration, which gives parameters for all hosts on the subnet that will be using DHCP, such as the "routers" (the default route), "subnet-mask", "broadcast-address", "domain-name-servers", etc. See `dhcpd.conf`(5) for details. In that example, **myclient** has an assigned IP address.

The DHCP parameters required for network bootstrapping a system will vary from platform to platform, as dictated by each system's firmware. In particular, because the DHCP is extensible, some hardware vendors have specified DHCP options to return information to requesting clients that are specific to that platform. Please see your platform's `boot`(8) for details.

TFTP

If booting a Sun system, or other system that expects to use TFTP, ensure that `inetd`(8) is configured to run `tftpd`(8). The `tftpd`(8) server should be set up to serve the directory `/tftpboot`.

If booting a SPARC system, install a copy of the appropriate diskless secondary boot loader (such as `/usr/mdec/boot` or `ofwboot.net`) in the `/tftpboot` directory. Make a link such that the boot program is accessible by a filename composed of the client's IP address in hexadecimal, a dot, and the architecture name (all upper case). For example:

```
# cd /tftpboot
# ln -s boot C0C5600C.SUN4
```

For a Sun-3 or UltraSPARC system, the filename would be just C0C5600C (these systems' firmware does not append the architecture name). The name used is architecture dependent, it simply has to match what the booting client's system firmware wishes to it to be.

If the client's system firmware fails to fetch the expected file, `tcpdump`(8) can be used to discover which filename the client is being requested. Also, examination of `tftpd`(8) log entries (typically in `/var/log/messages`) should show whether the server is hearing the client system, and what filename the client is asking for.

HP RMP

If booting an HP 300-series system, ensure that `/etc/rbootd.conf` is configured properly to transfer the boot program to the client. An entry might look like this:

```
08:00:09:01:23:E6      SYS_UBOOT      # myclient
```

The secondary bootstrap program for an HP 300-series system `SYS_UBOOT` (which may be called `uboot.lif` before installation) must be installed in the directory `/usr/mdec/rbootd`.

See the `rbootd`(8) manual page for more information.

Sun RPC BOOTPARAMS

Add **myclient** to the bootparams database in `/etc/bootparams`:

```
myclient  root=myserver:/export/myclient/root \
          swap=myserver:/export/myclient/root/swap \
          dump=myserver:/export/myclient/root/swap
```

and ensure that `rpc.bootparamd`(8) and `rpcbind`(8) are running. Both **myclient** and **myserver** must have IP addresses in the DNS or `/etc/hosts`.

Diskless Client File Systems

Build the swap file for **myclient** on the NFS server:

```
# cd /export/myclient/root
# dd if=/dev/zero of=swap bs=16k count=1024
```

This creates a 16 megabyte swap file.

Populate **myclient**'s root file system on the NFS server. How this is done depends on the client architecture and the version of the NetBSD distribution. It can be as simple as copying and modifying the server's root file system, or unpack a complete NetBSD binary distribution for the appropriate platform.

If the NFS server is going to support multiple different architectures (e.g. Alpha, PowerPC, SPARC, MIPS), then it is important to think carefully about how to lay out the NFS server's exported file systems, to share what can be shared (e.g. text files, configuration files, user home directories), and separate that which is distinct to each architecture (e.g. binary executables, libraries).

NFS

Export the client-populated file systems on the NFS server in `/etc/exports`:

```
/usr -ro myclient
# for SunOS:
# /export/myclient -rw=myclient,root=myclient
# for NetBSD:
/export/myclient -maproot=root -alldirs myclient
```

If the server and client are of the same architecture, then the client can share the server's `/usr` file system (as is done above). If not, you must build a properly fleshed out `/usr` partition for the client in some other part of the server's file system, to serve to the client.

If your server is a SPARC, and your client a Sun-3, you might create and fill `/export/usr.sun3` and then use the following `/etc/exports` lines:

```
/export/usr.sun3 -ro myclient
/export/myclient -rw=myclient,root=myclient
```

Of course, in either case you will have to have an NFS server running on the server side.

CLIENT CONFIGURATION

Copy and customize at least the following files in `/export/myclient/root`:

```
# cd /export/myclient/root/etc
# vi fstab
# cp /etc/hosts hosts
# echo 'hostname="myclient"' >> rc.conf
# echo "inet 192.197.96.12" > ifconfig.le0
```

Note that "le0" above should be replaced with the name of the network interface that the client will use for booting; the network interface name is device dependent in NetBSD.

Correct the critical mount points and the swap file in the client's `/etc/fstab` (which will be `/export/myclient/root/etc/fstab`) i.e.

```
myserver:/export/myclient/root  /    nfs  rw 0 0
myserver:/usr                   /usr nfs  rw 0 0
/swap                           none swap sw 0 0
```

Note, you *must* specify the swap file in `/etc/fstab` or it will not be used! See `swapctl`(8).

FILES

`/etc/hosts`	table of associated IP addresses and IP host names; see `hosts`(5)
`/etc/ethers`	table of associated Ethernet MAC addresses and IP host names used by `rarpd`(8); see `ethers`(5)
`/etc/bootparams`	client root pathname and swap pathname; see `bootparams`(5)
`/etc/exports`	exported NFS mount points; see `exports`(5)
`/etc/rbootd.conf`	configuration file for HP RMP; see `rbootd`(8)
`/usr/mdec/rbootd`	location of boot programs offered by `rbootd`(8)
`/tftpboot`	location of boot programs offered by `tftpd`(8)

SEE ALSO

`bootparams`(5), `dhcpd.conf`(5), `ethers`(5), `exports`(5), `fstab`(5), `hosts`(5), `networks`(5), `boot`(8), `dhcpd`(8), `mopd`(8), `mountd`(8), `nfsd`(8), `rarpd`(8), `rbootd`(8), `reboot`(8), `rpc.bootparamd`(8), `tftpd`(8)

Reverse Address Resolution Protocol, RFC, 903, June 1984.

Bootstrap Loading using TFTP, RFC, 906, June 1984.

Bootstrap Protocol, RFC, 951, September 1985.

The TFTP Protocol (Revision 2), RFC, 1350, July 1992.

Dynamic Host Configuration Protocol, RFC, 2131, March 1997.

DHCP Options and BOOTP Vendor Extensions, RFC, 2132, March 1997.

`http://www.rfc-editor.org/`

NAME

diskpart — calculate default disk partition sizes

SYNOPSIS

diskpart [**-d**] [**-p**] [**-s** *size*] *disk-type*

DESCRIPTION

diskpart is used to calculate the disk partition sizes based on the default rules used at Berkeley.

Available options and operands:

-d	An entry suitable for inclusion in the disk description file `/etc/disktab` is generated; for example, `disktab`(5).
-p	Tables suitable for inclusion in a device driver are produced.
-s *size*	The size of the disk may be limited to *size* with the **-s** option.

On disks that use `bad144`(8) type of bad-sector forwarding, space is normally left in the last partition on the disk for a bad sector forwarding table, although this space is not reflected in the tables produced. The space reserved is one track for the replicated copies of the table and sufficient tracks to hold a pool of 126 sectors to which bad sectors are mapped. For more information, see `bad144`(8). The **-s** option is intended for other controllers which reserve some space at the end of the disk for bad-sector replacements or other control areas, even if not a multiple of cylinders.

The disk partition sizes are based on the total amount of space on the disk as given in the table below (all values are supplied in units of sectors). The '`c`' partition is, by convention, used to access the entire physical disk. The device driver tables include the space reserved for the bad sector forwarding table in the '`c`' partition; those used in the disktab and default formats exclude reserved tracks. In normal operation, either the '`g`' partition is used, or the '`d`', '`e`', and '`f`' partitions are used. The '`g`' and '`f`' partitions are variable-sized, occupying whatever space remains after allocation of the fixed sized partitions. If the disk is smaller than 20 Megabytes, then **diskpart** aborts with the message "`disk too small, calculate by hand`".

Partition	20-60 MB	61-205 MB	206-355 MB	356+ MB
a	15884	15884	15884	15884
b	10032	33440	33440	66880
d	15884	15884	15884	15884
e	unused	55936	55936	307200
h	unused	unused	291346	291346

If an unknown disk type is specified, **diskpart** will prompt for the required disk geometry information.

SEE ALSO

`disktab`(5), `bad144`(8)

HISTORY

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

BUGS

Most default partition sizes are based on historical artifacts (like the RP06), and may result in unsatisfactory layouts.

When using the **-d** flag, alternative disk names are not included in the output.

NAME

dkctl — program to manipulate disks

SYNOPSIS

dkctl *device command* [*arg* [...]]

DESCRIPTION

dkctl allows a user or system administrator to manipulate and configure disks in various ways. It is used by specifying a disk to manipulate, the command to perform, and any arguments the command may require.

COMMANDS

The following commands are supported:

getcache Get and display the cache enables for the specified device.

setcache *none* | *r* | *w* | *rw* [*save*]
Set the cache enables for the specified device. The enables are as follows:

none	Disable all caches on the disk.
r	Enable the read cache, and disable all other caches on the disk.
w	Enable the write cache, and disable all other caches on the disk.
rw	Enable both the read and write caches on the disk.
save	If specified, and the cache enables are savable, saves the cache enables in the disk's non-volatile parameter storage.

synccache [*force*] Causes the cache on the disk to be synchronized, flushing all dirty write cache blocks to the media. If *force* is specified, the cache synchronization command will be issued even if the kernel does not believe that there are any dirty cache blocks in the disk's cache.

keeplabel [*yes* | *no*]
Specify to keep or drop the in-core disklabel on the last close of the disk device. (Keep if *yes* is specified, drop if *no* is specified.)

badsector *flush* | *list* | *retry*
Used for managing the kernel's bad sector list for wd(4) devices. The software bad sector list is only maintained if the option "WD_SOFTBADSECT" was specified on kernel configuration.

flush	Clears the in kernel list of bad sectors.
list	Prints out the list of bad sector ranges recorded by the kernel.
retry	Flushes the in kernel list and then retries all of the previously recorded bad sectors, causing the list to self update. This option *can only* be used with character devices.

addwedge *name startblk blkcnt ptype*
Define a "wedge" on the specified disk starting at block number *startblk* and spanning *blkcnt* blocks. The wedge will have the volume name *name* and the partition type *ptype*. The device name of the virtual block device assigned to the wedge will be displayed after the wedge has been successfully created. See dk(4) for more information about disk wedges.

delwedge *dk*	Delete the wedge specified by its device name *dk* from the specified disk.
getwedgeinfo	Display information about the specified disk wedge.
listwedges	List all of the wedges configured on the specified disk.
strategy [*name*]	Get and set the disk I/O scheduler (buffer queue strategy) on the drive. If you do not provide a *name* argument, the currently selected strategy will be shown. To set the bufq strategy, the *name* argument must be specified. *name* must be the name of one of the built-in kernel disk I/O schedulers. To get the list of supported schedulers, use the following command:

```
$ sysctl kern.bufq.strategies
```

SEE ALSO

ioctl(2), dk(4), sd(4), wd(4), disklabel(5), atactl(8), scsictl(8)

HISTORY

The **dkctl** command first appeared in NetBSD 1.6.

AUTHORS

The **dkctl** command was written by Jason R. Thorpe of Wasabi Systems, Inc.

NAME

dkscan_bsdlabel — program to create wedges from a BSD disklabel

SYNOPSIS

dkscan_bsdlabel [**-nv**] *device*

DESCRIPTION

dkscan_bsdlabel scans a disk for a BSD disklabel, which does not need to be the label variant used on the architecture currently running, or even the same endianess.

The following options are supported:

-n	No execution - list the wedges, but do not create them.
-v	Be more verbose - print additional information.

The argument *device* specifices the disk on which the disklabel is scanned and to which the wedges are added.

EXAMPLES

```
dkscan_bsdlabel -v wd1
```

Create wedges from all recognized partitions on wd1

SEE ALSO

dk(4), disklabel(5), dkctl(8)

HISTORY

The **dkscan_bsdlabel** command first appeared in NetBSD 5.0.

AUTHORS

Martin Huseman wrote the **dkscan_bsdlabel** utility. It is reusing a lot of kernel code written by Jason R. Thorpe.

NAME

dm — dungeon master

SYNOPSIS

ln −s dm *game*

DESCRIPTION

dm is a program used to regulate game playing. **dm** expects to be invoked with the name of a game that a user wishes to play. This is done by creating symbolic links to **dm**, in the directory `/usr/games` for all of the regulated games. The actual binaries for these games should be placed in a "hidden" directory, `/usr/games/hide`, that may only be accessed by the **dm** program. **dm** determines if the requested game is available and, if so, runs it. The file `/etc/dm.conf` controls the conditions under which games may be run.

The file `/etc/nogames` may be used to "turn off" game playing. If the file exists, no game playing is allowed; the contents of the file will be displayed to any user requesting a game.

FILES

`/etc/dm.conf`	configuration file
`/etc/nogames`	turns off game playing
`/usr/games/hide`	directory of ‘‘real’’ binaries
`/var/log/games.log`	game logging file

SEE ALSO

`dm.conf`(5)

HISTORY

The **dm** command appeared in 4.3BSD−Tahoe.

SECURITY CONSIDERATIONS

Two issues result from **dm** running the games setgid "games". First, all games that allow users to run UNIX commands should carefully set both the real and effective group ids immediately before executing those commands. Probably more important is that **dm** never be setgid anything but "games" so that compromising a game will result only in the user's ability to play games at will. Secondly, games which previously had no reason to run setgid and which accessed user files may have to be modified.

NAME

dmesg — display the system message buffer

SYNOPSIS

dmesg [**-M** *core*] [**-N** *system*]

DESCRIPTION

dmesg displays the contents of the system message buffer.

The options are as follows:

-M	Extract values associated with the name list from the specified core instead of the default "/dev/mem".
-N	Extract the name list from the specified system instead of the default "/netbsd".

The system message buffer is a circular buffer of a fixed size. If the buffer has been filled, the first line of the **dmesg** output may not be complete. The size of the message buffer is configurable at compile-time on most systems with the `MSGBUFSIZE` kernel option. Look for `MSGBUFSIZE` in `options`(4) for details.

FILES

`/var/run/dmesg.boot` copy of dmesg at the time of last boot.

SEE ALSO

`options`(4), `syslogd`(8)

HISTORY

The **dmesg** command appeared in 4.0BSD.

NAME

dmesgfs — refuse-based virtual file system to display devices found in dmesg

SYNOPSIS

dmesgfs [**-f**] [**-l**] [**-n** *nexus*] [**-v**] *mount_point*

DESCRIPTION

The **dmesgfs** utility can be used to mount a virtual file system which shows the tree of devices present in the computer. This tree is found by using the output of the dmesg(8) command.

The following arguments can be used:

-f present the attachment information in the virtual file system as files.

-l present the attachment information in the virtual file system as symbolic links. This is the default mode of operation.

-n *nexus*
Use the nexus name as the root of the device tree. The default value for Nexus is “mainbus0”.

-v Produce verbose output

The **dmesgfs** utility makes use of the virtdir(3) virtual directory routines.

The refuse(3) library is used to provide the file system features.

The mandatory parameter is the local mount point.

The dmesg(8) utility is used to retrieve the information.

SEE ALSO

librefuse(3), puffs(3), virtdir(3), dmesg(8).

HISTORY

The **dmesgfs** utility first appeared in NetBSD 5.0.

AUTHORS

Alistair Crooks ⟨agc@NetBSD.org⟩

NAME

dnssec−keygen − DNSSEC key generation tool

SYNOPSIS

dnssec−keygen {−a *algorithm*} {−b *keysize*} {−n *nametype*} [**−c** *class*] [**−e**] [**−f** *flag*] [**−g** *generator*] [**−h**] [**−k**] [**−p** *protocol*] [**−r** *randomdev*] [**−s** *strength*] [**−t** *type*] [**−v** *level*] {name}

DESCRIPTION

dnssec−keygen generates keys for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC <TBA\>. It can also generate keys for use with TSIG (Transaction Signatures), as defined in RFC 2845.

OPTIONS

−a *algorithm*

Selects the cryptographic algorithm. The value of **algorithm** must be one of RSAMD5 (RSA) or RSASHA1, DSA, DH (Diffie Hellman), or HMAC−MD5. These values are case insensitive.

Note 1: that for DNSSEC, RSASHA1 is a mandatory to implement algorithm, and DSA is recommended. For TSIG, HMAC−MD5 is mandatory.

Note 2: HMAC−MD5 and DH automatically set the −k flag.

−b *keysize*

Specifies the number of bits in the key. The choice of key size depends on the algorithm used. RSAMD5 / RSASHA1 keys must be between 512 and 2048 bits. Diffie Hellman keys must be between 128 and 4096 bits. DSA keys must be between 512 and 1024 bits and an exact multiple of 64. HMAC−MD5 keys must be between 1 and 512 bits.

−n *nametype*

Specifies the owner type of the key. The value of **nametype** must either be ZONE (for a DNSSEC zone key (KEY/DNSKEY)), HOST or ENTITY (for a key associated with a host (KEY)), USER (for a key associated with a user(KEY)) or OTHER (DNSKEY). These values are case insensitive.

−c *class*

Indicates that the DNS record containing the key should have the specified class. If not specified, class IN is used.

−e

If generating an RSAMD5/RSASHA1 key, use a large exponent.

−f *flag*

Set the specified flag in the flag field of the KEY/DNSKEY record. The only recognized flag is KSK (Key Signing Key) DNSKEY.

−g *generator*

If generating a Diffie Hellman key, use this generator. Allowed values are 2 and 5. If no generator is specified, a known prime from RFC 2539 will be used if possible; otherwise the default is 2.

−h

Prints a short summary of the options and arguments to **dnssec−keygen**.

−k

Generate KEY records rather than DNSKEY records.

−p *protocol*

Sets the protocol value for the generated key. The protocol is a number between 0 and 255. The default is 3 (DNSSEC). Other possible values for this argument are listed in RFC 2535 and its successors.

−r *randomdev*

Specifies the source of randomness. If the operating system does not provide a */dev/random* or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value *keyboard* indicates that keyboard input should be used.

−s *strength*
: Specifies the strength value of the key. The strength is a number between 0 and 15, and currently has no defined purpose in DNSSEC.

−t *type*
: Indicates the use of the key. **type** must be one of AUTHCONF, NOAUTHCONF, NOAUTH, or NOCONF. The default is AUTHCONF. AUTH refers to the ability to authenticate data, and CONF the ability to encrypt data.

−v *level*
: Sets the debugging level.

GENERATED KEYS

When **dnssec−keygen** completes successfully, it prints a string of the form *Knnnn.+aaa+iiiii* to the standard output. This is an identification string for the key it has generated.

- *nnnn* is the key name.
- *aaa* is the numeric representation of the algorithm.
- *iiiii* is the key identifier (or footprint).

dnssec−keygen creates two file, with names based on the printed string. *Knnnn.+aaa+iiiii.key* contains the public key, and *Knnnn.+aaa+iiiii.private* contains the private key.

The *.key* file contains a DNS KEY record that can be inserted into a zone file (directly or with a $INCLUDE statement).

The *.private* file contains algorithm specific fields. For obvious security reasons, this file does not have general read permission.

Both *.key* and *.private* files are generated for symmetric encryption algorithm such as HMAC−MD5, even though the public and private key are equivalent.

EXAMPLE

To generate a 768−bit DSA key for the domain **example.com**, the following command would be issued:

dnssec−keygen −a DSA −b 768 −n ZONE example.com

The command would print a string of the form:

Kexample.com.+003+26160

In this example, **dnssec−keygen** creates the files *Kexample.com.+003+26160.key* and *Kexample.com.+003+26160.private*

SEE ALSO

dnssec−signzone(8), BIND 9 Administrator Reference Manual, RFC 2535, RFC 2845, RFC 2539.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

dnssec−signzone − DNSSEC zone signing tool

SYNOPSIS

dnssec−signzone [**−a**] [**−c** *class*] [**−d** *directory*] [**−e** *end−time*] [**−f** *output−file*] [**−g**] [**−h**] [**−k** *key*] [**−l** *domain*] [**−i** *interval*] [**−I** *input−format*] [**−j** *jitter*] [**−N** *soa−serial−format*] [**−o** *origin*] [**−O** *output−format*] [**−p**] [**−r** *randomdev*] [**−s** *start−time*] [**−t**] [**−v** *level*] [**−z**] {zonefile} [key...]

DESCRIPTION

dnssec−signzone signs a zone. It generates NSEC and RRSIG records and produces a signed version of the zone. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a *keyset* file for each child zone.

OPTIONS

−a

Verify all generated signatures.

−c *class*

Specifies the DNS class of the zone.

−k *key*

Treat specified key as a key signing key ignoring any key flags. This option may be specified multiple times.

−l *domain*

Generate a DLV set in addition to the key (DNSKEY) and DS sets. The domain is appended to the name of the records.

−d *directory*

Look for *keyset* files in **directory** as the directory

−g

Generate DS records for child zones from keyset files. Existing DS records will be removed.

−s *start−time*

Specify the date and time when the generated RRSIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in YYYYMMDDHHMMSS notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by +N, which is N seconds from the current time. If no **start−time** is specified, the current time minus 1 hour (to allow for clock skew) is used.

−e *end−time*

Specify the date and time when the generated RRSIG records expire. As with **start−time**, an absolute time is indicated in YYYYMMDDHHMMSS notation. A time relative to the start time is indicated with +N, which is N seconds from the start time. A time relative to the current time is indicated with now+N. If no **end−time** is specified, 30 days from the start time is used as a default.

−f *output−file*

The name of the output file containing the signed zone. The default is to append *.signed* to the input file.

−h

Prints a short summary of the options and arguments to **dnssec−signzone**.

−i *interval*

When a previously signed zone is passed as input, records may be resigned. The **interval** option specifies the cycle interval as an offset from the current time (in seconds). If a RRSIG record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon, and it will be replaced.

The default cycle interval is one quarter of the difference between the signature end and start times. So if neither **end−time** or **start−time** are specified, **dnssec−signzone** generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Therefore, if any existing RRSIG records are due to expire in less than 7.5 days, they would be replaced.

−I *input−format*
: The format of the input zone file. Possible formats are **"text"** (default) and **"raw"**. This option is primarily intended to be used for dynamic signed zones so that the dumped zone file in a non−text format containing updates can be signed directly. The use of this option does not make much sense for non−dynamic zones.

−j *jitter*
: When signing a zone with a fixed signature lifetime, all RRSIG records issued at the time of signing expires simultaneously. If the zone is incrementally signed, i.e. a previously signed zone is passed as input to the signer, all expired signatures has to be regenerated at about the same time. The **jitter** option specifies a jitter window that will be used to randomize the signature expire time, thus spreading incremental signature regeneration over time.

  Signature lifetime jitter also to some extent benefits validators and servers by spreading out cache expiration, i.e. if large numbers of RRSIGs don't expire at the same time from all caches there will be less congestion than if all validators need to refetch at mostly the same time.

−n *ncpus*
: Specifies the number of threads to use. By default, one thread is started for each detected CPU.

−N *soa−serial−format*
: The SOA serial number format of the signed zone. Possible formats are **"keep"** (default), **"increment"** and **"unixtime"**.

  "keep"
  : Do not modify the SOA serial number.

  "increment"
  : Increment the SOA serial number using RFC 1982 arithmetics.

  "unixtime"
  : Set the SOA serial number to the number of seconds since epoch.

−o *origin*
: The zone origin. If not specified, the name of the zone file is assumed to be the origin.

−O *output−format*
: The format of the output file containing the signed zone. Possible formats are **"text"** (default) and **"raw"**.

−p
: Use pseudo−random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

−r *randomdev*
: Specifies the source of randomness. If the operating system does not provide a */dev/random* or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value *keyboard* indicates that keyboard input should be used.

−t
: Print statistics at completion.

−v *level*
: Sets the debugging level.

−z
: Ignore KSK flag on key when determining what to sign.

zonefile

The file containing the zone to be signed.

key

The keys used to sign the zone. If no keys are specified, the default all zone keys that have private key files in the current directory.

EXAMPLE

The following command signs the **example.com** zone with the DSA key generated in the **dnssec−keygen** man page. The zone's keys must be in the zone. If there are *keyset* files associated with child zones, they must be in the current directory. **example.com**, the following command would be issued:

dnssec−signzone −o example.com db.example.com Kexample.com.+003+26160

The command would print a string of the form:

In this example, **dnssec−signzone** creates the file *db.example.com.signed*. This file should be referenced in a zone statement in a *named.conf* file.

SEE ALSO

dnssec−keygen(8), BIND 9 Administrator Reference Manual, RFC 2535.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

dosboot — boot NetBSD/i386 from DOS

SYNOPSIS

dosboot [**−u**] [**−c** *command*] [**−i**] [*path* [**−adqsv**]]

DESCRIPTION

dosboot is an MS-DOS program. It is a boot loader for NetBSD/i386 designed to permit NetBSD to be booted directly from MS-DOS. By default, it boots a file with name `NETBSD` in the current MS-DOS directory. **dosboot** shares common code with the standard boot loader, `boot`(8).

The recognized options are:

−u Boot from a UFS filesystem instead of an MS-DOS filesystem.

−c Execute *command* (see below).

−i Enter interactive mode. **dosboot** will present a prompt, allowing input of commands (see below).

`path` Specifies the kernel file. In MS-DOS mode (default) a normal MS-DOS filename (with or without drive specification) is accepted. In UFS mode (after **−u** or after a **mode ufs** command), a path in a NetBSD filesystem is expected. By default, the file is looked up in partition 'a' of the first harddisk. Another device or partition can be specified by prepending a block device name in terms of NetBSD, followed by a colon (see `boot`(8) and examples).

−adqsv
Flags passed to the kernel, see `boot`(8).

The commands accepted after the **−c** flag or in interactive mode are:

boot [*device*:][*filename*] [**−acdqsvxz**]
Boot NetBSD. See **boot** in `boot`(8) for full details.

dev [device]
Set the default device and partition for subsequent filesystem operations. Without an operand, print the current setting. This setting doesn't apply to MS-DOS mode.

help
Print an overview about commands and arguments.

ls [`path`]
Print a directory listing of `path`, containing inode number, filename and file type. This command works in UFS mode only. path can contain a device specification.

mode *fstype*
Switch filesystem type; *fstype* should be one of *dos* or *ufs*.

quit
Leave the **dosboot** program and exit to MS-DOS.

dosboot is also installed in the `release`(7) hierarchy, under `installation/misc/dosboot.com`.

FILES

`/usr/mdec/dosboot.com`

EXAMPLES

To boot a NetBSD kernel located on MS-DOS drive D, one would issue:

```
dosboot D:\NODOS\NETBSD
```

To boot from a NetBSD floppy into single user mode, type e.g.:

```
dosboot -u fd0a:netbsd -s
```

SEE ALSO

`release`(7), `boot`(8), `w95boot`(8)

HISTORY

The NetBSD/i386 **`dosboot`** command first appeared in NetBSD 1.3.

BUGS

`dosboot` assumes that the processor is in real mode at startup. It does not work well in the presence of MS-DOS extenders and memory managers.

`dosboot` does not run directly under Windows 95. See `w95boot`(8) for a method of starting NetBSD from Windows 95 using dosboot.

In UFS mode, files can only be loaded from devices known to the BIOS. The device names do not necessarily comply with the names later used by the booted NetBSD kernel.

In MS-DOS mode, no useful boot device specification is passed to NetBSD. It is necessary to have the root device hardwired into the kernel configuration or to enter it manually.

NAME

drtest — stand-alone disk test program

DESCRIPTION

drtest is a stand-alone program used to read a disk track by track. It was primarily intended as a test program for new stand-alone drivers, but has shown useful in other contexts as well, such as verifying disks and running speed tests. For example, when a disk has been formatted (by format(8)), you can check that hard errors has been taken care of by running **drtest**. No hard errors should be found, but in many cases quite a few soft ECC errors will be reported.

While **drtest** is running, the cylinder number is printed on the console for every 10th cylinder read.

EXAMPLES

A sample run of **drtest** is shown below. In this example (using a 750), **drtest** is loaded from the root file system; usually it will be loaded from the machine's console storage device. Boldface means user input. As usual, "#" and "@" may be used to edit input.

```
>>>B/3
%%
loading hk(0,0)boot
Boot
: hk(0,0)drtest
Test program for stand-alone up and hp driver

Debugging level (1=bse, 2=ecc, 3=bse+ecc)?
Enter disk name [type(adapter,unit), e.g. hp(1,3)]? hp(0,0)
Device data: #cylinders=1024, #tracks=16, #sectors=32
Testing hp(0,0), chunk size is 16384 bytes.
(chunk size is the number of bytes read per disk access)
Start ...Make sure hp(0,0) is online
 ...
(errors are reported as they occur)
 ...
(...program restarts to allow checking other disks)
(...to abort halt machine with ^P)
```

DIAGNOSTICS

The diagnostics are intended to be self explanatory. Note, however, that the device number in the diagnostic messages is identified as *typeX* instead of *type(a,u)* where *X* = a∗8+u, e.g., hp(1,3) becomes hp11.

SEE ALSO

bad144(8), format(8)

HISTORY

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

AUTHORS

Helge Skrivervik

NAME

drvctl — tool to rescan busses and detach devices on user request

SYNOPSIS

drvctl **−r** [**−a** *attribute*] *busdevice* [locator ...]
drvctl **−d** *device*
drvctl **−l** *device*
drvctl **−p** *device*
drvctl **−Q** *device*
drvctl **−R** *device*
drvctl **−S** *device*

DESCRIPTION

The **drvctl** program works with the drvctl(4) pseudo-driver, and allows to rescan busses and to detach drivers from devices.

The following options are available:

−a Give the interface attribute where children are to be attached to (and which defines the interpretation of the locator information). This will only be needed in rare cases where the bus has multiple attributes.

−d Detach the device driver from the device given by the *device* argument.

−l List the children of the device specified by the *device* argument.

−p Get the properties for the device specified by the *device* argument. The properties are displayed as an XML property list.

−Q Resume the ancestors of *device*, *device* itself, and all of its descendants.

−R Resume both the ancestors of *device* and *device* itself.

−r Rescan the bus given by the *busdevice* argument. The scan range can be restricted by an optional *locator* list.

−S Suspend both the descendants of *device* and *device* itself.

FILES

/dev/drvctl

SEE ALSO

proplib(3), autoconf(9)

BUGS

Currently, there is no good way to get information about locator lengths and default values (which is present at kernel configuration time) out of a running kernel. Thus the locator handling is less intelligent as it could be.

NAME

dump, **rdump** — file system backup

SYNOPSIS

dump [**−0123456789aceFnStuX**] [**−B** *records*] [**−b** *blocksize*] [**−d** *density*]
[**−f** *file*] [**−h** *level*] [**−k** *read-blocksize*] [**−L** *label*] [**−l** *timeout*]
[**−r** *cachesize*] [**−s** *feet*] [**−T** *date*] [**−x** *snap-backup*] *files-to-dump*
dump [**−W | −w**]

(The 4.3BSD option syntax is implemented for backward compatibility, but is not documented here).

DESCRIPTION

dump examines files on a file system and determines which files need to be backed up. These files are copied to the given disk, tape or other storage medium for safe keeping (see the **−f** option below for doing remote backups). A dump that is larger than the output medium is broken into multiple volumes. On most media the size is determined by writing until an end-of-media indication is returned. This can be enforced by using the **−a** option.

On media that cannot reliably return an end-of-media indication (such as some cartridge tape drives) each volume is of a fixed size; the actual size is determined by the tape size and density and/or block count options below. By default, the same output file name is used for each volume after prompting the operator to change media.

files-to-dump is either a single file system, or a list of files and directories on a single file system to be backed up as a subset of the file system. In the former case, *files-to-dump* may be the device of a file system, the path to a currently mounted file system, the path to an unmounted file system listed in /etc/fstab, or, if **−F** is given, a file system image. In the latter case, certain restrictions are placed on the backup: **−u** is ignored, the only dump level that is supported is **−0**, and all of the files must reside on the same file system.

The following options are supported by **dump**:

−0−9 Dump levels. A level 0, full backup, guarantees the entire file system is copied (but see also the **−h** option below). A level number above 0, incremental backup, tells dump to copy all files new or modified since the last dump of a lower level. The default level is 9.

−a "auto-size". Bypass all tape length considerations, and enforce writing until an end-of-media indication is returned. This fits best for most modern tape drives. Use of this option is particularly recommended when appending to an existing tape, or using a tape drive with hardware compression (where you can never be sure about the compression ratio).

−B *records*
The number of kilobytes per volume, rounded down to a multiple of the blocksize. This option overrides the calculation of tape size based on length and density.

−b *blocksize*
The number of kilobytes per dump record.

−c Modify the calculation of the default density and tape size to be more appropriate for cartridge tapes.

−d *density*
Set tape density to *density*. The default is 1600 Bits Per Inch (BPI).

−e Eject tape automatically if a tape change is required.

−F Indicates that *files-to-dump* is a file system image.

−f *file*
Write the backup to *file*; *file* may be a special device file like `/dev/rst0` (a tape drive), `/dev/rsd1c` (a disk drive), an ordinary file, or '**−**' (the standard output). Multiple file names may be given as a single argument separated by commas. Each file will be used for one dump volume in the order listed; if the dump requires more volumes than the number of names given, the last file name will used for all remaining volumes after prompting for media changes. If the name of the file is of the form "host:file", or "user@host:file", **dump** writes to the named file on the remote host using `rmt`(8). Note that methods more secure than `rsh`(1) (such as `ssh`(1)) can be used to invoke `rmt`(8) on the remote host, via the environment variable RCMD_CMD. See `rcmd`(3) for more details.

−h *level*
Honor the user "nodump" flag (UF_NODUMP) only for dumps at or above the given *level*. The default honor level is 1, so that incremental backups omit such files but full backups retain them.

−k *read-blocksize*
The size in kilobyte of the read buffers, rounded up to a multiple of the file system block size. Default is 32k.

−l *timeout*
If a tape change is required, eject the tape and wait for the drive to be ready again. This is to be used with tape changers which automatically load the next tape when the tape is ejected. If after the timeout (in seconds) the drive is not ready **dump** falls back to the default behavior, and prompts the operator for the next tape.

−L *label*
The user-supplied text string *label* is placed into the dump header, where tools like `restore`(8) and `file`(1) can access it. Note that this label is limited to be at most LBLSIZE (currently 16) characters, which must include the terminating '\0'.

−n Whenever **dump** requires operator attention, notify all operators in the group "operator" using `wall`(1).

−r *cachesize*
Use that many buffers for read cache operations. A value of zero disables the read cache altogether, higher values improve read performance by reading larger data blocks from the disk and maintaining them in an LRU cache. See the **−k** option for the size of the buffers. Maximum is 512, the size of the cache is limited to 15% of the avail RAM by default.

−s *feet*
Attempt to calculate the amount of tape needed at a particular density. If this amount is exceeded, **dump** prompts for a new tape. It is recommended to be a bit conservative on this option. The default tape length is 2300 feet.

−S Display an estimate of the backup size and the number of tapes required, and exit without actually performing the dump.

−t All informational log messages printed by **dump** will have the time prepended to them. Also, the completion time interval estimations will have the estimated time at which the dump will complete printed at the end of the line.

−T *date*
Use the specified date as the starting time for the dump instead of the time determined from looking in `/etc/dumpdates`. The format of date is the same as that of `ctime`(3). This option is useful for automated dump scripts that wish to dump over a specific period of time. The **−T** option is

mutually exclusive from the **-u** option.

-u Update the file `/etc/dumpdates` after a successful dump. The format of `/etc/dumpdates` is readable by people, consisting of one free format record per line: file system name, increment level and `ctime`(3) format dump date. There may be only one entry per file system at each level. The file `/etc/dumpdates` may be edited to change any of the fields, if necessary. If a list of files or subdirectories is being dumped (as opposed to an entire file system), then **-u** is ignored.

-x *snap-backup*
Use a snapshot with *snap-backup* as backup for this dump. See `fss`(4) for more details. Snapshot support is *experimental*. Be sure you have a backup before you use it.

-X Similar to **-x** but uses a file system internal snapshot on the file system to be dumped.

-W **dump** tells the operator what file systems need to be dumped. This information is gleaned from the files `/etc/dumpdates` and `/etc/fstab`. The **-W** option causes **dump** to print out, for each file system in `/etc/dumpdates` the most recent dump date and level, and highlights those file systems that should be dumped. If the **-W** option is set, all other options are ignored, and **dump** exits immediately.

-w Is like W, but prints only those file systems which need to be dumped.

If **dump** honors the "nodump" flag (`UF_NODUMP`), files with the "nodump" flag will not be backed up. If a directory has the "nodump" flag, this directory and any file or directory under it will not be backed up.

dump requires operator intervention on these conditions: end of tape, end of dump, tape write error, tape open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **-n** option, **dump** interacts with the operator on **dump**'s control terminal at times when **dump** can no longer proceed, or if something is grossly wrong. All questions **dump** poses *must* be answered by typing "yes" or "no", appropriately.

Since making a dump involves a lot of time and effort for full dumps, **dump** checkpoints itself at the start of each tape volume. If writing that volume fails for some reason, **dump** will, with operator permission, restart itself from the checkpoint after the old tape has been rewound and removed, and a new tape has been mounted.

dump tells the operator what is going on at periodic intervals, including usually low estimates of the number of blocks to write, the number of tapes it will take, the time to completion, and the time to the tape change. The output is verbose, so that others know that the terminal controlling **dump** is busy, and will be for some time.

In the event of a catastrophic disk event, the time required to restore all the necessary backup tapes or files to disk can be kept to a minimum by staggering the incremental dumps. An efficient method of staggering incremental dumps to minimize the number of tapes follows:

- Always start with a level 0 backup, for example:

  ```
  /sbin/dump -0u -f /dev/nrst1 /usr/src
  ```

  This should be done at set intervals, say once a month or once every two months, and on a set of fresh tapes that is saved forever.

- After a level 0, dumps of active file systems are taken on a daily basis, using a modified Tower of Hanoi algorithm, with this sequence of dump levels:

  ```
  3 2 5 4 7 6 9 8 9 9 ...
  ```

  For the daily dumps, it should be possible to use a fixed number of tapes for each day, used on a weekly basis. Each week, a level 1 dump is taken, and the daily Hanoi sequence repeats beginning with 3. For weekly dumps, another fixed set of tapes per dumped file system is used, also on a cyclical basis.

After several months or so, the daily and weekly tapes should get rotated out of the dump cycle and fresh tapes brought in.

If **dump** receives a `SIGINFO` signal (see the "status" argument of `stty`(1)) whilst a backup is in progress, statistics on the amount completed, current transfer rate, and estimated finished time, will be written to the standard error output.

ENVIRONMENT

If the following environment variables exist, they are used by **dump**.

`TAPE` If no -f option was specified, **dump** will use the device specified via `TAPE` as the dump device. `TAPE` may be of the form "tapename", "host:tapename", or "user@host:tapename".

`RCMD_CMD` **dump** will use `RCMD_CMD` rather than `rsh`(1) to invoke `rmt`(8) on the remote machine.

`TIMEFORMAT`
can be used to control the format of the timestamps produced by the **−t** option. `TIMEFORMAT` is a string containing embedded formatting commands for `strftime`(3). The total formatted string is limited to about 80 characters, if this limit is exceeded then "ERROR: TIMEFORMAT too long, reverting to default" will be printed and the time format will revert to the default one. If `TIMEFORMAT` is not set then the format string defaults to "%T %Z"

FILES

`/dev/nrst0`	default tape unit to use. Taken from `_PATH_DEFTAPE` in `/usr/include/paths.h`.
`/dev/rst*`	raw SCSI tape interface
`/etc/dumpdates`	dump date records
`/etc/fstab`	dump table: file systems and frequency
`/etc/group`	to find group *operator*

DIAGNOSTICS

Many, and verbose.

dump exits with zero status on success. Startup errors are indicated with an exit code of 1; abnormal termination is indicated with an exit code of 3.

SEE ALSO

`chflags`(1), `rcmd`(1), `stty`(1), `wall`(1), `fts`(3), `rcmd`(3), `fss`(4), `st`(4), `fstab`(5), `environ`(7), `restore`(8), `rmt`(8)

HISTORY

A **dump** command appeared in Version 6 AT&T UNIX.

BUGS

Fewer than 32 read errors on the file system are ignored.

Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

dump with the **−W** or **−w** options does not report file systems that have never been recorded in `/etc/dumpdates`, even if listed in `/etc/fstab`.

When dumping a list of files or subdirectories, access privileges are required to scan the directory (as this is done via the `fts`(3) routines rather than directly accessing the file system).

It would be nice if **dump** knew about the dump sequence, kept track of the tapes scribbled on, told the operator which tape to mount when, and provided more assistance for the operator running `restore`(8).

Snapshot support is *experimental*. Be sure you have a backup before you use it.

NAME

dump_lfs, **rdump_lfs** — filesystem backup

SYNOPSIS

dump_lfs [**−0123456789aceFnStuX**] [**−B** *records*] [**−b** *blocksize*] [**−d** *density*] [**−f** *file*] [**−h** *level*] [**−k** *read-blocksize*] [**−L** *label*] [**−l** *timeout*] [**−r** *cachesize*] [**−s** *feet*] [**−T** *date*] [**−x** *snap-backup*] *files-to-dump*
dump_lfs [**−W | −w**]

(The 4.3BSD option syntax is implemented for backward compatibility, but is not documented here).

DESCRIPTION

dump_lfs examines files on a file system and determines which files need to be backed up. These files are copied to the given disk, tape or other storage medium for safe keeping (see the **−f** option below for doing remote backups). A dump that is larger than the output medium is broken into multiple volumes. On most media the size is determined by writing until an end-of-media indication is returned. This can be enforced by using the **−a** option.

On media that cannot reliably return an end-of-media indication (such as some cartridge tape drives) each volume is of a fixed size; the actual size is determined by the tape size and density and/or block count options below. By default, the same output file name is used for each volume after prompting the operator to change media.

files-to-dump is either a single file system, or a list of files and directories on a single file system to be backed up as a subset of the file system. In the former case, *files-to-dump* may be the device of a file system, the path to a currently mounted file system, the path to an unmounted file system listed in /etc/fstab, or, if **−F** is given, a file system image. In the latter case, certain restrictions are placed on the backup: **−u** is ignored, the only dump level that is supported is **−0**, and all of the files must reside on the same file system.

The following options are supported by **dump_lfs**:

−0−9 Dump levels. A level 0, full backup, guarantees the entire file system is copied (but see also the **−h** option below). A level number above 0, incremental backup, tells dump to copy all files new or modified since the last dump of a lower level. The default level is 9.

−a "auto-size". Bypass all tape length considerations, and enforce writing until an end-of-media indication is returned. This fits best for most modern tape drives. Use of this option is particularly recommended when appending to an existing tape, or using a tape drive with hardware compression (where you can never be sure about the compression ratio).

−B *records*
The number of kilobytes per volume, rounded down to a multiple of the blocksize. This option overrides the calculation of tape size based on length and density.

−b *blocksize*
The number of kilobytes per dump record.

−c Modify the calculation of the default density and tape size to be more appropriate for cartridge tapes.

−d *density*
Set tape density to *density*. The default is 1600 Bits Per Inch (BPI).

-e Eject tape automatically if a tape change is required.

-F Indicates that *files-to-dump* is a file system image.

-f *file*
Write the backup to *file*; *file* may be a special device file like /dev/rst0 (a tape drive), /dev/rsd1c (a disk drive), an ordinary file, or '**-**' (the standard output). Multiple file names may be given as a single argument separated by commas. Each file will be used for one dump volume in the order listed; if the dump requires more volumes than the number of names given, the last file name will used for all remaining volumes after prompting for media changes. If the name of the file is of the form "host:file", or "user@host:file", **dump_lfs** writes to the named file on the remote host using rmt(8). Note that methods more secure than rsh(1) (such as ssh(1)) can be used to invoke rmt(8) on the remote host, via the environment variable RCMD_CMD. See rcmd(3) for more details.

-h *level*
Honor the user "nodump" flag (UF_NODUMP) only for dumps at or above the given *level*. The default honor level is 1, so that incremental backups omit such files but full backups retain them.

-k *read-blocksize*
The size in kilobyte of the read buffers, rounded up to a multiple of the file system block size. Default is 32k.

-l *timeout*
If a tape change is required, eject the tape and wait for the drive to be ready again. This is to be used with tape changers which automatically load the next tape when the tape is ejected. If after the timeout (in seconds) the drive is not ready **dump_lfs** falls back to the default behavior, and prompts the operator for the next tape.

-L *label*
The user-supplied text string *label* is placed into the dump header, where tools like restore(8) and file(1) can access it. Note that this label is limited to be at most LBLSIZE (currently 16) characters, which must include the terminating '\0'.

-n Whenever **dump_lfs** requires operator attention, notify all operators in the group "operator" using wall(1).

-r *cachesize*
Use that many buffers for read cache operations. A value of zero disables the read cache altogether, higher values improve read performance by reading larger data blocks from the disk and maintaining them in an LRU cache. See the **-k** option for the size of the buffers. Maximum is 512, the size of the cache is limited to 15% of the avail RAM by default.

-s *feet*
Attempt to calculate the amount of tape needed at a particular density. If this amount is exceeded, **dump_lfs** prompts for a new tape. It is recommended to be a bit conservative on this option. The default tape length is 2300 feet.

-S Display an estimate of the backup size and the number of tapes required, and exit without actually performing the dump.

-t All informational log messages printed by **dump_lfs** will have the time prepended to them. Also, the completion time interval estimations will have the estimated time at which the dump will complete printed at the end of the line.

-T *date*
Use the specified date as the starting time for the dump instead of the time determined from looking in /etc/dumpdates. The format of date is the same as that of ctime(3). This option is useful

for automated dump scripts that wish to dump over a specific period of time. The **-T** option is mutually exclusive from the **-u** option.

-u Update the file `/etc/dumpdates` after a successful dump. The format of `/etc/dumpdates` is readable by people, consisting of one free format record per line: file system name, increment level and `ctime`(3) format dump date. There may be only one entry per file system at each level. The file `/etc/dumpdates` may be edited to change any of the fields, if necessary. If a list of files or subdirectories is being dumped (as opposed to an entire file system), then **-u** is ignored.

-X Prevent the log from wrapping until the dump completes, guaranteeing a consistent backup. Processes that write to the filesystem will continue as usual until the entire log is full, after which they will block until the dump is complete. This functionality is analogous to what `fss`(4) provides for other file systems. The **-x** flag is provided for compatibility with `dump`(8); it functions exactly as the **-X** flag does (its argument is ignored).

-W **dump_lfs** tells the operator what file systems need to be dumped. This information is gleaned from the files `/etc/dumpdates` and `/etc/fstab`. The **-W** option causes **dump_lfs** to print out, for each file system in `/etc/dumpdates` the most recent dump date and level, and highlights those file systems that should be dumped. If the **-W** option is set, all other options are ignored, and **dump_lfs** exits immediately.

-w Is like W, but prints only those file systems which need to be dumped.

If **dump_lfs** honors the "nodump" flag (`UF_NODUMP`), files with the "nodump" flag will not be backed up. If a directory has the "nodump" flag, this directory and any file or directory under it will not be backed up.

dump_lfs requires operator intervention on these conditions: end of tape, end of dump, tape write error, tape open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **-n** option, **dump_lfs** interacts with the operator on **dump_lfs**'s control terminal at times when **dump_lfs** can no longer proceed, or if something is grossly wrong. All questions **dump_lfs** poses *must* be answered by typing "yes" or "no", appropriately.

Since making a dump involves a lot of time and effort for full dumps, **dump_lfs** checkpoints itself at the start of each tape volume. If writing that volume fails for some reason, **dump_lfs** will, with operator permission, restart itself from the checkpoint after the old tape has been rewound and removed, and a new tape has been mounted.

dump_lfs tells the operator what is going on at periodic intervals, including usually low estimates of the number of blocks to write, the number of tapes it will take, the time to completion, and the time to the tape change. The output is verbose, so that others know that the terminal controlling **dump_lfs** is busy, and will be for some time.

In the event of a catastrophic disk event, the time required to restore all the necessary backup tapes or files to disk can be kept to a minimum by staggering the incremental dumps. An efficient method of staggering incremental dumps to minimize the number of tapes follows:

- Always start with a level 0 backup, for example:

  ```
  /sbin/dump -0u -f /dev/nrst1 /usr/src
  ```

  This should be done at set intervals, say once a month or once every two months, and on a set of fresh tapes that is saved forever.

- After a level 0, dumps of active file systems are taken on a daily basis, using a modified Tower of Hanoi algorithm, with this sequence of dump levels:

```
3 2 5 4 7 6 9 8 9 9 ...
```

For the daily dumps, it should be possible to use a fixed number of tapes for each day, used on a weekly basis. Each week, a level 1 dump is taken, and the daily Hanoi sequence repeats beginning with 3. For weekly dumps, another fixed set of tapes per dumped file system is used, also on a cyclical basis.

After several months or so, the daily and weekly tapes should get rotated out of the dump cycle and fresh tapes brought in.

If **dump_lfs** receives a SIGINFO signal (see the "status" argument of stty(1)) whilst a backup is in progress, statistics on the amount completed, current transfer rate, and estimated finished time, will be written to the standard error output.

ENVIRONMENT

If the following environment variables exist, they are used by **dump_lfs**.

TAPE
: If no -f option was specified, **dump_lfs** will use the device specified via TAPE as the dump device. TAPE may be of the form "tapename", "host:tapename", or "user@host:tapename".

RCMD_CMD
: **dump_lfs** will use RCMD_CMD rather than rsh(1) to invoke rmt(8) on the remote machine.

TIMEFORMAT
: can be used to control the format of the timestamps produced by the **-t** option. TIMEFORMAT is a string containing embedded formatting commands for strftime(3). The total formatted string is limited to about 80 characters, if this limit is exceeded then "ERROR: TIMEFORMAT too long, reverting to default" will be printed and the time format will revert to the default one. If TIMEFORMAT is not set then the format string defaults to "%T %Z"

FILES

/dev/nrst0	default tape unit to use. Taken from _PATH_DEFTAPE in /usr/include/paths.h.
/dev/rst*	raw SCSI tape interface
/etc/dumpdates	dump date records
/etc/fstab	dump table: file systems and frequency
/etc/group	to find group *operator*

DIAGNOSTICS

Many, and verbose.

dump_lfs exits with zero status on success. Startup errors are indicated with an exit code of 1; abnormal termination is indicated with an exit code of 3.

SEE ALSO

chflags(1), rcmd(1), stty(1), wall(1), fts(3), rcmd(3), st(4), fstab(5), environ(7), restore(8), rmt(8)

HISTORY

A **dump_lfs** command appeared in NetBSD 1.5.

BUGS

Fewer than 32 read errors on the file system are ignored.

Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

dump_lfs with the **-W** or **-w** options does not report file systems that have never been recorded in /etc/dumpdates, even if listed in /etc/fstab.

When dumping a list of files or subdirectories, access privileges are required to scan the directory (as this is done via the fts(3) routines rather than directly accessing the file system).

It would be nice if **dump_lfs** knew about the dump sequence, kept track of the tapes scribbled on, told the operator which tape to mount when, and provided more assistance for the operator running restore(8).

NAME

dumpfs — dump file system information

SYNOPSIS

dumpfs [**−acFimsv**] *filesys* | *device* [*...*]

DESCRIPTION

dumpfs prints out detailed information about the specified filesystems.

Options:

−a	Print details from each alternate superblock.
−c	Print details of each cylinder group.
−F	Dump a file system image from a file, not a special device.
−i	Print details of each allocated inode.
−m	Print details of the cylinder group summary.
−s	Print details of the superblock.
−v	Be even more verbose.

If none of **−a**, **−c**, **−i**, **−m**, or **−s** are given, then **dumpfs** defaults to **−cmsv**.

dumpfs is useful mostly for finding out certain file system information such as the file system block size, minimum free space percentage, and the file system level that can be upgraded with the **−c** option of `fsck_ffs`(8). All of this information is output by **dumpfs −s**.

SEE ALSO

`disktab`(5), `fs`(5), `disklabel`(8), `fsck`(8), `fsck_ffs`(8), `newfs`(8), `tunefs`(8)

HISTORY

The **dumpfs** command appeared in 4.2BSD. The **−a**, **−c**, **−i**, **−m**, and **−s** options, and the inode dump were added for NetBSD 2.0.

NAME

dumplfs — dump file system information

SYNOPSIS

dumplfs [**-adiS**] [**-b** *blkno*] [**-I** *blkno*] [**-s** *segno*] *filesys* | *device*

DESCRIPTION

dumplfs prints out the file system layout information for the LFS file system or special device specified. The listing is very long and detailed. This command is useful mostly for finding out certain file system information such as the file system block size.

The following flags are interpreted by **dumplfs**.

-a Dump the contents of all superblocks, not just the first. Superblocks appear in the dumplfs output with the segment containing them.

-b Use the block specified immediately after the flag as the super block for the filesystem.

-d Check partial segment data checksums and report mismatches. This makes **dumplfs** operate much more slowly.

-I Use the block specified immediately after the flag as the inode block containing the index file inode.

-i Dump information about the inode free list.

-S Dump information about the segment table.

-s Add the segment number immediately following the flag to a list of segments to dump. This flag may be specified more than once to dump more than one segment. The default is to dump all segments.

SEE ALSO

disktab(5), fs(5), disklabel(8), newfs_lfs(8)

HISTORY

The **dumplfs** command appeared in 4.4BSD.

NAME

edquota — edit user quotas

SYNOPSIS

edquota [**−u**] [**−f** *filesystem*] [**−p** *proto-username*] *username ...*
edquota −g [**−f** *filesystem*] [**−p** *proto-groupname*] *groupname ...*
edquota [**−u**] [**−f** *filesystem*] [**−h** *block#/inode#*] [**−s** *block#/inode#*] *username ...*
edquota −g [**−f** *filesystem*] [**−h** *block#/inode#*] [**−s** *block#/inode#*] *groupname ...*
edquota [**−u**] [**−f** *filesystem*] **−t**
edquota −g [**−f** *filesystem*] **−t**

DESCRIPTION

edquota is a quota editor. By default, or if the **−u** flag is specified, one or more users may be specified on the command line. Unless **−h** or **−s** are used, a temporary file is created for each user with an ASCII representation of the current disk quotas for that user. The list of filesystems with user quotas is determined from `/etc/fstab`. By default, quota for all quota-enabled filesystems are edited; the **−f** option can be used to restrict it to a single filesystem. An editor is invoked on the ASCII file. The editor invoked is `vi`(1) unless the environment variable `EDITOR` specifies otherwise.

The quotas may then be modified, new quotas added, etc. Setting a quota to zero indicates that no quota should be imposed. Setting a hard limit to one indicates that no allocations should be permitted. Setting a soft limit to one with a hard limit of zero indicates that allocations should be permitted on only a temporary basis (see **−t** below). The current usage information in the file is for informational purposes; only the hard and soft limits can be changed.

On leaving the editor, **edquota** reads the temporary file and modifies the binary quota files to reflect the changes made.

If the **−p** flag is specified, **edquota** will duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.

The **−h** and **−s** flags can be used to change quota limits (hard and soft, respectively) without user interaction, for usage in e.g. batch scripts. The arguments are the new block and inode number limit, separated by a slash.

If the **−g** flag is specified, **edquota** is invoked to edit the quotas of one or more groups specified on the command line. The **−p** flag can be specified in conjunction with the **−g** flag to specify a prototypical group to be duplicated among the listed set of groups.

Users are permitted to exceed their soft limits for a grace period that may be specified per filesystem. Once the grace period has expired, the soft limit is enforced as a hard limit. The default grace period for a filesystem is specified in `/usr/include/ufs/ufs/quota.h`. The **−t** flag can be used to change the grace period. By default, or when invoked with the **−u** flag, the grace period is set for all the filesystems with user quotas specified in `/etc/fstab`. When invoked with the **−g** flag the grace period is set for all the filesystems with group quotas specified in `/etc/fstab`. The grace period may be specified in days, hours, minutes, or seconds. Setting a grace period to zero indicates that the default grace period should be imposed. Setting a grace period to one second indicates that no grace period should be granted.

Only the super-user may edit quotas.

FILES

`quota.user`	at the filesystem root with user quotas
`quota.group`	at the filesystem root with group quotas
`/etc/fstab`	to find filesystem names and locations

DIAGNOSTICS

Various messages about inaccessible files; self-explanatory.

SEE ALSO

`quota`(1), `quotactl`(2), `fstab`(5), `quotacheck`(8), `quotaon`(8), `repquota`(8)

NAME

eeprom — display or modify contents of the EEPROM or openprom

SUN 3 SYNOPSIS

eeprom [**-**] [**-c**] [**-f** *device*] [**-i**] [*field*[=*value*] ...]

SPARC, SPARC64, MACPPC and PREP SYNOPSIS

eeprom [**-**] [**-c**] [**-f** *device*] [**-i**] [**-v**] [*field*[=*value*] ...]

DESCRIPTION

eeprom provides an interface for displaying and changing the contents of the EEPROM or openprom. Without any arguments, **eeprom** will list all of the known fields and their corresponding values. When given the name of a specific field, **eeprom** will display that value or set it if the field name is followed by '=' and a value. Only the super-user may modify the contents of the EEPROM or openprom.

The options are as follows:

- Commands are taken from stdin and displayed on stdout.

-c **eeprom** will fix incorrect checksum values and exit. This flag is quietly ignored on systems with an openprom.

-f *device*
On systems with an EEPROM, use *device* instead of the default `/dev/eeprom`. On systems with an openprom, use *device* instead of the default `/dev/openprom`.

-i If checksum values are incorrect, **eeprom** will ignore them and continue after displaying a warning. This flag is quietly ignored on systems with an openprom.

The following options are valid only on the SPARC and will produce an error when used on a Sun 3:

-v On systems with an openprom, be verbose when setting a value. Systems with an EEPROM are always verbose.

The **-v** option is also present on sparc64, macppc, and prep systems.

FIELDS AND VALUES

The following fields and values are for systems with an EEPROM:

hwupdate	A valid date, such as "7/12/95". The strings "today" and "now" are also acceptable.
memsize	How much memory, in megabytes, is installed in the system.
memtest	How much memory, in megabytes, is to be tested upon power-up.
scrsize	The size of the screen. Acceptable values are "1024x1024", "1152x900", "1600x1280", and "1440x1440".
watchdog_reboot	If true, the system will reboot upon reset. Otherwise, the system will fall into the monitor.
default_boot	If true, the system will use the boot device stored in `bootdev`.
bootdev	Specifies the default boot device in the form cc(x,x,x), where 'cc' is a combination of two letters such as 'sd' or 'le' and each 'x' is a hexadecimal number between 0 and ff, less the prepending '0x'.

kbdtype	This value is "0" for all Sun keyboards.
console	Specifies the console type. Valid values are "b&w", "ttya", "ttyb", "color", and "p4opt".
keyclick	If true, the keys click annoyingly.
diagdev	This is a string very similar to that used by `bootdev`. It specifies the default boot device when the diagnostic switch is turned on.
diagpath	A 40-character, NULL-terminated string specifying the kernel or standalone program to load when the diagnostic switch is turned on.
columns	An 8-bit integer specifying the number of columns on the console.
rows	An 8-bit integer specifying the number of rows on the console.
ttya_use_baud	Use the baud rate stored in `ttya_baud` instead of the default 9600.
ttya_baud	A 16-bit integer specifying the baud rate to use on ttya.
ttya_no_rtsdtr	If true, disables RTS/DTR.
ttyb_use_baud	Similar to `ttya_use_baud`, but for ttyb.
ttyb_baud	Similar to `ttya_baud`, but for ttyb.
ttyb_no_rtsdtr	Similar to `ttya_no_rtsdtr`, but for ttyb.
banner	An 80-character, NULL-terminated string to use at power-up instead of the default Sun banner.

Note that the `secure`, `bad_login`, and `password` fields are not currently supported.

Since the openprom is designed such that the field names are arbitrary, explaining them here is dubious. Below are field names and values that one is likely to see on a system with an openprom. NOTE: this list may be incomplete or incorrect due to differences between revisions of the openprom.

sunmon-compat?	If true, the old EEPROM-style interface will be used while in the monitor, rather than the openprom-style interface.
selftest-#megs	A 32-bit integer specifying the number of megabytes of memory to test upon power-up.
oem-logo	A 64bitx64bit bitmap in Sun Iconedit format. To set the bitmap, give the pathname of the file containing the image. NOTE: this property is not yet supported.
oem-logo?	If true, enables the use of the bitmap stored in `oem-logo` rather than the default Sun logo.
oem-banner	A string to use at power up, rather than the default Sun banner.
oem-banner?	If true, enables the use of the banner stored in `oem-banner` rather than the default Sun banner.
ttya-mode	A string of five comma separated fields in the format "9600,8,n,1,-". The first field is the baud rate. The second field is the number of data bits. The third field is the parity; acceptable values for parity are 'n' (none), 'e' (even), 'o' (odd), 'm' (mark), and 's' (space). The fourth field is the number of stop bits. The fifth field is the 'handshake' field; acceptable values are '-' (none), 'h' (RTS/CTS), and 's' (Xon/Xoff).

ttya-rts-dtr-off	If true, the system will ignore RTS/DTR.
ttya-ignore-cd	If true, the system will ignore carrier detect.
ttyb-mode	Similar to `ttya-mode`, but for ttyb.
ttyb-rts-dtr-off	Similar to `ttya-rts-dtr-off`, but for ttyb.
ttyb-ignore-cd	Similar to `ttya-ignore-cd`, but for ttyb.
sbus-probe-list	Four digits in the format "0123" specifying which order to probe the sbus at power-up. It is unlikely that this value should ever be changed.
screen-#columns	An 8-bit integer specifying the number of columns on the console.
screen-#rows	An 8-bit integer specifying the number of rows on the console.
auto-boot?	If true, the system will boot automatically at power-up.
watchdog-reboot?	If true, the system will reboot upon reset. Otherwise, system will fall into the monitor.
input-device	One of the strings "keyboard", "ttya", or "ttyb" specifying the default console input device.
output-device	One of the strings "screen", "ttya", or "ttyb" specifying the default console output device.
keyboard-click?	If true, the keys click annoyingly.
sd-targets	A string in the format "31204567" describing the translation of physical to logical target.
st-targets	Similar to `sd-targets`, but for tapes. The default translation is "45670123".
scsi-initiator-id	The SCSI ID of the on-board SCSI controller.
hardware-revision	A 7-character string describing a date, such as "25May95".
last-hardware-update	Similar to `hardware-revision`, describing when the CPU was last updated.
diag-switch?	If true, the system will boot and run in diagnostic mode.

FILES

/dev/eeprom	The EEPROM device on systems with an EEPROM.
/dev/openprom	The openprom device on systems with an openprom.
/dev/nvram	The nvram device on PReP systems.

BUGS

The fields and their values are not necessarily well defined on systems with an openprom. Your mileage may vary.

There are a few fields known to exist in some revisions of the EEPROM and/or openprom that are not yet supported. Most notable are those relating to password protection of the EEPROM or openprom.

Avoid gratuitously changing the contents of the EEPROM. It has a limited number of write cycles.

The date parser isn't very intelligent.

NAME

envstat — utility to handle environmental sensors

SYNOPSIS

envstat [**-DfIlSTx**] [**-c** *file*] [**-d** *device*] [**-i** *interval*]
[**-s** *device:sensor,...*] [**-w** *width*]

DESCRIPTION

envstat is a utility that handles various aspects of the sensors registered with the envsys(4) framework. It is capable of displaying sensor values as well as changing parameters and setting critical limits for the sensors.

In display mode, column widths as well as displayed sensors are fully customizable. Critical limits or other properties can be set via the configuration file. If critical limits were set previously, the display mode will show the critical limits in addition to the current values.

The following options are available:

-c Accepts a file as argument to set properties for sensors in devices registered with the framework. See the envsys.conf(5) manual page for more information.

-D Display the names of the drivers that were registered with the envsys(4) framework, one per line and some properties for the driver: refresh timeout value, for example.

-d *device*
Display only the sensors for the given *device*. This is useful when there are multiple devices registered and you want to only see results from a specific device.

-f Display temperature values in degrees Fahrenheit. The default is to display temperature values in degrees Celsius.

-I This flag skips the sensors with invalid state, these are normally shown using the "N/A" string by default.

-i *interval*
Repeat the display every *interval* seconds. Note that some devices do not provide fresh values on demand. See the individual devices manual page for meaningful values for *interval*. If not specified, or specified as 0, **envstat** produces one round of values and exits.

-l List the names of all supported sensors, one per line. Use of this flag causes **envstat** to ignore all other option flags.

-r This flag is provided for compatibility reasons and there's no need to use it. In the previous implementation, it was used to enable the row mode; this mode is now the default.

-S This flag is used to restore defaults to all devices registered with the framework. This will remove all properties that were set in the configuration file to the setting that the drivers use by default.

-s *device:sensor,...*
Restrict the display to the named sensors. The pair device and sensor description must be supplied as a comma separated list. Device as well as sensor descriptions are case sensitive.

-T Create and display max, min and average statistics for a sensor. Must be used with an *interval*, otherwise statistics cannot be collected up. Please note that to get realistic values a lower interval value should be used, but that will also increase overhead.

-w *width*
Use *width* as the column width for the output. Each column is additionally separated by a single space. The default is the length of the longest sensor name.

-x Shows the property list used by the `sysmon_envsys`(9) framework that contains details about all registered drivers and sensors.

EXAMPLES

To display the "charge" sensor of the driver *acpibat0* in one line every ten seconds:

```
$ envstat -s "acpibat0:charge" -i 10
```

To list the drivers that are currently registered with `envsys`(4):

```
$ envstat -D
```

To display the sensors of the driver *aiboost0*:

```
$ envstat -d aiboost0
```

To set all properties specified in the configuration file:

```
$ envstat -c /etc/envsys.conf
```

To remove all properties that were set previously in the configuration file:

```
$ envstat -S
```

To display statistics for all sensors and ignoring sensors with invalid states every second:

```
$ envstat -ITi1
```

SEE ALSO

`proplib`(3), `acpiacad`(4), `acpibat`(4), `acpitz`(4), `adt7463c`(4), `adt7467c`(4), `aiboost`(4), `amdtemp`(4), `aps`(4), `arcmsr`(4), `battery_pmu`(4), `cac`(4), `coretemp`(4), `envctrl`(4), `envsys`(4), `finsio`(4), `ipmi`(4), `itesio`(4), `lm`(4), `lmtemp`(4), `mfi`(4), `nsclpcsio`(4), `owtemp`(4), `pic16lc`(4), `smsc`(4), `tctrl`(4), `thinkpad`(4), `tm121temp`(4), `ug`(4), `viaenv`(4), `envsys.conf`(5)

HISTORY

envstat appeared in NetBSD 1.5. It was completely rewritten from scratch for NetBSD 5.0.

AUTHORS

The **envstat** utility that appeared in NetBSD 5.0 was written by Juan Romero Pardines. The previous version was written by Bill Squier.

NAME

error − Postfix error/retry mail delivery agent

SYNOPSIS

error [generic Postfix daemon options]

DESCRIPTION

The Postfix **error**(8) delivery agent processes delivery requests from the queue manager. Each request specifies a queue file, a sender address, the reason for non-delivery (specified as the next-hop destination), and recipient information. The reason may be prefixed with an RFC 3463-compatible detail code. This program expects to be run from the **master**(8) process manager.

Depending on the service name in master.cf, **error** or **retry**, the server bounces or defers all recipients in the delivery request using the "next-hop" information as the reason for non-delivery. The **retry** service name is supported as of Postfix 2.4.

Delivery status reports are sent to the **bounce**(8), **defer**(8) or **trace**(8) daemon as appropriate.

SECURITY

The **error**(8) mailer is not security-sensitive. It does not talk to the network, and can be run chrooted at fixed low privilege.

STANDARDS

RFC 3463 (Enhanced Status Codes)

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces and of other trouble.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **error**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

2bounce_notice_recipient (postmaster)
The recipient of undeliverable mail that cannot be returned to the sender.

bounce_notice_recipient (postmaster)
The recipient of postmaster notifications with the message headers of mail that Postfix did not deliver and of SMTP conversation transcripts of mail that Postfix did not receive.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

double_bounce_sender (double-bounce)
The sender address of postmaster notifications that are generated by the mail system.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

notify_classes (resource, software)
The list of error classes that are reported to the postmaster.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

qmgr(8), queue manager
bounce(8), delivery status reports
discard(8), Postfix discard delivery agent
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
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

eshconfig — configure Essential Communications' HIPPI network interface

SYNOPSIS

eshconfig [**−estx**] [**−b** *bytes*] [**−c** *bytes*] [**−d** *filename*] [**−i** *usecs*] [**−m** *bytes*] [**−r** *bytes*] [**−u** *filename*] [**−w** *bytes*] [*interface*]

DESCRIPTION

eshconfig is used to configure device-specific parameters and download new firmware to the Essential Communications RoadRunner-based HIPPI network interface. The interface is very sensitive to the DMA performance characteristics of the host, and so requires careful tuning to achieve reasonable performance. In addition, firmware is likely to change frequently, which necessitates a reasonably easy way to update that firmware.

Available operands for **eshconfig**:

−b *bytes*
Adjust the burst size for read (by NIC of host memory) DMA.

−c *bytes*
Adjust the burst size for write (by NIC of host memory) DMA.

−d *filename*
Filename for file to download into NIC firmware. This must be a file in the standard Essential format, with :04 preceding every line, and a tag line at the end indicating the characteristics of the firmware file.

−e Write data to EEPROM. Normally, setting tuning parameters will only persist until the system is rebooted. Setting this parameter ensures that the changes will be written to EEPROM.

−i *usecs*
Interrupt delay in microseconds.

−m *bytes*
Minimum number of bytes to DMA in one direction (read or write) before allowing a DMA in the other direction. Tuning this prevents one direction from dominating the flow of bytes, and artificially throttling the NIC.

−r *bytes*
Bytes before DMA starts for read (from host to NIC). This controls how soon the DMA is triggered; until this many bytes are requested, the DMA will not begin.

−s Show statistics for the HIPPI NIC. Repeat the option to suppress non-zero statistics.

−t Show current tuning parameters on the host.

−u *filename*
Name of file to which the NIC firmware should be uploaded. Not currently supported.

−w *bytes*
Number of bytes required before write (from NIC to host) DMA is started. Until this many bytes are ready to be written, the DMA will not start.

−x Reset the NIC. This is necessary for the HIPPI-FP support, as ifconfig(8) will no longer physically reset the NIC when the interfaces goes up and down.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exist or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

`esh`(4), `ifconfig`(8)

HISTORY

The **`eshconfig`** command first appeared in NetBSD 1.4.

NAME

etcupdate — update the configuration and startup files in `/etc`

SYNOPSIS

etcupdate [**−ahlv**] [**−p** *pager*] [**−s** {*srcdir* | *tgzdir* | *tgzfile*}] [**−t** *temproot*] [**−w** *width*]

DESCRIPTION

etcupdate is a tool that lets the administrator update the configuration and startup files in `/etc` (and some other directories like `/dev`, `/root` and `/var`) without having to manually check and modify every file. The administrator should run this script after performing an operating system update (e.g. after running `make build` in `/usr/src` or after extracting new binary distribution files) to update to the latest configuration and startup files.

etcupdate compares the new configuration files against the currently installed files. The user is given the option of installing, merging or deleting each modified or missing file. The user can also view the differences between the files. By default, it shows the differences in the unified diff format. The default format can be toggled to show the differences in unified, context, or side by side formats or an user-defined command may be used to view differences. (And if **wdiff** is installed, it can also show differences on a word by word basis.)

etcupdate also detects if the user installs certain special files and performs corresponding tasks like remaking device nodes or rebuilding a database from the `aliases`(5) file. Finally, **etcupdate** runs `postinstall`(8) to check the results.

etcupdate needs a clean set of new configuration files to compare the existing files against. These files, called the “reference files” in this manual, may be derived from either a source or binary distribution of NetBSD.

If the user is updating from sources (which is the default mode), **etcupdate** will first create a copy of the reference files by running `make distribution` in `/usr/src/etc`, installing the files to a so-called `temproot`. (See usage of the **−s** *srcdir* and **−t** *temproot* options later in this manual page.) Although this is the default mode, it is not recommended (see the “BUGS” section).

Instead of using sources, it is recommended that the user should extract one or more binary distribution sets in a special location and use those as the reference files (see usage of the **−s** *tgzdir* option later in this manual page), or specify one or more binary distribution sets directly (see usage of the **−s** *tgzfile* option later in this manual page).

The following options are available:

−a
: **etcupdate** can automatically update files which have not been modified locally. The **−a** flag instructs **etcupdate** to store MD5 checksums in `/var/etcupdate` and use these checksums to determine if there have been any local modifications.

−h
: Prints a help text.

−l
: Automatically skip files with unchanged RCS IDs. This has the effect of leaving alone files that have been altered locally but which have not been changed in the reference files. Since this works using RCS IDs, files without RCS IDs will not be skipped even if only modified locally. This flag may be used together with the **−a** flag described above.

−p *pager*
: The pager to use when displaying files. By default this is `more`(1) but it can be changed either with this option or by defining the `PAGER` variable.

−s {*srcdir* | *tgzdir* | *tgzfile*}
The location of the reference files, or the NetBSD source files used to create the reference files. This may be specified in one of three ways:

−s *srcdir* The top level directory of the NetBSD source tree. By default this is `/usr/src` but it can be changed either with this option or the `SRCDIR` variable. The reference files will be created by running "make distribution" in the *srcdir*/`etc` directory. Note that *srcdir* should refer to the top of the source directory tree; earlier versions of **etcupdate** expected *srcdir* to refer to the `etc` subdirectory within the source tree.

−s *tgzdir* A directory in which reference files have been extracted from one or more of the "`etc.tgz`" or "`xetc.tgz`" files from a binary distribution of NetBSD. The reference files from the specified directory will be copied to the `temproot` directory.

−s *tgzfile* The location of one or more set files (or "tgz files") from a binary distribution of NetBSD. Each set file is a compressed archive containing reference files. More than one set file may be specified, either by using a single **−s** option with a colon-separated list of *tgzfile* names, or by using multiple **−s** options; this is likely to be useful to specify the locations of both the `etc.tgz` and `xetc.tgz` set files. Note that file names may not contain embedded colon ('`:`') characters, because that would conflict with the use of the colon character to delimit multiple file names with a single argument. The files from the specified sets will be extracted to the `temproot` directory.

−t *temproot* Specifies the location of the `temproot` directory. This directory will be used for a temporary copy of the reference files created by running "make distribution" in the source directory specified by **−s** *srcdir*, or a temporary copy of the reference files extracted from the binary sets specified by **−s** *tgzfile*, or a temporary copy of the reference files from the directory specified by **−s** *tempdir*. By default this is `/tmp/temproot` but can be changed either with this option or the `TEMPROOT` environment variable.

−v Makes **etcupdate** verbose about its actions.

−w *width* Sets screen width used during interactive merge. By default this is the number of columns `stty`(1) reports but it can be changed either with this option or by defining the `WIDTH` variable. This is useful for `xterm`(1) users with wider shell windows.

ENVIRONMENT

`TEMPROOT`	Sets a default value for `temproot`. See **−t** above.
`SRCDIR`	The location of the NetBSD sources files. See **−s** above.
`PAGER`	The pager to use when displaying files. See **−p** above.
`WIDTH`	The screen width used during interactive merge. See **−w** above.
`IGNOREFILES`	A list of files that **etcupdate** should ignore. Files listed in this variable will never be considered for updating by **etcupdate**.

FILES

The environment variables can also be defined in the following configuration files. The user's personal configuration file settings override the global settings.

/etc/etcupdate.conf

~/.etcupdaterc

EXAMPLES

You have just upgraded your NetBSD host from 3.0 to 4.0 and now it's time to update the configuration files as well. To update the configuration files from the sources (if you have the `/usr/src/etc` directory):

```
etcupdate
```

The default location of the source files is `/usr/src` but this may be overridden with the **−s** *srcdir* command line argument:

```
etcupdate -s /some/where/src
```

To update the configuration files from binary distribution sets do something like this:

```
etcupdate -s /some/where/etc.tgz -s /some/where/xetc.tgz
```

or like this:

```
mkdir /tmp/temproot
cd /tmp/temproot
tar -xpzf /some/where/etc.tgz
tar -xpzf /some/where/xetc.tgz
etcupdate -s /tmp/temproot
```

You have modified only few files in the `/etc` directory so you would like install most of the updates without being asked. To automatically update the unmodified configuration files:

```
etcupdate -a
```

To get a better idea what's going on, use the **−v** flag:

```
etcupdate -v
```

SEE ALSO

`cmp`(1), `more`(1), `rcs`(1), `sdiff`(1), `stty`(1), `aliases`(5), `postinstall`(8)

HISTORY

The **etcupdate** command appeared in NetBSD 1.6.

In NetBSD 4.0, the **−s** *tgzfile* option was added, the **−b** *tempdir* option was converted to **−s** *tgzdir*, and the **−s** *srcdir* option was changed to refer to the top of the source directory tree rather than to the `etc` subdirectory.

AUTHORS

The script was written by Martti Kuparinen ⟨martti@NetBSD.org⟩ and improved by several other NetBSD users.

The idea for this script (including code fragments, variable names etc.) came from the FreeBSD mergemaster (by Douglas Barton). Unlike the FreeBSD mergemaster, this does not use CVS version tags by default to compare if the files need to be updated. Files are compared with `cmp`(1) as this is more reliable and the only way if the version numbers are the same even though the files are different.

BUGS

If a source directory is specified via the "**-s** *srcdir*" option (or if the `/usr/src` directory is used by default), then **etcupdate** will run "make distribution" in the `etc` subdirectory of the source directory, but it will not use the same options or environment variables that would be used during a full build of the operating system. For this reason, use of the "**-s** *srcdir*" option is not recommended, and use of the "**-s** *tgzdir*" or "**-s** *tgzfile*" options is recommended.

NAME

extattrctl — manage UFS1 extended attributes

SYNOPSIS

extattrctl start *path*
extattrctl stop *path*
extattrctl initattr [**−f**] [**−p** *path*] *attrsize attrfile*
extattrctl showattr *attrfile*
extattrctl enable *path attrnamespace attrname attrfile*
extattrctl disable *path attrnamespace attrname*

DESCRIPTION

The **extattrctl** utility is the management utility for extended attributes over the UFS1 file system. It allows the starting and stopping of extended attributes on a file system, as well as initialization of attribute backing files, and enabling and disabling of specific extended attributes on a file system.

The first argument on the command line indicates the operation to be performed. Operation must be one of the following:

start *path*
Start extended attribute support on the file system named using *path*. The file system must be a UFS1 file system, and the UFS_EXTATTR kernel option must have been enabled.

stop *path*
Stop extended attribute support on the file system named using *path*. Extended attribute support must previously have been started.

initattr [**−f**] [**−p** *path*] *attrsize attrfile*
Create and initialize a file to use as an attribute backing file. You must specify a maximum per-inode size for the attribute in bytes in *attrsize*, as well as the file where the attribute will be stored, using *attrfile*.

The **−f** argument may be used to indicate that it is alright to overwrite an existing attribute backing file; otherwise, if the target file exists, an error will be returned.

The **−p** *path* argument may be used to preallocate space for all attributes rather than relying on sparse files to conserve space. This has the advantage of guaranteeing that space will be available for attributes when they are written, preventing low disk space conditions from denying attribute service.

This file should not exist before running **initattr**.

showattr *attrfile*
Show the attribute header values in the attribute file named by *attrfile*.

enable *path attrnamespace attrname attrfile*
Enable an attribute named *attrname* in the namespace *attrnamespace* on the file system identified using *path*, and backed by initialized attribute file *attrfile*. Available namespaces are "user" and "system". The backing file must have been initialized using **initattr** before its first use. Attributes must have been started on the file system prior to the enabling of any attributes.

disable *path attrnamespace attrname*
Disable the attributed named *attrname* in namespace *attrnamespace* on the file system identified by *path*. Available namespaces are "user" and "system". The file system must have attributes started on it, and the attribute most have been enabled using **enable**.

The kernel also includes support for automatic starting of extended attributes on a file system at mount time once configured with **extattrctl**. If the kernel is built with the `UFS_EXTATTR_AUTOSTART` option, UFS will search for a `.attribute` sub-directory of the file system root during the mount operation. If found, extended attribute support will be started for the file system. UFS will then search for `system` and `user` sub-directories of the `.attribute` directory for any potential backing files and enable an extended attribute for each valid backing file with the backing file name as the attribute name.

EXAMPLES

```
extattrctl start /
```

Start extended attributes on the root file system.

```
extattrctl initattr 17 /.attribute/system/md5
```

Create an attribute backing file in `/.attribute/system/md5`, and set the maximum size of each attribute to 17 bytes, with a sparse file used for storing the attributes.

```
extattrctl enable / system md5 /.attribute/system/md5
```

Enable an attribute named md5 on the root file system, backed from the file `/.attribute/system/md5`.

```
extattrctl disable / md5
```

Disable the attribute named md5 on the root file system.

```
extattrctl stop /
```

Stop extended attributes on the root file system.

SEE ALSO

`extattr_get_file`(2), `getextattr`(8), `extattr`(9)

HISTORY

Extended attribute support was developed as part of the TrustedBSD Project, and introduced in FreeBSD 5.0 and NetBSD 4.0. It was developed to support security extensions requiring additional labels to be associated with each file or directory.

AUTHORS

Robert N M Watson

BUGS

extattrctl works only on UFS1 file systems. The kernel support for extended attribute backing files and this control program should be generalized for any file system that lacks native extended attribute support.

NAME

faithd — FAITH IPv6/v4 translator daemon

SYNOPSIS

faithd [**-dp**] [**-f** *configfile*] *service* [*serverpath* [*serverargs*]]
faithd

DESCRIPTION

faithd provides IPv6-to-IPv4 TCP relay. **faithd** must be used on an IPv4/v6 dual stack router.

When **faithd** receives TCPv6 traffic, **faithd** will relay the TCPv6 traffic to TCPv4. Destination for relayed TCPv4 connection will be determined by the last 4 octets of the original IPv6 destination. For example, if `3ffe:0501:4819:ffff::` is reserved for **faithd**, and the TCPv6 destination address is `3ffe:0501:4819:ffff::0a01:0101`, the traffic will be relayed to IPv4 destination `10.1.1.1`.

To use **faithd** translation service, an IPv6 address prefix must be reserved for mapping IPv4 addresses into. Kernel must be properly configured to route all the TCP connection toward the reserved IPv6 address prefix into the `faith`(4) pseudo interface, by using `route`(8) command. Also, `sysctl`(8) should be used to configure `net.inet6.ip6.keepfaith` to `1`.

The router must be configured to capture all the TCP traffic toward reserved IPv6 address prefix, by using `route`(8) and `sysctl`(8) commands.

faithd needs a special name-to-address translation logic, so that hostnames gets resolved into special IPv6 address prefix. For small-scale installation, use `hosts`(5). For large-scale installation, it is useful to have a DNS server with special address translation support. An implementation called **totd** is available at `http://www.vermicelli.pasta.cs.uit.no/ipv6/software.html`. Make sure you do not propagate translated DNS records to normal DNS cloud, it is highly harmful.

Daemon mode

When **faithd** is invoked as a standalone program, **faithd** will daemonize itself. **faithd** will listen to TCPv6 port *service*. If TCPv6 traffic to port *service* is found, it relays the connection.

Since **faithd** listens to TCP port *service*, it is not possible to run local TCP daemons for port *service* on the router, using `inetd`(8) or other standard mechanisms. By specifying *serverpath* to **faithd**, you can run local daemons on the router. **faithd** will invoke local daemon at *serverpath* if the destination address is local interface address, and will perform translation to IPv4 TCP in other cases. You can also specify *serverargs* for the arguments for the local daemon.

The following options are available:

-d Debugging information will be generated using `syslog`(3).

-f *configfile*
Specify a configuration file for access control. See below.

-p Use privileged TCP port number as source port, for IPv4 TCP connection toward final destination. For relaying `ftp`(1) this flag is not necessary as special program code is supplied.

faithd will relay both normal and out-of-band TCP data. It is capable of emulating TCP half close as well. **faithd** includes special support for protocols used by `ftp`(1). When translating FTP protocol, **faithd** translates network level addresses in `PORT/LPRT/EPRT` and `PASV/LPSV/EPSV` commands.

Inactive sessions will be disconnected in 30 minutes, to avoid stale sessions from chewing up resources. This may be inappropriate for some of the services (should this be configurable?).

inetd mode

When **faithd** is invoked via `inetd`(8), **faithd** will handle connection passed from standard input. If the connection endpoint is in the reserved IPv6 address prefix, **faithd** will relay the connection. Otherwise, **faithd** will invoke service-specific daemon like `telnetd`(8), by using the command argument passed from `inetd`(8).

faithd determines operation mode by the local TCP port number, and enables special protocol handling whenever necessary/possible. For example, if **faithd** is invoked via `inetd`(8) on FTP port, it will operate as a FTP relay.

Access control

To prevent malicious accesses, **faithd** implements a simple address-based access control. With `/etc/faithd.conf` (or *configfile* specified by **−f**) , **faithd** will avoid relaying unwanted traffic. The `faithd.conf` contains directives with the following format:

- *src/slen* `deny` *dst/dlen*

  If the source address of a query matches *src/slen*, and the translated destination address matches *dst/dlen*, deny the connection.

- *src/slen* `permit` *dst/dlen*

  If the source address of a query matches *src/slen*, and the translated destination address matches *dst/dlen*, permit the connection.

The directives are evaluated in sequence, and the first matching entry will be effective. If there is no match (if we reach the end of the ruleset) the traffic will be denied.

With inetd mode, traffic may be filtered by using access control functionality in `inetd`(8).

EXIT STATUS

faithd exits with `EXIT_SUCCESS` (0) on success, and `EXIT_FAILURE` (1) on error.

EXAMPLES

Before invoking **faithd**, `faith`(4) interface has to be configured properly.

```
# sysctl -w net.inet6.ip6.accept_rtadv=0
# sysctl -w net.inet6.ip6.forwarding=1
# sysctl -w net.inet6.ip6.keepfaith=1
# ifconfig faith0 create up
# route add -inet6 3ffe:501:4819:ffff:: -prefixlen 96 ::1
# route change -inet6 3ffe:501:4819:ffff:: -prefixlen 96 -ifp faith0
```

Daemon mode samples

To translate `telnet` service, and provide no local telnet service, invoke **faithd** as follows:

```
# faithd telnet
```

If you would like to provide local telnet service via `telnetd`(8) on `/usr/libexec/telnetd`, use the following command line:

```
# faithd telnet /usr/libexec/telnetd telnetd
```

If you would like to pass extra arguments to the local daemon:

```
# faithd ftp /usr/libexec/ftpd ftpd -l
```

Here are some other examples. You may need **-p** if the service checks the source port range.

```
# faithd ssh
# faithd telnet /usr/libexec/telnetd telnetd
```

inetd mode samples

Add the following lines into `inetd.conf`(5).

```
telnet  stream  faith/tcp6  nowait  root  faithd  telnetd
ftp     stream  faith/tcp6  nowait  root  faithd  ftpd -l
ssh     stream  faith/tcp6  nowait  root  faithd  /usr/sbin/sshd -i
```

`inetd`(8) will open listening sockets with enabling kernel TCP relay support. Whenever connection comes in, **faithd** will be invoked by `inetd`(8). If it the connection endpoint is in the reserved IPv6 address prefix. **faithd** will relay the connection. Otherwise, **faithd** will invoke service-specific daemon like `telnetd`(8).

Access control samples

The following illustrates a simple `faithd.conf` setting.

```
# permit anyone from 3ffe:501:ffff::/48 to use the translator,
# to connect to the following IPv4 destinations:
# - any location except 10.0.0.0/8 and 127.0.0.0/8.
# Permit no other connections.
#
3ffe:501:ffff::/48 deny 10.0.0.0/8
3ffe:501:ffff::/48 deny 127.0.0.0/8
3ffe:501:ffff::/48 permit 0.0.0.0/0
```

SEE ALSO

`faith`(4), `route`(8), `sysctl`(8)

Jun-ichiro itojun Hagino and Kazu Yamamoto, "An IPv6-to-IPv4 transport relay translator", *RFC 3142*, June 2001, ftp://ftp.isi.edu/in-notes/rfc3142.txt.

HISTORY

The **faithd** command first appeared in WIDE Hydrangea IPv6 protocol stack kit.

SECURITY CONSIDERATIONS

It is very insecure to use IP-address based authentication, for connections relayed by **faithd**, and any other TCP relaying services.

Administrators are advised to limit accesses to **faithd** using `faithd.conf`, or by using IPv6 packet filters. It is to protect **faithd** service from malicious parties and avoid theft of service/bandwidth. IPv6 destination address can be limited by carefully configuring routing entries that points to `faith`(4), using `route`(8). IPv6 source address needs to be filtered by using packet filters. Documents listed in **SEE ALSO** have more discussions on this topic.

NAME

fanoutfs — fan out file system implementation for refuse and puffs

SYNOPSIS

fanoutfs [**-v**] [**-f** *configfile*] *mount_point*

DESCRIPTION

The **fanoutfs** utility can be used to mount a number of directories under one mount point. This kind of file system is traditionally known as a fanout file system, and also as a union file system.

The mandatory parameters is the local mount point.

SEE ALSO

puffs(3), refuse(3)

HISTORY

The **fanoutfs** utility first appeared in NetBSD 5.0.

NAME

fastboot, **fasthalt** — reboot/halt the system without checking the disks

SYNOPSIS

fastboot [*boot-options*]
fasthalt [*halt-options*]

DESCRIPTION

fastboot and **fasthalt** are shell scripts which reboot or halt the system, and when next started, the system will skip the normal the file systems checks. This is done by creating a file /fastboot, then invoking the reboot(8) program. The system startup script, /etc/rc, looks for this file and, if present, skips the normal invocation of fsck(8).

SEE ALSO

halt(8), rc(8), reboot(8)

HISTORY

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

NAME

fdisk — MS-DOS partition maintenance program

SYNOPSIS

fdisk [**−afiuvBFS**] [**−0** | **−1** | **−2** | **−3**] [**−t** *disktab*] [**−T** *disktype*] [**−E** *number*] [**−b** *cylinders/heads/sectors*] [**−s** *id/start/size*[*/bootmenu*]] [**−c** *bootcode*] [**−r|w** *file*] [*device*]
fdisk −l

DESCRIPTION

The **fdisk** program is used to display or update the *master boot record* or *MBR* in the first sector (sector 0) of a disk that uses the MBR style of partitioning. The following NetBSD ports use this style of disk partitioning: amd64, arc, bebox, cobalt, hpcarm, hpcmips, hpcsh, i386, macppc, mvmeppc, netwinder, ofppc, playstation2, and prep.

The MBR contains bootable code, a partition table, an indication of which partition is 'active', and (optionally, depending on the boot code) a menu for selecting a partition to be booted. There can be at most 4 partitions defined in sector 0, one of which can be an extended partition which can be split into any number of sub-partitions.

The boot code in the MBR is usually invoked by the BIOS or firmware, and the MBR passes control to the next stage boot code stored in the first sector of the partition to be booted (the *partition boot record* or *PBR*).

After booting, NetBSD does not use the partitioning done by **fdisk**, instead it uses a NetBSD disklabel saved in sector 1 of the NetBSD partition. See mbrlabel(8) for a way of using information from the MBR to construct a NetBSD disklabel.

The standard MBR boot code will only boot the 'active' partition. However NetBSD contains additional boot programs which allow the user to interactively select which of the partitions to boot. The 'mbr_ext' code will also boot NetBSD from an extended partition but will not work on old systems that do not support LBA reads, the 'mbr_com0' and 'mbr_com0_9600' will read and write from a serial port. At the start the **fdisk** program will determine whether the disk sector 0 is valid as a boot sector. (This is determined by checking the magic number.) If not, **fdisk** will initialise the boot code as well as the partition table. During this, all four partitions will be marked empty.

The flags **−a**, **−i** or **−u** are used to indicate that the partition data is to be updated. The **fdisk** program will enter an interactive conversational mode. This mode is designed not to change any data unless you explicitly tell it to; **fdisk** selects defaults for its questions to guarantee that behaviour.

fdisk will calculate the correct *cylinder*, *head*, and *sector* values for any partition you edit. If you specify **−v** you will be asked whether you want to specify them yourself.

Finally, when all the data for the first sector has been accumulated, **fdisk** will ask if you really want to write the new partition table. Only if you reply affirmatively to this question will **fdisk** write anything to the disk.

Available options:

−0 Update partition slot 0.

−1 Update partition slot 1.

−2 Update partition slot 2.

−3 Update partition slot 3.

-a Change the active partition. In interactive mode this question will be asked after the partitions have been processed.

-b *cylinders/heads/sectors*
Specify the BIOS parameters for *cylinders*, *heads*, and *sectors*. It is used only in conjunction with the **-u** flag.

-B On an i386 or amd64 system, interactively update the boot selector settings. (The boot selector permits the user to interactively select the boot partition, and thus which operating system is run, at system boot time. See mbr(8) for more information.)

-c *bootcode*
Specify the filename that **fdisk** should read the bootcode from. If the name of a directory is specified, then **fdisk** will look for files with the default names in that directory. The default is to read from /usr/mdec/mbr, /usr/mdec/mbr_bootsel or /usr/mdec/mbr_ext depending on whether *bootmenu* was specified for any partitions on an i386 machine, and leave the bootcode empty for other machines.

-E *number*
Update extended partition *number*. If the specified extended partition doesn't exist an additional extended partition will be created.

-f Run **fdisk** in a non-interactive mode. In this mode, you can only change the disk parameters by using the **-b** flag. This is provided only so scripts or other programs may use **fdisk** as part of an automatic installation process.

Using the **-f** flag with **-u** makes it impossible to specify the starting and ending *cylinder*, *head*, and *sector* fields. They will be automatically computed using the BIOS geometry.

If **-u** and **-s** are specified then the details of the specified partition will be changed. Any other partitions which overlap the requested part of the disk will be silently deleted.

-F Indicate that *device* is a regular file. Unless the geometry of *device* is told to **fdisk** by **-T** *disktype*, **fdisk** will count the 512-byte sectors in *device* and produce a fake geometry.

-i Explicitly request initialisation of the master boot code (similar to what **fdisk /mbr** does under MS-DOS), even if the magic number in the first sector is ok. The partition table is left alone by this (but see above).

-l Lists known *sysid* values and exit.

-r *file*
Read the boot record from file *file* instead of the specified disk. The geometry information used is still that of the disk volume. Any changes are written back to the file.

-s *id/start/size*[*/bootmenu*]
Specify the partition *id*, *start*, *size*, and optionally *bootmenu*. This flag requires the use of a partition selection flag (**-0**, **-1**, **-2**, **-3**, or **-E** *number*)

-S When used with no other flags print a series of /bin/sh commands for setting variables to the partition information. This could be used by installation scripts.

-t *disktab*
Read *disktype* from the named disktab(5) file instead of from /etc/disktab.

-T *disktype*
Use the disklabel *disktype* instead of the disklabel on *device*.

−u Display the partitions and interactively ask which one you want to edit. **fdisk** will step through each field showing the old value and asking for a new one. The *start* and *size* can be specified in blocks (nn), cylinders (nnc), megabytes (nnm), or gigabytes (nng), values in megabytes and gigabytes will be rounded to the nearest cylinder boundary. The *size* may be specified as *$* in which case the partition will extend to the end of the available free space.

fdisk will not allow you to create partitions which overlap.

If *bootmenu* is specified for any partition **fdisk** will determine whether the installed boot code supports the bootselect code, if it doesn't you will be asked whether you want to install the required boot code. To remove a *bootmenu* label, simply press ⟨space⟩ followed by ⟨return⟩.

−v Be more verbose, specifying **−v** more than once may increase the amount of output.

Using **−v** with **−u** allows the user to change more parameters than normally permitted.

−w *file*
Write the modified partition table to file *file* instead of the disk.

When called with no arguments, it prints the partition table. An example follows:

```
Disk: /dev/rwd0d
NetBSD disklabel disk geometry:
cylinders: 16383, heads: 16, sectors/track: 63 (1008 sectors/cylinder)
total sectors: 40032696

BIOS disk geometry:
cylinders: 1023, heads: 255, sectors/track: 63 (16065 sectors/cylinder)
total sectors: 40032696

Partition table:
0: NetBSD (sysid 169)
   bootmenu: net 1.5.
   start 4209030, size 8289540 (4048 MB, Cyls 262-778), Active
1: Primary DOS with 32 bit FAT (sysid 11)
   bootmenu: win98
   start 63, size 4208967 (2055 MB, Cyls 0-262)
2: NetBSD (sysid 169)
   bootmenu: current
   start 32515560, size 7517136 (3670 MB, Cyls 2024-2491/234/40)
3: Ext. partition - LBA (sysid 15)
   start 12498570, size 20016990 (9774 MB, Cyls 778-2024)
Extended partition table:
E0: NetBSD (sysid 169)
   bootmenu: test
   start 12498633, size 12305727 (6009 MB, Cyls 778-1544)
E1: Primary DOS with 32 bit FAT (sysid 11)
   start 24804423, size 4096512 (2000 MB, Cyls 1544-1799)
E2: Primary DOS with 32 bit FAT (sysid 11)
   start 28900998, size 3614562 (1765 MB, Cyls 1799-2024)
Bootselector enabled, infinite timeout.
First active partition: 0
```

This example disk is divided into four partitions the last of which is an extended partition. The sub-partitions of the extended partition are also shown. In this case there is no free space in either the disk or in the extended partition.

The various fields in each partition entry are:

> *ptn_number*: *id_name* (sysid *id_number*)
>
> > bootmenu: *bootmenu*
> >
> > start *start*, size *size* (*MB* MB, Cyls *first*-*next*) [, Active]

ptn_number
: is the number of the partition.

id_name
: is the name of the filesystem type or operating system that uses this partition.

id_number
: is the number that identifies the partition type. 169 decimal is used for NetBSD partitions, 15 decimal to create an extended partition and 0 to mark a partition as unused. Use **fdisk −l** to list the known partition types.

bootmenu
: is the menu prompt output by the interactive boot code for this partition. This line is omitted if the prompt is not defined.

start, *size*
: are the start address and size of the partition in sectors.

MB
: is the size of the partition in megabytes.

first, *next*
: are the bounds of this partition displayed as cylinder/head/sector. If the partition starts (or ends) on a cylinder boundary the head and sector values are omitted. If **−v** is not specified the start of extended partitions and the first partition on the disk are rounded down to include the mandatory red tape in the preceding track.

Active
: is output if this is the active partition.

If the **−v** flag is specified, the beginning and end of each partition are also displayed as follows:

> beg: cylinder *cylinder*, head *head*, sector *sector*
>
> end: cylinder *cylinder*, head *head*, sector *sector*

cylinder, *head*, *sector*
: are the beginning or ending address of a partition.

: *Note:* these numbers are read from the bootblock, so are the values calculated by a previous run of **fdisk**.

fdisk attempts to check whether each partition is bootable, by checking the magic number and some other characteristics of the first sector of each partition (the PBR). If the partition does not apear to be bootable, **fdisk** will print a line containing "PBR is not bootable" followed by an error message. If the partition is bootable, and if the **−v** flag is specified, **fdisk** will print "PBR appears to be bootable". If the **−v** flag is specified more than once, **fdisk** will print the heading "Information from PBR:" followed by one or more lines of information gleaned from the PBR; this additional information may be incorrect or misleading, because different operating systems use different PBR formats. Note that, even if no errors are reported, an attempt to boot from the partition might fail. NetBSD partitions may be made bootable using `installboot`(8).

NOTES

This program is only available (and useful) on systems with PC-platform-style MBR partitioning.

Traditionally the partition boundaries should be on cylinder boundaries using the BIOS geometry, with the exception of the first partition, which traditionally begins in the second track of the first cylinder (cylinder 0, head 1, sector 1). Although the BIOS geometry is typically different from the geometry reported by the drive, neither will match the actual physical geometry for modern disks (the actual geometry will vary across the disk). Keeping the partition boundaries on cylinder boundaries makes partitioning a driver easier as only relatively small numbers need be entered.

The automatic calculation of the starting cylinder and other parameters uses a set of figures that represent what the BIOS thinks is the geometry of the drive. The default values should be correct for the system on which **fdisk** is run, however if you move the disk to a different system the BIOS of that system might use a different geometry translation.

If you run the equivalent of **fdisk** on a different operating system then the *bootmenu* strings associated with extended partitions may be lost.

Editing an existing partition is risky, and may cause you to lose all the data in that partition.

You should run this program interactively once or twice to see how it works. This is completely safe as long as you answer the last question in the negative. You can also specify **-w** *file* to write the output to a file and later specify **-r** *file* to read back the updated information. This can be done without having write access to the disk volume.

FILES

`/usr/mdec/mbr`	Default location of i386 bootcode
`/usr/mdec/mbr_bootsel`	Default location of i386 bootselect code
`/usr/mdec/mbr_ext`	Default location of i386 bootselect for extended partitions

SEE ALSO

disktab(5), boot(8), disklabel(8), installboot(8), mbr(8), mbrlabel(8)

BUGS

The word ‘partition’ is used to mean both an MBR partition and a NetBSD partition, sometimes in the same sentence.

There are subtleties that the program detects that are not explained in this manual page.

NAME

fingerd — remote user information server

SYNOPSIS

fingerd [**-8ghlmpSsu**] [**-P** *filename*]

DESCRIPTION

fingerd is a simple protocol based on RFC 1288 that provides an interface to the Name and Finger programs at several network sites. The program is supposed to return a friendly, human-oriented status report on either the system at the moment or a particular person in depth. There is no required format and the protocol consists mostly of specifying a single "command line".

fingerd is started by inetd(8), which listens for TCP requests at port 79. Once handed a connection, **fingerd** reads a single command line terminated by a ⟨CRLF⟩ which it then passes to finger(1). **fingerd** closes its connections as soon as the output is finished.

If the line is null (i.e., just a ⟨CRLF⟩ is sent) then finger(1) returns a "default" report that lists all people logged into the system at that moment.

If a user name is specified (e.g., eric⟨CRLF⟩) then the response lists more extended information for only that particular user, whether logged in or not. Allowable "names" in the command line include both "login names" and "user names". If a name is ambiguous, all possible derivations are returned.

The following options may be passed to **fingerd** as server program arguments in /etc/inetd.conf:

-8	Enable 8-bit output.
-g	Do not show any gecos information besides the users' real names.
-h	Display the name of the remote host in short mode, instead of the office location and office phone.
-l	Enable logging. The name of the host originating the query, and the actual request is reported via syslog(3) at LOG_NOTICE priority. A request of the form '/W' or '/w' will return long output. Empty requests will return all currently logged in users. All other requests look for specific users. See RFC 1288 for details.
-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.
-P *filename*	Use an alternate program as the local information provider. The default local program executed by **fingerd** is finger(1). By specifying a customized local server, this option allows a system manager to have more control over what information is provided to remote sites.
-p	Prevents finger(1) from displaying the contents of the ".plan" and ".project" files.
-S	Prints user information in short mode, one line per user. This overrides the "Whois switch" that may be passed in from the remote client.
-s	Disable forwarding of queries to other remote hosts.
-u	Queries without a user name are rejected.

SEE ALSO

finger(1), inetd(8)

HISTORY

The **fingerd** command appeared in 4.3BSD.

BUGS

Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. **fingerd** should be taught to filter out IAC's and perhaps even respond negatively (IAC WON'T) to all option commands received.

NAME

fixmount − fix remote mount entries

SYNOPSIS

fixmount [**−adervq**] [**−h** *name*] *host* ...

DESCRIPTION

fixmount is a variant of **showmount**(8) that can delete bogus mount entries in remote **mountd**(8) daemons. The actions specified by the options are performed for each *host* in turn.

OPTIONS

−a −d −e
These options work as in **showmount**(8) except that only entries pertaining to the local host are printed.

−r Removes those remote mount entries on *host* that do not correspond to current mounts, i.e., which are left-over from a crash or are the result of improper mount protocol. The actuality of mounts is verified using the entries in **/etc/mtab**.

−v Verify remote mounts. Similar to **−r** except that only a notification message is printed for each bogus entry found. The remote mount table is not changed.

−A Issues a command to the remote mountd declaring that ALL of its filesystems have been unmounted. This should be used with caution, as it removes all remote mount entries pertaining to the local system, whether or not any filesystems are still mounted locally.

−q Be quiet. Suppresses error messages due to timeouts and "Program not registered", i.e., due to remote hosts not supporting RPC or not running mountd.

−h *name*
Pretend the local hostname is *name*. This is useful after the local hostname has been changed and rmtab entries using the old name remain on a remote machine. Unfortunately, most mountd's won't be able to successfully handle removal of such entries, so this option is useful in combination with **−v** only.
This option also saves time as comparisons of remotely recorded and local hostnames by address are avoided.

FILES

/etc/mtab	List of current mounts.
/etc/rmtab	Backup file for remote mount entries on NFS server.

SEE ALSO

showmount(8), **mtab**(5), **rmtab**(5), **mountd**(8C).

''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

BUGS

No attempt is made to verify the information in **/etc/mtab** itself.

Since swap file mounts are not recorded in **/etc/mtab**, a heuristic specific to SunOS is used to determine whether such a mount is actual (replacing the string "swap" with "root" and verifying the resulting path).

Symbolic links on the server will cause the path in the remote entry to differ from the one in **/etc/mtab**. To catch those cases, a filesystem is also deemed mounted if its *local* mount point is identical to the remote entry. I.e., on a SunOS diskless client, **server:/export/share/sunos.4.1.1** is actually **/usr/share**. Since the local mount point is **/usr/share** as well this will be handled correctly.

There is no way to clear a stale entry in a remote mountd after the local hostname (or whatever reverse name resolution returns for it) has been changed. To take care of these cases, the remote /etc/rmtab file has to be edited and mountd restarted.

The RPC timeouts for mountd calls can only be changed by recompiling. The defaults are 2 seconds for client handle creation and 5 seconds for RPC calls.

AUTHORS

Andreas Stolcke <stolcke@icsi.berkeley.edu>.

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

flush − Postfix fast flush server

SYNOPSIS

flush [generic Postfix daemon options]

DESCRIPTION

The **flush**(8) server maintains a record of deferred mail by destination. This information is used to improve the performance of the SMTP **ETRN** request, and of its command-line equivalent, "**sendmail -qR**" or "**postqueue -f**". This program expects to be run from the **master**(8) process manager.

The record is implemented as a per-destination logfile with as contents the queue IDs of deferred mail. A logfile is append-only, and is truncated when delivery is requested for the corresponding destination. A destination is the part on the right-hand side of the right-most **@** in an email address.

Per-destination logfiles of deferred mail are maintained only for eligible destinations. The list of eligible destinations is specified with the **fast_flush_domains** configuration parameter, which defaults to **$relay_domains**.

This server implements the following requests:

add *sitename queueid*
: Inform the **flush**(8) server that the message with the specified queue ID is queued for the specified destination.

send_site *sitename*
: Request delivery of mail that is queued for the specified destination.

send_file *queueid*
: Request delivery of the specified deferred message.

refresh
: Refresh non-empty per-destination logfiles that were not read in **$fast_flush_refresh_time** hours, by simulating send requests (see above) for the corresponding destinations.

  Delete empty per-destination logfiles that were not updated in **$fast_flush_purge_time** days.

  This request completes in the background.

purge
: Do a **refresh** for all per-destination logfiles.

SECURITY

The **flush**(8) server is not security-sensitive. It does not talk to the network, and it does not talk to local users. The fast flush server can run chrooted at fixed low privilege.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

Fast flush logfiles are truncated only after a "send" request, not when mail is actually delivered, and therefore can accumulate outdated or redundant data. In order to maintain sanity, "refresh" must be executed periodically. This can be automated with a suitable wakeup timer setting in the **master.cf** configuration file.

Upon receipt of a request to deliver mail for an eligible destination, the **flush**(8) server requests delivery of all messages that are listed in that destination's logfile, regardless of the recipients of those messages. This is not an issue for mail that is sent to a **relay_domains** destination because such mail typically only has recipients in one domain.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **flush**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

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.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

fast_flush_domains ($relay_domains)
Optional list of destinations that are eligible for per-destination logfiles with mail that is queued to those destinations.

fast_flush_refresh_time (12h)
The time after which a non-empty but unread per-destination "fast flush" logfile needs to be refreshed.

fast_flush_purge_time (7d)
The time after which an empty per-destination "fast flush" logfile is deleted.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

parent_domain_matches_subdomains (see 'postconf -d' output)
What Postfix features match subdomains of "domain.tld" automatically, instead of requiring an explicit ".domain.tld" pattern.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

FILES

/var/spool/postfix/flush, "fast flush" logfiles.

SEE ALSO

smtpd(8), SMTP server
qmgr(8), queue manager
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

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

This service was introduced with Postfix version 1.0.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

format — how to format disks and tapes

DESCRIPTION

Currently, there are no "native" NetBSD media formatting utilities. Formatting of both disks and cartridge tapes must be done either stand-alone or under HP-UX using the *mediainit* utility distributed by HP. Note that HP-brand cartridge tapes come pre-formatted, and HP disks are supposed to.

HISTORY

The **format** utility first appeared in 4.4BSD.

NAME

format — how to format disk packs

DESCRIPTION

There are two ways to format disk packs. The simplest is to use the **format** program. The alternative is to use the DEC standard formatting software which operates under the DEC diagnostic supervisor. This manual page describes the operation of **format**, then concludes with some remarks about using the DEC formatter.

format is a standalone program used to format and check disks prior to constructing file systems. In addition to the formatting operation, **format** records any bad sectors encountered according to DEC standard 144. Formatting is performed one track at a time by writing the appropriate headers and a test pattern and then checking the sector by reading and verifying the pattern, using the controller's ECC for error detection. A sector is marked bad if an unrecoverable media error is detected, or if a correctable ECC error too many bits in length is detected (such errors are indicated as "ECC" in the summary printed upon completing the format operation). After the entire disk has been formatted and checked, the total number of errors are reported, any bad sectors and skip sectors are marked, and a bad sector forwarding table is written to the disk in the first five even numbered sectors of the last track. It is also possible to reformat sections of the disk in units of tracks. **format** may be used on any UNIBUS or MASSBUS drive supported by the *up* and *hp* device drivers which uses 4-byte headers (everything except RP's).

The test pattern used during the media check may be selected from one of: 0xf00f (RH750 worst case), 0xec6d (media worst case), and 0xa5a5 (alternating 1's and 0's). Normally the media worst case pattern is used.

format also has an option to perform an extended "severe burn-in", which makes a number of passes using different patterns. The number of passes can be selected at run time, up to a maximum of 48, with provision for additional passes or termination after the preselected number of passes. This test runs for many hours, depending on the disk and processor.

Each time **format** is run to format an entire disk, a completely new bad sector table is generated based on errors encountered while formatting. The device driver, however, will always attempt to read any existing bad sector table when the device is first opened. Thus, if a disk pack has never previously been formatted, or has been formatted with different sectoring, five error messages will be printed when the driver attempts to read the bad sector table; these diagnostics should be ignored.

Formatting a 400 megabyte disk on a MASSBUS disk controller usually takes about 20 minutes. Formatting on a UNIBUS disk controller takes significantly longer. For every hundredth cylinder formatted **format** prints a message indicating the current cylinder being formatted. (This message is just to reassure people that nothing is amiss.)

format uses the standard notation of the standalone I/O library in identifying a drive to be formatted. A drive is specified as *zz(x,y)*, where *zz* refers to the controller type (either *hp* or *up*), *x* is the unit number of the drive; 8 times the UNIBUS or MASSBUS adaptor number plus the MASSBUS drive number or UNIBUS drive unit number; and *y* is the file system partition on drive *x* (this should always be 0). For example, "hp(1,0)" indicates that drive 1 on MASSBUS adaptor 0 should be formatted; while "up(10,0)" indicates that UNIBUS drive 2 on UNIBUS adaptor 1 should be formatted.

Before each formatting attempt, **format** prompts the user in case debugging should be enabled in the appropriate device driver. A carriage return disables debugging information.

format should be used prior to building file systems (with `newfs`(8) to ensure that all sectors with uncorrectable media errors are remapped. If a drive develops uncorrectable defects after formatting, either `bad144`(8) or `badsect`(8) should be able to avoid the bad sectors.

EXAMPLES

A sample run of **format** is shown below. In this example (using a VAX-11/780), **format** is loaded from the console floppy; on an 11/750 **format** will be loaded from the root file system with `boot`(8) following a “B/3” command. Boldface means user input. As usual, “#” and “@” may be used to edit input.

```
>>>L FORMAT
                LOAD DONE, 00004400 BYTES LOADED
>>>S 2
Disk format/check utility

Enable debugging (0=none, 1=bse, 2=ecc, 3=bse+ecc)? 0
Device to format? hp(8,0)
(error messages may occur as old bad sector table is read)
Formatting drive hp0 on adaptor 1: verify (yes/no)? yes
Device data: #cylinders=842, #tracks=20, #sectors=48
Starting cylinder (0):
Starting track (0):
Ending cylinder (841):
Ending track (19):
Available test patterns are:
                1 - (f00f) RH750 worst case
                2 - (ec6d) media worst case
                3 - (a5a5) alternating 1’s and 0’s
                4 - (ffff) Severe burnin (up to 48 passes)
Pattern (one of the above, other to restart)? 2
Maximum number of bit errors to allow for soft ECC (3):
Start formatting...make sure the drive is online
 ...
(soft ecc’s and other errors are reported as they occur)
 ...
(if 4 write check errors were found, the program terminates like this...)
 ...
Errors:
Bad sector: 0
Write check: 4
Hard ECC: 0
Other hard: 0
Marked bad: 0
Skipped: 0
Total of 4 hard errors revectored.
Writing bad sector table at block 808272
(808272 is the block # of the first block in the bad sector table)
Done
(...program restarts to allow formatting other disks)
(...to abort halt machine with ^P)
```

DIAGNOSTICS

The diagnostics are intended to be self explanatory.

USING DEC SOFTWARE TO FORMAT

Warning: These instructions are for people with 11/780 CPUs. The steps needed for 11/750 or 11/730 CPU's are similar, but not covered in detail here.

The formatting procedures are different for each type of disk. Listed here are the formatting procedures for RK07's, RP0X, and RM0X disks.

You should shut down UNIX and halt the machine to do any disk formatting. Make certain you put in the pack you want formatted. It is also a good idea to spin down or write protect the disks you don't want to format, just in case.

Formatting an RK07

Load the console floppy labeled, "RX11 VAX DSK LD DEV #1" in the console disk drive, and type the following commands:

```
>>>BOOT
DIAGNOSTIC SUPERVISOR.  ZZ-ESSAA-X5.0-119  23-JAN-1980 12:44:40.03
DS>ATTACH DW780 SBI DW0 3 5
DS>ATTACH RK611 DMA
DS>ATTACH RK07 DW0 DMA0
DS>SELECT DMA0
DS>LOAD EVRAC
DS>START/SEC:PACKINIT
```

Formatting an RP0X

Follow the above procedures except that the ATTACH and SELECT lines should read:

```
DS>ATTACH RH780 SBI RH0 8 5
DS>ATTACH RP0X RH0 DBA0              (RP0X is, e.g., RP06)
DS>SELECT DBA0
```

This is for drive 0 on mba0; use 9 instead of 8 for mba1, etc.

Formatting an RM0X

Follow the above procedures except that the ATTACH and SELECT lines should read:

```
DS>ATTACH RH780 SBI RH0 8 5
DS>ATTACH RM0X RH0 DRA0
DS>SELECT DRA0
```

Don't forget to put your UNIX console floppy back in the floppy disk drive.

SEE ALSO

`bad144`(8), `badsect`(8), `newfs`(8)

BUGS

An equivalent facility should be available which operates under a running UNIX system.

It should be possible to reformat or verify part or all of a disk, then update the existing bad sector table.

NAME

fsck — file system consistency check and interactive repair

SYNOPSIS

fsck [**-dfnPpqvy**] [**-l** *maxparallel*] [**-T** *fstype:fsoptions*] [**-t** *fstype*]
[special | node ...]

DESCRIPTION

The **fsck** command invokes file system-specific programs to check the special devices listed in the fstab(5) file or in the command line for consistency.

It is normally used in the script /etc/rc during automatic reboot. If no file systems are specified, and ‘‘preen’’ mode is enabled (**-p** option) **fsck** reads the table /etc/fstab to determine which file systems to check, in what order. Only partitions in fstab that are mounted ‘‘rw,’’ ‘‘rq’’ or ‘‘ro’’ and that have non-zero pass number are checked. File systems with pass number 1 (normally just the root file system) are checked one at a time. When pass 1 completes, all remaining file systems are checked, running one process per disk drive. By default, file systems which are already mounted read-write are not checked. The disk drive containing each file system is inferred from the longest prefix of the device name that ends in a digit; the remaining characters are assumed to be the partition designator.

The options are as follows:

-d Debugging mode. Just print the commands without executing them.

-f Force checking of file systems, even when they are marked clean (for file systems that support this), or when they are mounted read-write.

-l *maxparallel*
Limit the number of parallel checks to the number specified in the following argument. By default, the limit is the number of disks, running one process per disk. If a smaller limit is given, the disks are checked round-robin, one file system at a time.

-n Causes **fsck** to assume no as the answer to all operator questions, except "CONTINUE?".

-P Display a progress meter for each file system check. This option also disables parallel checking. Note that progress meters are not supported by all file system types.

-p Enter preen mode. In preen mode, **fsck** will check all file systems listed in /etc/fstab according to their pass number, and will make minor repairs without human intervention.

-q Quiet mode, do not output any messages for clean filesystems.

-T *fstype:fsoptions*
List of comma separated file system specific options for the specified file system type, in the same format as mount(8).

-t *fstype*
Invoke **fsck** only for the comma separated list of file system types. If the list starts with “no” then invoke **fsck** for the file system types that are not specified in the list.

-v Print the commands before executing them.

-y Causes **fsck** to assume yes as the answer to all operator questions.

EXIT STATUS

fsck exits with 0 on success. Any major problems will cause **fsck** to exit with the following non-zero exit(3) codes, so as to alert any invoking program or script that human intervention is required.

`1`	Usage problem.
`2`	Unresolved errors while checking the filesystem. Re-running **fsck** on the filesystem(s) is required.
`4`	The root filesystem was changed in the process of checking, and updating the mount was unsuccessful. A reboot (without sync) is required.
`8`	The filesystem check has failed, and a subsequent check is required that will require human intervention.
`12`	**fsck** exited because of the result of a signal (usually `SIGINT` or `SIGQUIT` from the terminal).

FILES

`/etc/fstab` file system table

SEE ALSO

`fstab`(5), `fsck_ext2fs`(8), `fsck_ffs`(8), `fsck_lfs`(8), `fsck_msdos`(8), `mount`(8)

NAME

fsck_ext2fs — EXT2 File System consistency check and interactive repair

SYNOPSIS

fsck_ext2fs [**-b** *block#*] [**-c** *level*] [**-d**] [**-f**] [**-m** *mode*] [**-p**] [**-y**] [**-n**]
filesystem ...

DESCRIPTION

fsck_ext2fs performs interactive filesystem consistency checks and repair for each of the filesystems specified on the command line. It is normally invoked from fsck(8).

The kernel takes care that only a restricted class of innocuous filesystem inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

Unreferenced inodes
Link counts in inodes too large
Missing blocks in the free map
Blocks in the free map also in files
Counts in the super-block wrong

These are the only inconsistencies that **fsck_ext2fs** in ‘‘preen’’ mode (with the **-p** option) will correct; if it encounters other inconsistencies, it exits with an abnormal return status. For each corrected inconsistency one or more lines will be printed identifying the filesystem on which the correction will take place, and the nature of the correction. After successfully correcting a filesystem, **fsck_ext2fs** will print the number of files on that filesystem and the number of used and free blocks.

If sent a QUIT signal, **fsck_ext2fs** will finish the filesystem checks, then exit with an abnormal return status.

Without the **-p** option, **fsck_ext2fs** audits and interactively repairs inconsistent conditions for filesystems. If the filesystem is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that some of the corrective actions which are not correctable under the **-p** option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond yes or no. If the operator does not have write permission on the filesystem **fsck_ext2fs** will default to a **-n** action.

The following flags are interpreted by **fsck_ext2fs**.

-b Use the block specified immediately after the flag as the super block for the filesystem. Block 8193 is usually an alternate super block.

-d Print debugging output.

-f Force checking of file systems. Normally, if a file system is cleanly unmounted, the kernel will set a “clean flag” in the file system superblock, and **fsck_ext2fs** will not check the file system. This option forces **fsck_ext2fs** to check the file system, regardless of the state of the clean flag.

-m Use the mode specified in octal immediately after the flag as the permission bits to use when creating the lost+found directory rather than the default 1777. In particular, systems that do not wish to have lost files accessible by all users on the system should use a more restrictive set of permissions such as 700.

-n Assume a no response to all questions asked by **fsck_ext2fs** except for CONTINUE?, which is assumed to be affirmative; do not open the filesystem for writing.

-p Specify ''preen'' mode, described above.

-y Assume a yes response to all questions asked by **fsck_ext2fs**; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free map.
2. Blocks claimed by an inode outside the range of the filesystem.
3. Incorrect link counts.
4. Size checks:
   Directory size not a multiple of filesystem block size.
   Partially truncated file.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   File pointing to unallocated inode.
   Inode number out of range.
   Dot or dot-dot not the first two entries of a directory or having the wrong inode number.
8. Super Block checks:
   More blocks for inodes than there are in the filesystem.
   Bad free block map format.
   Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. If the `lost+found` directory does not exist, it is created. If there is insufficient space its size is increased.

Because of inconsistencies between the block device and the buffer cache, the raw device should always be used.

DIAGNOSTICS

The diagnostics produced by **fsck_ext2fs** are fully enumerated and explained in Appendix A of *Fsck − The UNIX File System Check Program*.

SEE ALSO

`fs`(5), `fstab`(5), `fsck`(8), `fsdb`(8), `newfs`(8), `reboot`(8)

NAME

fsck_ffs — Fast File System consistency check and interactive repair

SYNOPSIS

fsck_ffs [**−adFfPpq**] [**−B** *byteorder*] [**−b** *block*] [**−c** *level*] [**−m** *mode*] [**−y** | **−n**]
filesystem ...

DESCRIPTION

fsck_ffs performs interactive file system consistency checks and repair for each of the file systems specified on the command line. It is normally invoked from `fsck`(8).

The kernel takes care that only a restricted class of innocuous file system inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

Unreferenced inodes
Link counts in inodes too large
Missing blocks in the free map
Blocks in the free map also in files
Counts in the super-block wrong

These are the only inconsistencies that **fsck_ffs** in "preen" mode (with the **−p** option) will correct; if it encounters other inconsistencies, it exits with an abnormal return status. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature of the correction. After successfully correcting a file system, **fsck_ffs** will print the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

If sent a `QUIT` signal, **fsck_ffs** will finish the file system checks, then exit with an abnormal return status.

If **fsck_ffs** receives a `SIGINFO` signal (see the **status** argument for `stty`(1)), a line will be written to the standard error output indicating the name of the device currently being checked, the current phase number and phase-specific progress information.

Without the **−p** option, **fsck_ffs** audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that some of the corrective actions which are not correctable under the **−p** option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond `yes` or `no`. If the operator does not have write permission on the file system **fsck_ffs** will default to a **−n** action.

fsck_ffs has more consistency checks than its predecessors *check*, *dcheck*, *fcheck*, and *icheck* combined.

The following flags are interpreted by **fsck_ffs**.

−a	Interpret the filesystem as an Apple UFS filesystem, even if there is no Apple UFS volume label present.
−B *byteorder*	Convert the file system metadata to *byteorder* byte order if needed. Valid byte orders are "be" and "le". If **fsck_ffs** is interrupted while swapping the metadata byte order, the file system cannot be recovered. **fsck_ffs** will print a message in interactive mode if the file system is not in host byte order.
−b *block*	Use the block number *block* as the super block for the file system. Block 32 is usually an alternative super block.
−c *level*	Convert the file system to the level *level*. Note that the level of a file system can only be raised. There are currently five levels defined:

0 The file system is in the old (static table) format.

1 The file system is in the new (dynamic table) format.

2 The file system supports 32-bit UIDs and GIDs, short symbolic links are stored in the inode, and directories have an added field showing the file type.

3 If maxcontig is greater than one, build the free segment maps to aid in finding contiguous sets of blocks. If maxcontig is equal to one, delete any existing segment maps.

4 Rearrange the super block to the same layout as UFS2; disable the rotational layout tables and per cylinder group block totals.

In interactive mode, **fsck_ffs** will list the conversion to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the file system. In preen mode, the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the second line of output from `dumpfs`(8).

−d Print debugging output.

−F Indicates that *filesystem* is a file system image, rather than a raw character device. *filesystem* will be accessed 'as-is', and no attempts will be made to read a disklabel.

−f Force checking of file systems. Normally, if a file system is cleanly unmounted, the kernel will set a "clean flag" in the file system super block, and **fsck_ffs** will not check the file system. This option forces **fsck_ffs** to check the file system, regardless of the state of the clean flag.

−m *mode* Use the octal value *mode* as the permission bits to use when creating the `lost+found` directory rather than the default 1700. In particular, systems that do not wish to have lost files accessible by all users on the system should use a more restrictive set of permissions such as 700.

−n Assume a no response to all questions asked by **fsck_ffs** except for `CONTINUE?`, which is assumed to be affirmative; do not open the file system for writing.

−P Display a progress meter for the file system check. A new meter is displayed for each of the 5 file system check passes, unless **−p** is specified, in which case only one meter for overall progress is displayed. Progress meters are disabled if the **−d** option is specified.

−p Specify "preen" mode, described above.

−q Quiet mode, do not output any messages for clean filesystems.

−y Assume a yes response to all questions asked by **fsck_ffs**; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free map.
2. Blocks claimed by an inode outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   - Directory size not a multiple of DIRBLKSIZ.
   - Partially truncated file.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   - File pointing to unallocated inode.
   - Inode number out of range.
   - Dot or dot-dot not the first two entries of a directory or having the wrong inode number.
8. Super Block checks:
   - More blocks for inodes than there are in the file system.
   - Bad free block map format.
   - Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. If the `lost+found` directory does not exist, it is created. If there is insufficient space its size is increased.

Because of inconsistencies between the block device and the buffer cache, the raw device should always be used.

DIAGNOSTICS

The diagnostics produced by **fsck_ffs** are fully enumerated and explained in Appendix A of *Fsck − The UNIX File System Check Program*.

SEE ALSO

`fs`(5), `fstab`(5), `fsck`(8), `fsdb`(8), `newfs`(8), `reboot`(8)

NAME

fsck_lfs — Log-structured File System consistency check and interactive repair

SYNOPSIS

fsck_lfs [**-dfpq**] [**-b** *block*] [**-m** *mode*] [**-y** | **-n**] *filesystem ...*

DESCRIPTION

fsck_lfs performs interactive filesystem consistency checks and repair for each of the filesystems specified on the command line. It is normally invoked from `fsck`(8).

The design of LFS takes care that no filesystem inconsistencies can happen unless hardware or software failures intervene. **fsck_lfs** will report and optionally correct any such inconsistencies.

For each corrected inconsistency one or more lines will be printed identifying the filesystem on which the correction will take place, and the nature of the correction. After successfully correcting a filesystem, **fsck_lfs** will print the number of files on that filesystem, the number of used and free blocks, and the percentage of fragmentation.

If sent a `QUIT` signal, **fsck_lfs** will finish the filesystem checks, then exit with an abnormal return status.

Without the **-p** option, **fsck_lfs** audits and interactively repairs inconsistent conditions for filesystems. If the filesystem is inconsistent, the operator is prompted for concurrence before each correction is attempted. It should be noted that some of the corrective actions will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond `yes` or `no`. If the operator does not have write permission on the filesystem **fsck_lfs** will default to a **-n** action.

The following flags are interpreted by **fsck_lfs**:

-b *block* Use *block* as the super block for the filesystem.

-d Print debugging output.

-f Force checking of file systems. Normally, if a file system is cleanly unmounted, the kernel will set a "clean flag" in the file system superblock, and **fsck_lfs** will not check the file system. This option forces **fsck_lfs** to check the file system, regardless of the state of the clean flag.

-m *mode* Use *mode* specified in octal as the permission bits to use when creating the `lost+found` directory rather than the default 1700. In particular, systems that do not wish to have lost files accessible by all users on the system should use a more restrictive set of permissions such as 700.

-n Assume a no response to all questions asked by **fsck_lfs** except for `CONTINUE?`, which is assumed to be affirmative; do not open the filesystem for writing.

-p Specify "preen" mode. Currently, in this mode **fsck_lfs** rolls forward from the older checkpoint, and performs no other action.

-q Quiet mode, do not output any messages for clean filesystems.

-y Assume a yes response to all questions asked by **fsck_lfs**; this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode.
2. Blocks claimed by an inode outside the range of the filesystem.
3. Incorrect link counts.
4. Size checks:
   Directory size not a multiple of DIRBLKSIZ.
   Partially truncated file.
5. Bad inode format.
6. Directory checks:
   File pointing to unallocated inode.
   Inode number out of range.
   Dot or dot-dot not the first two entries of a directory or having the wrong inode number.
7. Super Block checks:
   More blocks for inodes than there are in the filesystem.
8. Index File checks:
   "In use" inodes on free list, or free inodes not on free list.
   Segment block counts incorrect, or "clean" segments containing live data.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. If the `lost+found` directory does not exist, it is created. If there is insufficient space its size is increased.

Because of inconsistencies between the block device and the buffer cache, the raw device should always be used.

DIAGNOSTICS

The diagnostics produced by **fsck_lfs** are fully enumerated and explained in Appendix A of *Fsck − The UNIX File System Check Program*.

SEE ALSO

`fstab`(5), `fsck`(8), `newfs_lfs`(8), `reboot`(8)

HISTORY

The **fsck_lfs** program was first made available in NetBSD 1.4.

AUTHORS

Most of the **fsck_lfs** program was taken from `fsck_ffs`(8); what was not was written by Konrad Schroder ⟨perseant@NetBSD.org⟩.

NAME

fsck_msdos — DOS/Windows (FAT) filesystem consistency checker

SYNOPSIS

fsck_msdos **−p** [**−f**] *filesystem ...*
fsck_msdos [**−fny**] *filesystem ...*

DESCRIPTION

The **fsck_msdos** utility verifies and repairs FAT filesystems (more commonly known as DOS filesystems).

The first form of **fsck_msdos** preens the specified filesystems. It is normally started by fsck(8) run from /etc/rc during automatic reboot, when a FAT filesystem is detected. When preening file systems, **fsck_msdos** will fix common inconsistencies non-interactively. If more serious problems are found, **fsck_msdos** does not try to fix them, indicates that it was not successful, and exits.

The second form of **fsck_msdos** checks the specified file systems and tries to repair all detected inconsistencies, requesting confirmation before making any changes.

The options are as follows:

−f This option is ignored by **fsck_msdos**, and is present only for compatibility with programs that check other file system types for consistency, such as fsck_ffs(8).

−n Causes **fsck_msdos** to assume no as the answer to all operator questions, except "CONTINUE?".

−p Preen the specified filesystems.

−y Causes **fsck_msdos** to assume yes as the answer to all operator questions.

SEE ALSO

fsck(8), fsck_ffs(8), mount_msdos(8)

BUGS

fsck_msdos is still under construction.

NAME

fsdb — FFS debugging/editing tool

SYNOPSIS

fsdb [**−dFn**] **−f** *fsname*

DESCRIPTION

fsdb opens *fsname* (usually a raw disk partition) and runs a command loop allowing manipulation of the file system's inode data. You are prompted to enter a command with "fsdb (inum X)>" where *X* is the currently selected i-number. The initial selected inode is the root of the filesystem (i-number 2). The command processor uses the editline(3) library, so you can use command line editing to reduce typing if desired. When you exit the command loop, the file system superblock is marked dirty and any buffered blocks are written to the file system.

The **−d** option enables additional debugging output (which comes primarily from fsck(8)-derived code).

The **−F** option indicates that *filesystem* is a file system image, rather than a raw character device. It will be accessed 'as-is', and no attempts will be made to read a disklabel.

The **−n** option disables writing to the device, preventing any changes from being made to the filesystem.

COMMANDS

Besides the built-in editline(3) commands, **fsdb** supports these commands:

help Print out the list of accepted commands.

inode *i-number*
Select inode *i-number* as the new current inode.

back Revert to the previously current inode.

clri Clear the current inode.

lookup *name*
cd *name*
Find *name* in the current directory and make its inode the current inode. *Name* may be a multi-component name or may begin with slash to indicate that the root inode should be used to start the lookup. If some component along the pathname is not found, the last valid directory encountered is left as the active inode.
This command is valid only if the starting inode is a directory.

active
print Print out the active inode.

uplink Increment the active inode's link count.

downlink
Decrement the active inode's link count.

linkcount *number*
Set the active inode's link count to *number*.

ls List the current inode's directory entries. This command is valid only if the current inode is a directory.

blks List the current inode's blocks numbers.

findblk *disk block number ...*
: Find the inode(s) owning the specified disk block(s) number(s). Note that these are not absolute disk blocks numbers, but offsets from the start of the partition.

rm *name*
del *name*
: Remove the entry *name* from the current directory inode. This command is valid only if the current inode is a directory.

ln *ino name*
: Create a link to inode *ino* under the name *name* in the current directory inode. This command is valid only if the current inode is a directory.

chinum *dirslot inum*
: Change the i-number in directory entry *dirslot* to *inum*.

chname *dirslot name*
: Change the name in directory entry *dirslot* to *name*. This command cannot expand a directory entry. You can only rename an entry if the name will fit into the existing directory slot.

chtype *type*
: Change the type of the current inode to *type*. *type* may be one of: *file*, *dir*, *socket*, or *fifo*.

chmod *mode*
: Change the mode bits of the current inode to *mode*. You cannot change the file type with this subcommand; use **chtype** to do that.

chflags *flags*
: Change the file flags of the current inode to *flags*.

chown *uid*
: Change the owner of the current inode to *uid*.

chgrp *gid*
: Change the group of the current inode to *gid*.

chgen *gen*
: Change the generation number of the current inode to *gen*.

mtime *time*
ctime *time*
atime *time*
: Change the modification, change, or access time (respectively) on the current inode to *time*. *Time* should be in the format *YYYYMMDDHHMMSS[.nsec]* where *nsec* is an optional nanosecond specification. If no nanoseconds are specified, the *mtimensec*, *ctimensec*, or *atimensec* field will be set to zero.

quit, q, exit, ⟨*EOF*⟩
: Exit the program.

SEE ALSO

editline(3), fs(5), clri(8), fsck(8)

HISTORY

fsdb uses the source code for fsck(8) to implement most of the file system manipulation code. The remainder of **fsdb** first appeared in NetBSD 1.1.

WARNING

Use this tool with extreme caution -- you can damage an FFS file system beyond what `fsck`(8) can repair.

BUGS

Manipulation of “short” symlinks doesn't work (in particular, don't try changing a symlink's type).
You must specify modes as numbers rather than symbolic names.
There are a bunch of other things that you might want to do which **fsdb** doesn't implement.

NAME

fsinfo − coordinate site-wide filesystem information

SYNOPSIS

fsinfo [**−v**] [**−a** *autodir*] [**−b** *bootparams*] [**−d** *dumpsets*] [**−e** *exports*] [**−f** *fstabs*] [**−h** *hostname*] [**−m** *automounts*] [**−I** *dir*] [**−D** *string[=string]*] [**−U** *string[=string]*] *config* ...

DESCRIPTION

The **fsinfo** utility takes a set of system configuration information, and generates a coordinated set of *amd* , *mount* and *mountd* configuration files.

The **fsinfo** command is fully described in the document *Amd - The 4.4BSD Automounter*

SEE ALSO

amd(8), **mount**(8), **mountd**(8).

‘‘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

HISTORY

The **fsinfo** command first appeared in 4.4BSD.

AUTHORS

Jan-Simon Pendry <jsp@doc.ic.ac.uk>, Department of Computing, Imperial College, London, UK.

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

fsirand — install random inode generation numbers in a filesystem

SYNOPSIS

fsirand [**−F**] [**−p**] [**−x** *constant*] *special*

DESCRIPTION

fsirand writes random inode generation numbers for all the inodes on device *special*. These random numbers make the NFS filehandles less predictable, increasing security of exported filesystems.

fsirand should be run on a clean and unmounted filesystem.

The options are as follows:

−F Indicates that *special* is a file system image, rather than a device name. *special* will be accessed ‘as-is’, without requiring that it is a raw character device and without attempting to read a disklabel.

−p Print the current inode generation numbers; the filesystem is not modified.

−x *constant*
Exclusive-or the given constant with the random number used in the generation process.

fsirand exits zero on success, non-zero on failure.

If **fsirand** receives a SIGINFO signal, statistics on the amount of work completed and estimated completion time (in minutes:seconds) will be written to the standard error output.

SEE ALSO

fsck_ffs(8), newfs(8)

NAME

fssconfig — configure file system snapshot devices

SYNOPSIS

fssconfig [**−cxv**] *device path backup* [*cluster* [*size*]]
fssconfig −u [**−v**] *device*
fssconfig −l [**−v**] [*device*]

DESCRIPTION

The **fssconfig** command configures file system snapshot pseudo disk devices. It will associate the file system snapshot disk *device* with a snapshot of *path* allowing the latter to be accessed as though it were a disk.

If *backup* resides on the snapshotted file system a persistent snapshot will be created. This snapshot is active until *backup* is unlinked. This snapshot mode is only supported for ffs files systems.

Otherwise data written through the *path* will be saved in *backup*. If *backup* is a regular file, it will be created with length *size*. Default size is the size of *path*. Data is saved to *backup* in units of *cluster* bytes.

Options indicate an action to be performed:

−c Configures the device. If successful, references to *device* will access the contents of *path* at the time the snapshot was taken. If *backup* is a directory, a temporary file will be created in this directory. This file will be unlinked on exit.

−l List the snapshot devices and indicate which ones are in use. If a specific *device* is given, then only that will be described.

−u Unconfigures the *device*.

−v Be more verbose listing the snapshot devices.

−x Unlink *backup* after the *device* is configured.

If no action option is given, **−c** is assumed.

FILES

`/dev/rfss?`
`/dev/fss?`

EXAMPLES

```
fssconfig fss0 /usr /tmp/back
```

Configures the snapshot device `fss0` for a snapshot of the `/usr` file system. Data written through `/usr` will be backed up in `/tmp/back`.

```
fssconfig fss1 / /dev/rsd0e 8192
```

Configures the snapshot device `fss1` for a snapshot of the `/` file system. Data written through `/` will be backed up in `/dev/rsd0e`. The backup will take place in units of 8192 bytes.

```
fssconfig -u fss0
```

Unconfigures the `fss0` device.

SEE ALSO

`opendisk`(3), `fss`(4), `mount`(8), `umount`(8)

HISTORY

The **`fssconfig`** command appeared in NetBSD 2.0.

BUGS

The `fss`(4) driver is *experimental*. Be sure you have a backup before you use it.

NAME

ftp-proxy — Internet File Transfer Protocol proxy server

SYNOPSIS

ftp-proxy -i [**-AnrVw**] [**-a** *address*] [**-D** *debuglevel*] [**-g** *group*] [**-M** *maxport*] [**-m** *minport*] [**-R** *address[:port]*] [**-S** *address*] [**-t** *timeout*] [**-u** *user*]
ftp-proxy -p [**-AnrVw**] [**-a** *address*] [**-D** *debuglevel*] [**-g** *group*] [**-M** *maxport*] [**-m** *minport*] [**-R** *address[:port]*] [**-S** *address*] [**-t** *timeout*] [**-u** *user*]

DESCRIPTION

ftp-proxy is a proxy for the Internet File Transfer Protocol. The proxy uses pf(4) or ipnat(4) and expects to have the FTP control connection as described in services(5) redirected to it via a pf.conf(5) or ipnat.conf(5) *rdr* command. An example of how to do that is further down in this document.

The options are as follows:

-A Permit only anonymous FTP connections. The proxy will allow connections to log in to other sites as the user "ftp" or "anonymous" only. Any attempt to log in as another user will be blocked by the proxy.

-a *address*
Specify the local IP address to use in bind(2) as the source for connections made by **ftp-proxy** when connecting to destination FTP servers. This may be necessary if the interface address of your default route is not reachable from the destinations **ftp-proxy** is attempting connections to, or this address is different from the one connections are being NATed to. In the usual case this means that *address* should be a publicly visible IP address assigned to one of the interfaces on the machine running **ftp-proxy** and should be the same address to which you are translating traffic if you are using the **-n** option.

-D *debuglevel*
Specify a debug level, where the proxy emits verbose debug output into syslogd(8) at level LOG_DEBUG. Meaningful values of debuglevel are 0-3, where 0 is no debug output and 3 is lots of debug output, the default being 0.

-g *group*
Specify the named group to drop group privileges to, after doing pf(4) lookups which require root. By default, **ftp-proxy** uses the default group of the user it drops privilege to.

-i Set **ftp-proxy** for use with the ipnat(4) part of IP-Filter.

-M *maxport*
Specify the upper end of the port range the proxy will use for the data connections it establishes. The default is IPPORT_HILASTAUTO defined in ⟨netinet/in.h⟩ as 65535.

-m *minport*
Specify the lower end of the port range the proxy will use for all data connections it establishes. The default is IPPORT_HIFIRSTAUTO defined in ⟨netinet/in.h⟩ as 49152.

-n Activate network address translation (NAT) mode. In this mode, the proxy will not attempt to proxy passive mode (PASV or EPSV) data connections. In order for this to work, the machine running the proxy will need to be forwarding packets and doing network address translation to allow the outbound passive connections from the client to reach the server. See pf.conf(5) for more details on NAT. The proxy only ignores passive mode data connections when using this flag; it will still proxy PORT and EPRT mode data connections. Without this flag, **ftp-proxy** does not require any IP forwarding or NAT beyond the *rdr* necessary to capture the FTP control connection.

-p Set **ftp-proxy** for use with pf.

-R *address:[port]*
Reverse proxy mode for FTP servers running behind a NAT gateway. In this mode, no redirection is needed. The proxy is run from inetd(8) on the port that external clients connect to (usually 21). Control connections and passive data connections are forwarded to the server.

-r Use reverse host (reverse DNS) lookups for logging and libwrap use. By default, the proxy does not look up hostnames for libwrap or logging purposes.

-S *address*
Source address to use for data connections made by the proxy. Useful when there are multiple addresses (aliases) available to the proxy. Clients may expect data connections to have the same source address as the control connections, and reject or drop other connections.

-t *timeout*
Specifies a timeout, in seconds. The proxy will exit and close open connections if it sees no data for the duration of the timeout. The default is 0, which means the proxy will not time out.

-u *user*
Specify the named user to drop privilege to, after doing pf(4) lookups which require root privilege. By default, **ftp-proxy** drops privilege to the user *proxy*.

Running as root means that the source of data connections the proxy makes for PORT and EPRT will be the RFC mandated port 20. When running as a non-root user, the source of the data connections from **ftp-proxy** will be chosen randomly from the range *minport* to *maxport* as described above.

-V Be verbose. With this option the proxy logs the control commands sent by clients and the replies sent by the servers to syslogd(8).

-w Use the tcp wrapper access control library hosts_access(3), allowing connections to be allowed or denied based on the tcp wrapper's hosts.allow(5) and hosts.deny(5) files. The proxy does libwrap operations after determining the destination of the captured control connection, so that tcp wrapper rules may be written based on the destination as well as the source of FTP connections.

ftp-proxy is run from inetd(8) and requires that FTP connections are redirected to it using a *rdr* rule. A typical way to do this would be to use either an ipnat.conf(5) rule such as

```
int_if = "xl0";
rdr $int_if 0/0 port 21 -> 127.0.0.1 port 8021 tcp
```

or a pf.conf(5) rule such as

```
int_if = "xl0"
rdr pass on $int_if proto tcp from any to any port 21 -> 127.0.0.1 port 8021
```

inetd(8) must then be configured to run **ftp-proxy** on the port from above using

```
127.0.0.1:8021 stream tcp nowait root /usr/libexec/ftp-proxy ftp-proxy -[ip]
```

in inetd.conf(5).

ftp-proxy accepts the redirected control connections and forwards them to the server. The proxy replaces the address and port number that the client sends through the control connection to the server with its own address and proxy port, where it listens for the data connection. When the server opens the data connection back to this port, the proxy forwards it to the client. If you're using IP-Filter, the ipf.conf(5) rules need to let pass connections to these proxy ports (see options **-u**, **-m**, and **-M** above) in on the external interface.

The following example allows only ports 49152 to 65535 to pass in statefully:

```
block in on $ext_if proto tcp all
pass  in on $ext_if inet proto tcp from any to $ext_if \
    port > 49151 keep state
```

If you're using pf, then the `pf.conf`(5) rules need to let pass connections to these proxy ports (see options **-u**, **-m**, and **-M** above) in on the external interface. The following example allows only ports 49152 to 65535 to pass in statefully:

```
block in on $ext_if proto tcp all
pass  in on $ext_if inet proto tcp from any to $ext_if \
    port > 49151 keep state
```

Alternatively, `pf.conf`(5) rules can make use of the fact that by default, **ftp-proxy** runs as user "proxy" to allow the backchannel connections, as in the following example:

```
block in on $ext_if proto tcp all
pass  in on $ext_if inet proto tcp from any to $ext_if \
    user proxy keep state
```

These examples do not cover the connections from the proxy to the foreign FTP server. If one does not pass outgoing connections by default additional rules are needed.

SEE ALSO

`ftp`(1), `pf`(4), `hosts.allow`(5), `hosts.deny`(5), `inetd.conf`(5), `ipf.conf`(5), `ipnat.conf`(5), `pf.conf`(5), `inetd`(8), `ipf`(8), `ipnat`(8), `pfctl`(8), `syslogd`(8)

BUGS

Extended Passive mode (EPSV) is not supported by the proxy and will not work unless the proxy is run in network address translation mode. When not in network address translation mode, the proxy returns an error to the client, hopefully forcing the client to revert to passive mode (PASV) which is supported. EPSV will work in network address translation mode, assuming a configuration setup which allows the EPSV connections through to their destinations.

IPv6 is not yet supported.

NAME

ftpd — Internet File Transfer Protocol server

SYNOPSIS

ftpd [**−a** *authmode*] [**−dilvU**] [**−g** *umask*] [**−p** *port*] [**−T** *maxtimeout*] [**−t** *timeout*] [**−-gss-bindings**] [**−I** | **−-no-insecure-oob**] [**−u** *default umask*] [**−B** | **−-builtin-ls**] [**−-good-chars=***string*]

DESCRIPTION

Ftpd is the Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the “ftp” service specification; see services(5).

Available options:

−a Select the level of authentication required. Kerberised login can not be turned off. The default is to only allow kerberised login. Other possibilities can be turned on by giving a string of comma separated flags as argument to **−a**. Recognised flags are:

plain Allow logging in with plaintext password. The password can be a(n) OTP or an ordinary password.

otp Same as *plain*, but only OTP is allowed.

ftp Allow anonymous login.

The following combination modes exists for backwards compatibility:

none Same as *plain,ftp*.

safe Same as *ftp*.

user Ignored.

−d Debugging information is written to the syslog using LOG_FTP.

−g Anonymous users will get a umask of *umask*.

−-gss-bindings
require the peer to use GSS-API bindings (ie make sure IP addresses match).

−i Open a socket and wait for a connection. This is mainly used for debugging when ftpd isn't started by inetd.

−l Each successful and failed ftp(1) session is logged using syslog with a facility of LOG_FTP. If this option is specified twice, the retrieve (get), store (put), append, delete, make directory, remove directory and rename operations and their filename arguments are also logged.

−p Use *port* (a service name or number) instead of the default *ftp/tcp*.

−T A client may also request a different timeout period; the maximum period allowed may be set to *timeout* seconds with the **−T** option. The default limit is 2 hours.

−t The inactivity timeout period is set to *timeout* seconds (the default is 15 minutes).

−u Set the initial umask to something else than the default 027.

−U In previous versions of **ftpd**, when a passive mode client requested a data connection to the server, the server would use data ports in the range 1024..4999. Now, by default, if the system supports the IP_PORTRANGE socket option, the server will use data ports in the range 49152..65535. Specifying this option will revert to the old behavior.

−v Verbose mode.

−B, **--builtin-ls**
use built-in ls to list files

--good-chars=*string*
allowed anonymous upload filename chars

−I --no-insecure-oob
don't allow insecure out of band. Heimdal ftp clients before 0.6.3 doesn't support secure oob, so turning on this option makes them no longer work.

The file `/etc/nologin` can be used to disable ftp access. If the file exists, **ftpd** displays it and exits. If the file `/etc/ftpwelcome` exists, **ftpd** prints it before issuing the "ready" message. If the file `/etc/motd` exists, **ftpd** prints it after a successful login.

The ftp server currently supports the following ftp requests. The case of the requests is ignored.

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory ("`ls -lgA`")
MKD	make a directory
MDTM	show last modification time of file
MODE	specify data transfer *mode*
NLST	give name list of files in directory
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
REST	restart incomplete transfer
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name
SITE	non-standard commands (see next section)
SIZE	return size of file
STAT	return status of server
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer *structure*
SYST	show operating system type of server system
TYPE	specify data transfer *type*
USER	specify user name

XCUP	change to parent of current working directory (deprecated)
XCWD	change working directory (deprecated)
XMKD	make a directory (deprecated)
XPWD	print the current working directory (deprecated)
XRMD	remove a directory (deprecated)

The following commands are specified by RFC2228.

AUTH	authentication/security mechanism
ADAT	authentication/security data
PROT	data channel protection level
PBSZ	protection buffer size
MIC	integrity protected command
CONF	confidentiality protected command
ENC	privacy protected command
CCC	clear command channel

The following non-standard or UNIX specific commands are supported by the SITE request.

UMASK	change umask, (e.g. **`SITE UMASK 002`**)
IDLE	set idle-timer, (e.g. **`SITE IDLE 60`**)
CHMOD	change mode of a file (e.g. **`SITE CHMOD 755 filename`**)
FIND	quickly find a specific file with GNU `locate`(1).
HELP	give help information.

The following Kerberos related site commands are understood.

KAUTH	obtain remote tickets.
KLIST	show remote tickets

The remaining ftp requests specified in Internet RFC 959 are recognized, but not implemented. MDTM and SIZE are not specified in RFC 959, but will appear in the next updated FTP RFC.

The ftp server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959. If a STAT command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

Ftpd interprets file names according to the "globbing" conventions used by `csh`(1). This allows users to use the metacharacters "`*?[]{}~`".

Ftpd authenticates users according to these rules.

1. If Kerberos authentication is used, the user must pass valid tickets and the principal must be allowed to login as the remote user.

2. The login name must be in the password data base, and not have a null password (if Kerberos is used the password field is not checked). In this case a password must be provided by the client before any file operations may be performed. If the user has an OTP key, the response from a successful USER command will include an OTP challenge. The client may choose to respond with a PASS command giving either a standard password or an OTP one-time password. The server will automatically determine which type of password it has been given and attempt to authenticate accordingly. See `otp`(1) for more information on OTP authentication.

3. The login name must not appear in the file `/etc/ftpusers`.

4. The user must have a standard shell returned by `getusershell`(3).

5. If the user name appears in the file `/etc/ftpchroot` the session's root will be changed to the user's login directory by `chroot`(2) as for an "anonymous" or "ftp" account (see next item). However, the user must still supply a password. This feature is intended as a compromise between a fully anonymous account and a fully privileged account. The account should also be set up as for an anonymous account.

6. If the user name is "anonymous" or "ftp", an anonymous ftp account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention an email address for the user should be used as the password).

In the last case, **ftpd** takes special measures to restrict the client's access privileges. The server performs a `chroot`(2) to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care, consider following these guidelines for anonymous ftp.

In general all files should be owned by "root", and have non-write permissions (644 or 755 depending on the kind of file). No files should be owned or writable by "ftp" (possibly with exception for the `~ftp/incoming`, as specified below).

`~ftp` The "ftp" homedirectory should be owned by root.

`~ftp/bin` The directory for external programs (such as `ls`(1)). These programs must either be statically linked, or you must setup an environment for dynamic linking when running chrooted. These programs will be used if present:

ls	Used when listing files.
compress	When retrieving a filename that ends in `.Z`, and that file isn't present, **ftpd** will try to find the filename without `.Z` and compress it on the fly.
gzip	Same as compress, just with files ending in `.gz`.
gtar	Enables retrieval of whole directories as files ending in `.tar`. Can also be combined with compression. You must use GNU Tar (or some other that supports the **−z** and **−Z** flags).
locate	Will enable "fast find" with the **SITE FIND** command. You must also create a `locatedb` file in `~ftp/etc`.

`~ftp/etc` If you put copies of the `passwd`(5) and `group`(5) files here, ls will be able to produce owner names rather than numbers. Remember to remove any passwords from these files.

The file `motd`, if present, will be printed after a successful login.

`~ftp/dev` Put a copy of `/dev/null`(7) here.

`~ftp/pub` Traditional place to put whatever you want to make public.

If you want guests to be able to upload files, create a `~ftp/incoming` directory owned by "root", and group "ftp" with mode 730 (make sure "ftp" is member of group "ftp"). The following restrictions apply to anonymous users:

- Directories created will have mode 700.
- Uploaded files will be created with an umask of 777, if not changed with the **−g** option.
- These command are not accessible: **DELE**, **RMD**, **RNTO**, **RNFR**, **SITE UMASK**, and **SITE CHMOD**.

- Filenames must start with an alpha-numeric character, and consist of alpha-numeric characters or any of the following: `+` (plus), `-` (minus), `=` (equal), `_` (underscore), `.` (period), and `,` (comma).

FILES

`/etc/ftpusers`	Access list for users.
`/etc/ftpchroot`	List of normal users who should be chroot'd.
`/etc/ftpwelcome`	Welcome notice.
`/etc/motd`	Welcome notice after login.
`/etc/nologin`	Displayed and access refused.
`~/.klogin`	Login access for Kerberos.

SEE ALSO

`ftp`(1), `otp`(1), `getusershell`(3), `ftpusers`(5), `syslogd`(8)

STANDARDS

RFC 959	FTP PROTOCOL SPECIFICATION
RFC 1938	OTP Specification
RFC 2228	FTP Security Extensions.

BUGS

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

HISTORY

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

NAME

ftpd — Internet File Transfer Protocol server

SYNOPSIS

ftpd [**-46DdHlnQqrsUuWwX**] [**-a** *anondir*] [**-C** *user*] [**-c** *confdir*] [**-e** *emailaddr*]
[**-h** *hostname*] [**-L** *xferlogfile*] [**-P** *dataport*] [**-V** *version*]

DESCRIPTION

ftpd is the Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the "ftp" service specification; see services(5).

Available options:

-4 When **-D** is specified, bind to IPv4 addresses only.

-6 When **-D** is specified, bind to IPv6 addresses only.

-a *anondir*
Define *anondir* as the directory to chroot(2) into for anonymous logins. Default is the home directory for the ftp user. This can also be specified with the ftpd.conf(5) **chroot** directive.

-C *user*
Check whether *user* would be granted access under the restrictions given in ftpusers(5) and exit without attempting a connection. **ftpd** exits with an exit code of 0 if access would be granted, or 1 otherwise. This can be useful for testing configurations.

-c *confdir*
Change the root directory of the configuration files from "/etc" to *confdir*. This changes the directory for the following files: /etc/ftpchroot, /etc/ftpusers, /etc/ftpwelcome, /etc/motd, and the file specified by the ftpd.conf(5) **limit** directive.

-D Run as daemon. **ftpd** will listen on the default FTP port for incoming connections and fork a child for each connection. This is lower overhead than starting **ftpd** from inetd(8) and thus might be useful on busy servers to reduce load.

-d Debugging information is written to the syslog using a facility of LOG_FTP.

-e *emailaddr*
Use *emailaddr* for the "%E" escape sequence (see **Display file escape sequences**)

-H Equivalent to "-h 'hostname'".

-h *hostname*
Explicitly set the hostname to advertise as to *hostname*. The default is the hostname associated with the IP address that **ftpd** is listening on. This ability (with or without **-h**), in conjunction with **-c** *confdir*, is useful when configuring 'virtual' FTP servers, each listening on separate addresses as separate names. Refer to inetd.conf(5) for more information on starting services to listen on specific IP addresses.

-L *xferlogfile*
Log wu-ftpd style 'xferlog' entries to *xferlogfile*.

-l Each successful and failed FTP session is logged using syslog with a facility of LOG_FTP. If this option is specified more than once, the retrieve (get), store (put), append, delete, make directory, remove directory and rename operations and their file name arguments are also logged.

-n Don't attempt translation of IP addresses to hostnames.

-P *dataport*
Use *dataport* as the data port, overriding the default of using the port one less that the port **ftpd** is listening on.

-Q Disable the use of pid files for keeping track of the number of logged-in users per class. This may reduce the load on heavily loaded FTP servers.

-q Enable the use of pid files for keeping track of the number of logged-in users per class. This is the default.

-r Permanently drop root privileges once the user is logged in. The use of this option may result in the server using a port other than the (listening-port - 1) for **PORT** style commands, which is contrary to the **RFC 959** specification, but in practice very few clients rely upon this behaviour. See **SECURITY CONSIDERATIONS** below for more details.

-s Require a secure authentication mechanism like Kerberos or S/Key to be used.

-U Don't log each concurrent FTP session to /var/run/utmp. This is the default.

-u Log each concurrent FTP session to /var/run/utmp, making them visible to commands such as who(1).

-V *version*
Use *version* as the version to advertise in the login banner and in the output of **STAT** and **SYST** instead of the default version information. If *version* is empty or '-' then don't display any version information.

-W Don't log each FTP session to /var/log/wtmp.

-w Log each FTP session to /var/log/wtmp, making them visible to commands such as last(1). This is the default.

-X Log wu-ftpd style 'xferlog' entries to the syslog, prefixed with "xferlog: ", using a facility of LOG_FTP. These syslog entries can be converted to a wu-ftpd style xferlog file suitable for input into a third-party log analysis tool with a command similar to:

```
      grep 'xferlog: ' /var/log/xferlog | \
         sed -e 's/^.*xferlog: //' > wuxferlog
```

The file /etc/nologin can be used to disable FTP access. If the file exists, **ftpd** displays it and exits. If the file /etc/ftpwelcome exists, **ftpd** prints it before issuing the "ready" message. If the file /etc/motd exists (under the chroot directory if applicable), **ftpd** prints it after a successful login. This may be changed with the ftpd.conf(5) directive **motd**.

The **ftpd** server currently supports the following FTP requests. The case of the requests is ignored.

Request	**Description**
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
EPSV	prepare for server-to-server transfer
EPRT	specify data connection port
FEAT	list extra features that are not defined in **RFC 959**

HELP	give help information
LIST	give list files in a directory (“`ls -lA`”)
LPSV	prepare for server-to-server transfer
LPRT	specify data connection port
MLSD	list contents of directory in a machine-processable form
MLST	show a pathname in a machine-processable form
MKD	make a directory
MDTM	show last modification time of file
MODE	specify data transfer *mode*
NLST	give name list of files in directory
NOOP	do nothing
OPTS	define persistent options for a given command
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
REST	restart incomplete transfer
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name
SITE	non-standard commands (see next section)
SIZE	return size of file
STAT	return status of server
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer *structure*
SYST	show operating system type of server system
TYPE	specify data transfer *type*
USER	specify user name
XCUP	change to parent of current working directory (deprecated)
XCWD	change working directory (deprecated)
XMKD	make a directory (deprecated)
XPWD	print the current working directory (deprecated)
XRMD	remove a directory (deprecated)

The following non-standard or UNIX specific commands are supported by the SITE request.

Request	**Description**
CHMOD	change mode of a file, e.g. ‘‘SITE CHMOD 755 filename’’
HELP	give help information.
IDLE	set idle-timer, e.g. ‘‘SITE IDLE 60’’
RATEGET	set maximum get rate throttle in bytes/second, e.g. ‘‘SITE RATEGET 5k’’
RATEPUT	set maximum put rate throttle in bytes/second, e.g. ‘‘SITE RATEPUT 5k’’
UMASK	change umask, e.g. ‘‘SITE UMASK 002’’

The following FTP requests (as specified in **RFC 959** and **RFC 2228**) are recognized, but are not implemented: **ACCT**, **ADAT**, **AUTH**, **CCC**, **CONF**, **ENC**, **MIC**, **PBSZ**, **PROT**, **REIN**, and **SMNT**.

The **ftpd** server will abort an active file transfer only when the **ABOR** command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet **RFC 959**. If a **STAT** command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

ftpd interprets file names according to the "globbing" conventions used by csh(1). This allows users to use the metacharacters "*?[]{}~".

User authentication

ftpd authenticates users according to five rules.

1. The login name must be in the password data base, /etc/pwd.db, and not have a null password. In this case a password must be provided by the client before any file operations may be performed. If the user has an S/Key key, the response from a successful **USER** command will include an S/Key challenge. The client may choose to respond with a **PASS** command giving either a standard password or an S/Key one-time password. The server will automatically determine which type of password it has been given and attempt to authenticate accordingly. See skey(1) for more information on S/Key authentication. S/Key is a Trademark of Bellcore.

2. The login name must be allowed based on the information in ftpusers(5).

3. The user must have a standard shell returned by getusershell(3). If the user's shell field in the password database is empty, the shell is assumed to be /bin/sh. As per shells(5), the user's shell must be listed with full path in /etc/shells.

4. If directed by the file ftpchroot(5) the session's root directory will be changed by chroot(2) to the directory specified in the ftpd.conf(5) **chroot** directive (if set), or to the home directory of the user. This facility may also be triggered by enabling the boolean **ftp-chroot** in login.conf(5). However, the user must still supply a password. This feature is intended as a compromise between a fully anonymous account and a fully privileged account. The account should also be set up as for an anonymous account.

5. If the user name is "anonymous" or "ftp", an anonymous FTP account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention an email address for the user should be used as the password).

   The server performs a chroot(2) to the directory specified in the ftpd.conf(5) **chroot** directive (if set), the **-a** *anondir* directory (if set), or to the home directory of the "ftp" user.

   The server then performs a chdir(2) to the directory specified in the ftpd.conf(5) **homedir** directive (if set), otherwise to /.

   If other restrictions are required (such as disabling of certain commands and the setting of a specific umask), then appropriate entries in ftpd.conf(5) are required.

   If the first character of the password supplied by an anonymous user is "-", then the verbose messages displayed at login and upon a **CWD** command are suppressed.

Display file escape sequences

When **ftpd** displays various files back to the client (such as /etc/ftpwelcome and /etc/motd), various escape strings are replaced with information pertinent to the current connection.

The supported escape strings are:

Escape	**Description**
%c	Class name.
%C	Current working directory.
%E	Email address given with **-e**.
%L	Local hostname.
%M	Maximum number of users for this class. Displays "unlimited" if there's no limit.

%N	Current number of users for this class.
%R	Remote hostname.
%s	If the result of the most recent "%M" or "%N" was not "`1`", print an "s".
%S	If the result of the most recent "%M" or "%N" was not "`1`", print an "S".
%T	Current time.
%U	User name.
%%	A "%" character.

Setting up a restricted ftp subtree

In order that system security is not breached, it is recommended that the subtrees for the "ftp" and "chroot" accounts be constructed with care, following these rules (replace "ftp" in the following directory names with the appropriate account name for 'chroot' users):

`~ftp` Make the home directory owned by "root" and unwritable by anyone.

`~ftp/bin` Make this directory owned by "root" and unwritable by anyone (mode 555). Generally any conversion commands should be installed here (mode 111).

`~ftp/etc` Make this directory owned by "root" and unwritable by anyone (mode 555). The files `pwd.db` (see `passwd`(5)) and `group` (see `group`(5)) must be present for the **LIST** command to be able to display owner and group names instead of numbers. The password field in `passwd`(5) is not used, and should not contain real passwords. The file `motd`, if present, will be printed after a successful login. These files should be mode 444.

`~ftp/pub` This directory and the subdirectories beneath it should be owned by the users and groups responsible for placing files in them, and be writable only by them (mode 755 or 775). They should *not* be owned or writable by ftp or its group.

`~ftp/incoming` This directory is where anonymous users place files they upload. The owners should be the user "ftp" and an appropriate group. Members of this group will be the only users with access to these files after they have been uploaded; these should be people who know how to deal with them appropriately. If you wish anonymous FTP users to be able to see the names of the files in this directory the permissions should be 770, otherwise they should be 370.

The following `ftpd.conf`(5) directives should be used:

```
modify guest off
umask  guest 0707
upload guest on
```

This will result in anonymous users being able to upload files to this directory, but they will not be able to download them, delete them, or overwrite them, due to the umask and disabling of the commands mentioned above.

`~ftp/tmp` This directory is used to create temporary files which contain the error messages generated by a conversion or **LIST** command. The owner should be the user "ftp". The permissions should be 300.

If you don't enable conversion commands, or don't want anonymous users uploading files here (see `~ftp/incoming` above), then don't create this directory. However, error messages from conversion or **LIST** commands won't be returned to the user. (This is the traditional behaviour.) Note that the `ftpd.conf`(5) directive **upload** can be used to prevent users uploading here.

To set up "ftp-only" accounts that provide only FTP, but no valid shell login, you can copy/link `/sbin/nologin` to `/sbin/ftplogin`, and enter `/sbin/ftplogin` to `/etc/shells` to allow logging-in via FTP into the accounts, which must have `/sbin/ftplogin` as login shell.

FILES

`/etc/ftpchroot`	List of normal users whose root directory should be changed via `chroot`(2).
`/etc/ftpd.conf`	Configure file conversions and other settings.
`/etc/ftpusers`	List of unwelcome/restricted users.
`/etc/ftpwelcome`	Welcome notice before login.
`/etc/motd`	Welcome notice after login.
`/etc/nologin`	If it exists, displayed and access is refused.
`/var/run/ftpd.pids-CLASS`	State file of logged-in processes for the **ftpd** class 'CLASS'.
`/var/run/utmp`	List of logged-in users on the system.
`/var/log/wtmp`	Login history database.

SEE ALSO

`ftp`(1), `skey`(1), `who`(1), `getusershell`(3), `ftpchroot`(5), `ftpd.conf`(5), `ftpusers`(5), `login.conf`(5), `syslogd`(8)

STANDARDS

ftpd recognizes all commands in **RFC 959**, follows the guidelines in **RFC 1123**, recognizes all commands in **RFC 2228** (although they are not supported yet), and supports the extensions from **RFC 2389**, **RFC 2428**, and **RFC 3659**.

HISTORY

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

Various features such as the `ftpd.conf`(5) functionality, **RFC 2389**, and **RFC 3659** support was implemented in NetBSD 1.3 and later releases by Luke Mewburn.

BUGS

The server must run as the super-user to create sockets with privileged port numbers (i.e, those less than `IPPORT_RESERVED`, which is 1024). If **ftpd** is listening on a privileged port it maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to privileged sockets. The **−r** option can be used to override this behaviour and force privileges to be permanently revoked; see **SECURITY CONSIDERATIONS** below for more details.

ftpd may have trouble handling connections from scoped IPv6 addresses, or IPv4 mapped addresses (IPv4 connection on `AF_INET6` socket). For the latter case, running two daemons, one for IPv4 and one for IPv6, will avoid the problem.

SECURITY CONSIDERATIONS

RFC 959 provides no restrictions on the **PORT** command, and this can lead to security problems, as **ftpd** can be fooled into connecting to any service on any host. With the "checkportcmd" feature of the `ftpd.conf`(5), **PORT** commands with different host addresses, or TCP ports lower than `IPPORT_RESERVED` will be rejected. This also prevents 'third-party proxy ftp' from working. Use of this option is *strongly* recommended, and enabled by default.

By default **ftpd** uses a port that is one less than the port it is listening on to communicate back to the client for the **EPRT**, **LPRT**, and **PORT** commands, unless overridden with **−P** *dataport*. As the default port for **ftpd** (21) is a privileged port below `IPPORT_RESERVED`, **ftpd** retains the ability to switch back to root privileges to bind these ports. In order to increase security by reducing the potential for a bug in **ftpd** providing a remote root compromise, **ftpd** will permanently drop root privileges if one of the following is true:

1. **ftpd** is running on a port greater than `IPPORT_RESERVED` and the user has logged in as a 'guest' or 'chroot' user.
2. **ftpd** was invoked with **-r**.

Don't create `~ftp/tmp` if you don't want anonymous users to upload files there. That directory is only necessary if you want to display the error messages of conversion commands to the user. Note that if uploads are disabled with the `ftpd.conf`(5) directive **upload**, then this directory cannot be abused by the user in this way, so it should be safe to create.

To avoid possible denial-of-service attacks, **SIZE** requests against files larger than 10240 bytes will be denied if the current transfer **TYPE** is '`A`' (ASCII).

NAME

fusermount — manage librefuse mount items

SYNOPSIS

fusermount [**−chpVx**] [**−d** *name*] *refuseoptions*
fusermount −u *mountpoint(s)*

DESCRIPTION

The **fusermount** utility acts as a frontend to the refuse(3) library, allowing mounting and unmounting of refuse-based file systems.

There are essentially two forms of the **fusermount** command. The first, and default option, is to mount a refuse-based file system. By using the **−u** argument, the file system can be unmounted.

The arguments to **fusermount** are as follows:

−c Set a flag to enable kernel caching of files. At present this option has no effect.

−d *name*
Make the name argument appear as the file system name in mount(8) and df(1) output.

−h Print a usage message and exit.

−p Check the file permissions. At present this option has no effect.

−V Display the **fusermount** version on stdout, and then exit successfully.

−x Allow mortal (non-root) users to access the file system. At present, this option has no effect.

The **fusermount** utility is included mainly for compatibility reasons, since some file systems demand its existence.

EXIT STATUS

fusermount returns 0 for successful operation, or non-zero if one of the operations did not complete successfully.

EXAMPLES

The command

```
fusermount -d ntfs-3g unused mount.ntfs-3g ntfs.img /mnt
```

will mount the file ntfs.img on the directory /mnt. Please note the unused argument in the command, which is necessary for compatibility with other implementations of the **fusermount** command.

SEE ALSO

df(1), puffs(3), refuse(3), mount(8)

HISTORY

The **fusermount** utility first appeared in NetBSD 5.0.

AUTHORS

The **fusermount** utility was written by Alistair Crooks ⟨agc@NetBSD.org⟩.

NAME

fwctl — IEEE1394 control utility

SYNOPSIS

fwctl [**-prt**] [**-b** *pri_req*] [**-c** *node*] [**-d** *node*] [**-g** *gap_count*] [**-l** *file*]
[**-M** *mode*] [**-m** *EUI64* | *hostname*] [**-o** *node*] [**-R** *filename*] [**-S** *filename*]
[**-s** *node*] [**-u** *bus_num*]

DESCRIPTION

The **fwctl** utility is designed to provide a way for users to access and control the NetBSD IEEE1394 subsystem. Without options, **fwctl** will output a list of devices that are/were connected to the bus.

The following options are available:

-b *pri_req* Set the PRIORITY_BUDGET register on all supported nodes.

-c *node* Show the configuration ROM on the node.

-d *node* Hex dump of the configuration ROM.

-g *gap_count* Broadcast *gap_count* by phy_config packet.

-l *file* Load hex dump file of the configuration ROM and parse it.

-M *modeExplicitly specify either*
dv or *mpeg* mode for the incoming stream. Only meaningful in case of and must precede the **-R** option. If not specified, the program will try to guess. If you get an error complaining about "format 0x20", try to force the "mpeg" mode.

-m *EUI64* | *hostname* Set default fwmem target. Hostname will be converted to EUI64.

-o *node* Send a link-on PHY packet to the node.

-p Dump PHY registers.

-R *filename* Receive DV or MPEG TS stream and dump it to a file. Use Ctrl-C to stop the receiving. Some DV cameras seem not to send the stream if a bus manager exits. If you cannot get the stream, try the following commands:

```
sysctl hw.ieee1394if.try_bmr=0
fwctl -r
```

The resulting file contains raw DV data excluding isochronous header and CIP header. It can be handled by the pkgsrc/multimedia/libdv package. Resulting MPEG TS stream can be played and sent over a network using the VideoLAN **vlc** tool in the FreeBSD Ports Collection. The stream can be piped directly to **vlc,** see EXAMPLES.

-r Initiate bus reset.

-S *filename* Send a DV file as isochronous stream.

-s *node* Write to the RESET_START register on the node.

-t Show the topology map.

-u *bus_num* Specify the IEEE1394 bus number to be operated on.

FILES

`/dev/fw0.0`

EXAMPLES

Each DV frame has a fixed size and it is easy to edit the frame order.

```
fwctl -R original.dv
```

Receive a DV stream with DV camera attached.

```
dd if=original.dv of=first.dv bs=120000 count=30
```

Get first 30 frames (NTSC).

```
dd if=original.dv of=second.dv bs=120000 skip=30 count=30
```

Get second 30 frames (NTSC).

```
cat second.dv first.dv | fwctl -S /dev/stdin
```

Swap first and second 30 frames and send them to DV recorder.

For PAL, replace "`bs=120000`" with "`bs=144000`".

```
fwcontrol -R file.m2t
```

Receive an MPEG TS stream from a camera producing MPEG transport stream. This has been tested with SONY HDR-FX1E camera that produces HD MPEG-2 stream at 25 Mbps bandwidth.

To send the stream from the camera over the network using TCP (which supprisingly works better with vlc), you can use

```
fwcontrol -R - | nc 192.168.10.11 9000
```

with **netcat** from ports and to receive the stream, use

```
nc -l -p 9000 | vlc -
```

To netcast via UDP, you need to use **buffer** program from ports, since vlc is not fast enough to read UDP packets from buffers and thus it experiences dropouts when run directly. The sending side can use

```
fwcontrol -R - | nc 192.168.10.11 9000
```

and to receive the stream, use

```
nc -l -u -p 9000 | buffer -s 10k -b 1000 -m 20m -p 5 | vlc -
```

For more information on how to work with **vlc** see its docs.

SEE ALSO

mplayer(1), vlc(1), fwip(4), fwohci(4), ieee1394if(4), sbp(4)

HISTORY

The **fwctl** command first appeared in FreeBSD 5.0, as **fwcontrol**. It was added to NetBSD 4.0 under its present name.

AUTHORS

Hidetoshi Shimokawa ⟨simokawa@FreeBSD.org⟩
Petr Holub ⟨hopet@ics.muni.cz⟩ - MPEG TS mode.
KIYOHARA Takashi ⟨kiyohara@NetBSD.org⟩

BUGS

This utility is still under development and provided for debugging purposes. Especially MPEG TS reception support is very rudimental and supports only high-bandwidth MPEG-2 streams (fn field in CIP header equals 3).

NAME

getNAME — get NAME sections from manual source for whatis/apropos data base

SYNOPSIS

getNAME [**−itvw**] *path* [*path ...*]

DESCRIPTION

The **getNAME** utility looks inside manual page sources to find the name of the page. It can be used to create a table of contents, report the style of manual, or to create an introductory manual page. By default, **getNAME** returns data for use in an apropos(1) database. **getNAME** is designed to be called from manual grovelling tools, not to be used directly.

Historically, makewhatis(8) used to use **getNAME** to get manpage names, but that's no longer the case.

OPTIONS

The following options are available:

−i Print information useful in creating an introduction manual. See intro(1), intro(2), etc. for examples.

−t Print information useful for creating a table of contents.

−v Complain about incorrectly formatted man pages.

−w Print information whether the manpage uses traditional man ("OLD") , new mandoc ("NEW") , or some unknown ("UNKNOWN") macros.

SEE ALSO

man(1), catman(8), makewhatis(8)

HISTORY

The **getNAME** command first appeared in 2.0BSD.

BUGS

It would be nice if **getNAME** could deal with compressed and/or preformatted manual pages. Looks for .S[Hh] NAME for consistency checking, but that breaks man pages in other languages.

NAME

getencstat — get SCSI Environmental Services Device enclosure status

SYNOPSIS

getencstat [**−v**] *device* [*device ...*]

DESCRIPTION

getencstat gets summary and detailed SCSI Environmental Services (or SAF-TE) device enclosure status. The overall status is printed out. If the overall status is considered okay, nothing else is printed out (unless the **−v** option is used).

A SCSI Environmental Services device enclosure may be either in the state of being **OK**, or in one or more of the states of **INFORMATIONAL**, **NON-CRITICAL**, **CRITICAL or UNRECOVERABLE** states. These overall states reflect a summary of the states of each object within such a device (such as power supplies or disk drives).

With the **−v** option, the status of all objects within the device is printed, whether **OK** or not. Along with the status of each object is the object identifier.

The user may then use `setencstat`(8) to try and clear overall device status, or may use `setobjstat`(8) to set specific object status.

FILES

`/dev/ses`*N*
SCSI Environmental Services Devices

SEE ALSO

`ses`(4), `sesd`(8), `setencstat`(8), `setobjstat`(8)

BUGS

NAME

getty, **uugetty** — set terminal modes for system access

SYNOPSIS

getty [*type* [*tty*]]
uugetty [*type* [*tty*]]

DESCRIPTION

The **getty** program is called by init(8) to open and initialize the tty line, read a login name, and invoke login(1). The devices on which to run **getty** are normally determined by ttys(5).

The **getty** program can also recognize a Point to Point Protocol (PPP) negotiation, and, if the **pp** attribute in gettytab(5) is set, invoke the program given by that string, e.g., pppd(8), instead of login(1). This makes it possible to use a single serial port for either a "shell" account with command line interface, or a PPP network link.

The argument *tty* is the special device file in /dev to open for the terminal (for example, "ttyh0"). If there is no argument or the argument is '**−**', the tty line is assumed to be open as file descriptor 0.

The *type* argument can be used to make **getty** treat the terminal line specially. This argument is used as an index into the gettytab(5) database, to determine the characteristics of the line. If there is no argument, or there is no such table, the *default* table is used. If there is no /etc/gettytab a set of system defaults is used. If indicated by the table located, **getty** will clear the terminal screen, print a banner heading, and prompt for a login name. Usually either the banner or the login prompt will include the system hostname.

getty uses the ttyaction(3) facility with an action of "getty" and user "root" to execute site-specific commands when it starts.

Most of the default actions of **getty** can be circumvented, or modified, by a suitable gettytab(5) table.

The **getty** program can be set to timeout after some interval, which will cause dial up lines to hang up if the login name is not entered reasonably quickly.

The **uugetty** program is the same, except that it uses pidlock(3) to respect the locks in /var/spool/lock of processes that dial out on that tty.

FILES

/etc/gettytab
/etc/ttys
/var/spool/lock/LCK..ttyXX

DIAGNOSTICS

ttyxx: No such device or address.
ttyxx: No such file or address. A terminal which is turned on in the ttys(5) file cannot be opened, likely because the requisite lines are either not configured into the system, the associated device was not attached during boot-time system configuration, or the special file in /dev does not exist.

SEE ALSO

login(1), ioctl(2), pidlock(3), ttyaction(3), tty(4), gettytab(5), ttys(5), init(8), pppd(8)

HISTORY

A **getty** program appeared in Version 6 AT&T UNIX.

NAME

gpioctl — control GPIO devices

SYNOPSIS

gpioctl [**−hq**] [**−d** *device*] [*pin*] [*0* | *1* | *2*]
gpioctl [**−hq**] [**−d** *device*] **−c** *pin* [*flags*]

DESCRIPTION

The **gpioctl** program allows manipulation of GPIO (General Purpose Input/Output) device pins. Such devices can be either part of the chipset or embedded CPU, or a separate chip. The usual way of using GPIO is to connect some simple devices such as LEDs, 1-wire thermal sensors, etc., to its pins.

Each GPIO device has an associated device file in the `/dev` directory. By default **gpioctl** uses `/dev/gpio0`, which corresponds to the first found GPIO device in the system. If more than one GPIO device is present, an alternative device file can be specified with the **−d** option in order to access a particular GPIO device.

When executed without any arguments, **gpioctl** reads information about the GPIO device and displays it.

GPIO pins can be either "read" or "written" with the values of logical 0 or 1. If only a *pin* number is specified on the command line, the pin state will be read from the GPIO controller and displayed. To write to a pin, a value must be specified after the *pin* number. Values can be either 0 or 1. A value of 2 has a special meaning: it "toggles" the pin, i.e. changes its state to the opposite.

Each pin can be configured with different flags with the **−c** option. The following configuration flags are supported by the GPIO framework:

in	input direction
out	output direction
inout	bi-directional
od	open-drain output
pp	push-pull output
tri	tri-state (output disabled)
pu	internal pull-up enabled
pd	internal pull-down enabled
iin	invert input
iout	invert output

Note that not all the flags can be supported by the particular GPIO controller. The list of supported flags is always displayed when executing **gpioctl** with the **−c** option. If only a *pin* number is specified on the command line, the current pin flags will be displayed. To change pin flags, a new flags set separated by spaces must be specified after the *pin* number.

The **−q** option causes **gpioctl** to operate quietly i.e. nothing is printed to stdout. The **−h** option displays a usage summary.

FILES

/dev/gpio*u*	GPIO device unit *u* file.

EXAMPLES

Configure pin 20 to have push-pull output:

```
# gpioctl -c 20 out pp
```

Write logical 1 to pin 20:

```
      # gpioctl 20 1
```

SEE ALSO

`elansc`(4), `gcscpcib`(4), `gpio`(4), `gscpcib`(4), `nsclpcsio`(4)

HISTORY

The **gpioctl** command first appeared in OpenBSD 3.6 and NetBSD 4.0.

AUTHORS

The **gpioctl** program was written by Alexander Yurchenko ⟨grange@openbsd.org⟩.

NAME

gpt — GUID partition table maintenance utility

SYNOPSIS

gpt [*general_options*] *command* [*command_options*] *device ...*

DESCRIPTION

The **gpt** utility provides the necessary functionality to manipulate GUID partition tables (GPTs), but see **BUGS** below for how and where functionality is missing. The basic usage model of the **gpt** tool follows that of the cvs(1) tool. The general options are described in the following paragraph. The remaining paragraphs describe the individual commands with their options. Here we conclude by mentioning that a *device* is either a special file corresponding to a disk-like device or a regular file. The command is applied to each *device* listed on the command line.

General Options

The general options allow the user to change default settings or otherwise change the behaviour that is applicable to all commands. Not all commands use all default settings, so some general options may not have an effect on all commands.

The **−p** *count* option allows the user to change the number of partitions the GPT can accommodate. This is used whenever a new GPT is created. By default, the **gpt** utility will create space for 128 partitions (or 32 sectors of 512 bytes).

The **−r** option causes the **gpt** utility to open the device for reading only. Currently this option is primarily useful for the **show** command, but the intent is to use it to implement dry-run behaviour.

The **−v** option controls the verbosity level. The level increases with every occurrence of this option. There is no formalized definition of the different levels yet.

Commands

gpt add [**−b** *number*] [**−i** *index*] [**−s** *count*] [**−t** *type*] *device ...*

The **add** command allows the user to add a new partition to an existing table. By default, it will create a UFS partition covering the first available block of an unused disk space. The command-specific options can be used to control this behaviour.

The **−b** *number* option allows the user to specify the starting (beginning) sector number of the partition. The minimum sector number is 1, but has to fall inside an unused region of disk space that is covered by the GPT.

The **−i** *index* option allows the user to specify which (free) entry in the GPT table is to be used for the new partition. By default, the first free entry is selected.

The **−s** *count* option allows the user to specify the size of the partition in sectors. The minimum size is 1.

The **−t** *type* option allows the user to specify the partition type. The type is given as an UUID, but **gpt** accepts **efi**, **swap**, **ufs**, **hfs**, **linux**, and **windows** as aliases for the most commonly used partition types.

gpt create [**−fp**] *device ...*

The **create** command allows the user to create a new (empty) GPT. By default, one cannot create a GPT when the device contains a MBR, however this can be overridden with the **−f** option. If the **−f** option is specified, an existing MBR is destroyed and any partitions described by the MBR are lost.

The **-p** option tells **gpt** to create only the primary table and not the backup table. This option is only useful for debugging and should not be used otherwise.

gpt destroy [**-r**] *device ...*

The **destroy** command allows the user to destroy an existing, possibly not empty GPT.

The **-r** option instructs **gpt** to destroy the table in a way that it can be recovered.

gpt label [**-a**] ⟨**-f** *file* | **-l** *label*⟩ *device ...*

gpt label [**-b** *number*] [**-i** *index*] [**-s** *count*] [**-t** *type*] ⟨**-f** *file* | **-l** *label*⟩ *device ...*

The **label** command allows the user to label any partitions that match the selection. At least one of the following selection options must be specified.

The **-a** option specifies that all partitions should be labeled. It is mutually exclusive with all other selection options.

The **-b** *number* option selects the partition that starts at the given block number.

The **-i** *index* option selects the partition with the given partition number.

The **-s** *count* option selects all partitions that have the given size. This can cause multiple partitions to be removed.

The **-t** *type* option selects all partitions that have the given type. The type is given as an UUID or by the aliases that the **add** command accepts. This can cause multiple partitions to be removed.

The **-f** *file* or **-l** *label* options specify the new label to be assigned to the selected partitions. The **-f** *file* option is used to read the label from the specified file. Only the first line is read from the file and the trailing newline character is stripped. If the file name is the dash or minus sign (**-**), the label is read from the standard input. The **-l** *label* option is used to specify the label in the command line. The label is assumed to be encoded in UTF-8.

gpt migrate [**-fs**] *device ...*

The **migrate** command allows the user to migrate an MBR-based disk partitioning into a GPT-based partitioning. By default, the MBR is not migrated when it contains partitions of an unknown type. This can be overridden with the **-f** option. Specifying the **-f** option will cause unknown partitions to be ignored and any data in it to be lost.

The **-s** option prevents migrating BSD disk labels into GPT partitions by creating the GPT equivalent of a slice.

gpt remove [**-a**] *device ...*

gpt remove [**-b** *number*] [**-i** *index*] [**-s** *count*] [**-t** *type*] *device ...*

The **remove** command allows the user to remove any and all partitions that match the selection. It uses the same selection options as the **label** command. See above for a description of these options. Partitions are removed by clearing the partition type. No other information is changed.

gpt show [**-lu**] *device ...*

The **show** command displays the current partitioning on the listed devices and gives an overall view of the disk contents. With the **-l** option the GPT partition label will be displayed instead of the GPT partition type. The option has no effect on non-GPT partitions. With the **-u** option the GPT partition type is displayed as an UUID instead of in a user friendly form. The **-l** option takes precedence over the **-u** option.

SEE ALSO

`fdisk`(8), `mount`(8), `newfs`(8), `swapon`(8)

HISTORY

The **gpt** utility appeared in FreeBSD 5.0 for ia64.

BUGS

The development of the **gpt** utility is still work in progress. Many necessary features are missing or partially implemented. In practice this means that the manual page, supposed to describe these features, is farther removed from being complete or useful. As such, missing functionality is not even documented as missing. However, it is believed that the currently present functionality is reliable and stable enough that this tool can be used without bullet-proof footware if one thinks one does not make mistakes.

It is expected that the basic usage model does not change, but it is possible that future versions will not be compatible in the strictest sense of the word. For example, the **−p** *count* option may be changed to a command option rather than a generic option. There are only two commands that use it so there is a chance that the natural tendency for people is to use it as a command option. Also, options primarily intended for diagnostic or debug purposes may be removed in future versions.

Another possibility is that the current usage model is accompanied by other interfaces to make the tool usable as a back-end. This all depends on demand and thus feedback.

NAME

grfconfig — alter grf device screen mode definitions at run time

SYNOPSIS

grfconfig [**-r**] *device* [*file*]

DESCRIPTION

grfconfig is used to change or view the screen mode definition list contained in a grf device. You may alter the console screen definition as well as the definitions for the graphic screen. The console will automatically reinitialize itself to the new screen mode.

The following flags and arguments are interpreted by **grfconfig**:

-r	Print out a raw listing of the mode definitions instead of the pretty list normally shown.
device	The grf device to manipulate. This argument is required.
file	The file which contains the mode definitions. If this argument is not specified, **grfconfig** will print out of a list of the modes currently loaded into the grf device.

MODE DEFINITION FILE

The mode definitions are taken from a file which has lines of the format:

num clk wid hi dep hbs hss hse ht vbs vss vse vt flags

num The mode number or 'c' for the console mode.

clk The pixel clock in Hz.

wid The screen mode's width.

hi The screen mode's height.

dep The bitdepth of the mode. Use 4 for a text console mode.

hbs hss hse ht
The horizontal timing parameters for the mode in pixel values. All the values are relative to the end of the horizontal blank (beginning of the displayed area).

vbs vss vse vt
The vertical timing parameters for the mode in line values. All the values are relative to the end of vertical blank (beginning of the displayed area).

flags By default every mode uses negative horizontal and vertical sync pulses, it is non-interlaced and does not use scandoubling.

default	Use the default flags: -hsync -vsync
doublescan	Doublescan mode
interlace	Interlace mode
+hsync	Positive horizontal sync pulses
-hsync	Negative horizontal sync pulses
+vsync	Positive vertical sync pulses
-vsync	Negative vertical sync pulses

sync-on-green Composite sync on green

```
------------------------------------------------------
|                           ^                         |
|                          vse                        |
|      (0,0)                                          |
|        *-----------------------------------         |
|        |                  ^        ^      |         |
|        |                 vbe       !      |         |
|        |                           !      |         |
|        |                           !      |         |
|<-hse   |<-hbe                      !      |<-hbs    |      |      |
|        |                           !      |   hss->| hse->| hbe->|
|        |                          hi      |        |      |      |
|        |                           !      |         |
|        |                           !      |         |
|        |≤=========== wid =========+=====>|        |
|        |                           !      |         |
|        |                           !      |         |
|        |                           !      |         |
|        |                           v      |         |
|        ------------------------------------         |
|                           ^                         |
|                          vbs                        |
|                                                     |
------------------------------------------------------
                            ^
                           vss
                       - ------- -
                            ^
                           vse
                       - ------- -
                            ^
                           vbe
```

SEE ALSO

console(4), grfcl(4), grfcv(4), grfcv3d(4), grfet(4), grfrh(4), grfrt(4), grful(4), iteconfig(8)

HISTORY

The **grfconfig** command first appeared in NetBSD 1.0.

The mode definition file changed two times.

In NetBSD 1.0 all horizontal values were videoclock cycle values instead of pixel values:

num	clk	wid	hi	dep	hbs	hss	hse	hbe	ht	vbs	vss	vse	vbe	vt
1	31000000	640	480	8	80	86	96	102	104	480	489	492	517	520
2	31000000	640	480	8	80	86	96	102	104	240	244	246	258	260
3	31000000	640	480	8	80	86	96	102	104	960	978	984	1034	1040

In NetBSD 1.1 and NetBSD 1.2:

```
num clk      wid hi  dep hbs hss hse hbe ht  vbs vss vse vbe  vt
 1   31000000 640 480  8  640 688 768 816 832 480 489 492  517  520
 2   31000000 640 480  8  640 688 768 816 832 240 244 246  258  260
 3   31000000 640 480  8  640 688 768 816 832 960 978 984 1034 1040
```

the vertical values were used to select the interlace or doublescan mode. All vertical values were half the width for the interlace mode and twice the width for the doublescan mode.

Beginning with NetBSD 1.3:

```
num clk      wid hi  dep hbs hss hse ht  vbs vss vse vt  flags
 1   31000000 640 480  8  640 688 768 832 480 489 492 520 default
 2   31000000 640 480  8  640 688 768 832 480 489 492 520 interlace
 3   31000000 640 480  8  640 688 768 832 480 489 492 520 doublescan
 4   31000000 640 480  8  640 688 768 832 480 489 492 520 +hsync +vsync
```

hbe and vbe are computed in the grf drivers.

BUGS

grfconfig can not set the modes for `/dev/grf1`, `/dev/grf2` and `/dev/grf4` and it will not work for `/dev/grf0`.

NAME

group — manage group information on the system

SYNOPSIS

group add [options] *group*
group del [options] *group*
group info [options] *group*
group mod [options] *group*

DESCRIPTION

The **group** utility acts as a frontend to the groupadd(8), groupmod(8), groupinfo(8), and groupdel(8) commands. The utilities by default are built with EXTENSIONS. This allows for further functionality.

For a full explanation of the options available, please see the relevant manual page.

EXIT STATUS

The **group** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

group(5), groupadd(8), groupdel(8), groupinfo(8), groupmod(8)

HISTORY

The **group** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **group** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

groupadd — add a group to the system

SYNOPSIS

groupadd [**-ov**] [**-g** *gid*] [**-r** *lowgid..highgid*] *group*

DESCRIPTION

The **groupadd** utility adds a group to the system. See group(8) for more information about EXTENSIONS. The options are as follows:

-g *gid*
Give the numeric group identifier to be used for the new group.

-o Allow the new group to have a gid which is already in use for another group.

-r *lowgid..highgid*
Set the low and high bounds of a gid range for new groups. A new group can only be created if there are gids which can be assigned inside the range. This option is included if built with EXTENSIONS.

-v Enable verbose mode - explain the commands as they are executed. This option is included if built with EXTENSIONS.

EXIT STATUS

The **groupadd** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

group(5), group(8), user(8)

HISTORY

The **groupadd** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **groupadd** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

groupdel — remove a group from the system

SYNOPSIS

groupdel [**-v**] *group*

DESCRIPTION

The **groupdel** utility removes a group from the system. See group(8) for more information about EXTENSIONS. The options are as follows:

-v Enable verbose mode - explain the commands as they are executed. This option is included if built with EXTENSIONS.

EXIT STATUS

The **groupdel** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

group(5), group(8), user(8)

HISTORY

The **groupdel** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **groupdel** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

groupinfo — displays group information

SYNOPSIS

groupinfo [**-ev**] *group*

DESCRIPTION

The **groupinfo** utility retrieves the group information from the system. The **groupinfo** utility is only available if built with `EXTENSIONS`. See `group`(8) for more information.

The following command line options are recognised:

-e Return 0 if the group exists, and non-zero if the group does not exist, on the system. No information is displayed. This form of the command is useful for scripts which need to check whether a particular group name or gid is already in use on the system.

-v Perform any actions in a verbose manner.

The *group* argument may either be a group's name, or a gid.

EXIT STATUS

The **groupinfo** utility exits 0 on success, and >0 if an error occurs.

FILES

`/etc/usermgmt.conf`

SEE ALSO

`passwd`(5), `group`(8)

HISTORY

The **groupinfo** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **groupinfo** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

groupmod — modify an existing group on the system

SYNOPSIS

groupmod [**-ov**] [**-g** *gid*] [**-n** *newname*] *group*

DESCRIPTION

The **groupmod** utility modifies an existing group on the system. See group(8) for more information about EXTENSIONS. The options are as follows:

-g *gid*
: Give the numeric group identifier to be used for the new group.

-n *new-group-name*
: Give the new name which the group shall have.

-o Allow the new group to have a gid which is already in use for another group.

-v Enable verbose mode - explain the commands as they are executed. This option is included if built with EXTENSIONS.

EXIT STATUS

The **groupmod** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

group(5), group(8), user(8)

HISTORY

The **groupmod** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **groupmod** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

gspa — assembler for the GSP chip

SYNOPSIS

gspa [**-c** *c_array_name*] [**-l** *list_file*] [**-o** *hex_file*] [infile]

DESCRIPTION

gspa is an assembler for the TMS34010 and TMS34020 graphics processor chips.

The supported options are:

-c *c_array_name*
: Create a pre-initialized C-structure *c_array_name* with the hex code of the assembler output.

-l *list_file*
: Create commented (with the input file) hex code of the assembler output in *list_file*.

-o *hex_file*
: Create the output in *hex_file*. If **-c** is used, C code will be written to *hex_file*, otherwise uncommented hex code of the assembler output will be written.

If no **-o** option is given, output will be written to stdout.

If no *infile* is given, input will be read from stdin.

HISTORY

gspa appeared in NetBSD 1.1.

AUTHORS

gspa was written by Paul Mackerras. The **-c** mode was added by
Ignatios Souvatzis ⟨is@NetBSD.org⟩. This man page was written by
Thomas Klausner ⟨wiz@NetBSD.org⟩.

NAME

hlfsd − home-link file system daemon

SYNOPSIS

hlfsd [**−fhnpvC**] [**−a** *alt_dir*] [**−c** *cache-interval*] [**−g** *group*] [**−i** *reload-interval*] [**−l** *logfile*] [**−o** *mount-options*] [**−x** *log-options*] [**−D** *debug-options*] [**−P** *password-file*] [*linkname* [*subdir*]]

DESCRIPTION

Hlfsd is a daemon which implements a filesystem containing a symbolic link to subdirectory within a user's home directory, depending on the user which accessed that link. It was primarily designed to redirect incoming mail to users' home directories, so that it can read from anywhere.

Hlfsd operates by mounting itself as an NFS server for the directory containing *linkname*, which defaults to **/hlfs/home**. Lookups within that directory are handled by **hlfsd**, which uses the password map to determine how to resolve the lookup. The directory will be created if it doesn't already exist. The symbolic link will be to the accessing user's home directory, with *subdir* appended to it. If not specified, *subdir* defaults to **.hlfsdir**. This directory will also be created if it does not already exist.

A SIGTERM sent to **hlfsd** will cause it to shutdown. A SIGHUP will flush the internal caches, and reload the password map. It will also close and reopen the log file, to enable the original log file to be removed or rotated. A SIGUSR1 will cause it to dump its internal table of user IDs and home directories to the file **/usr/tmp/hlfsd.dump.XXXXXX**.

OPTIONS

−a *alt_dir*
: Alternate directory. The name of the directory to which the symbolic link returned by **hlfsd** will point, if it cannot access the home directory of the user. This defaults to **/var/hlfs**. This directory will be created if it doesn't exist. It is expected that either users will read these files, or the system administrators will run a script to resend this "lost mail" to its owner.

−c *cache-interval*
: Caching interval. **Hlfsd** will cache the validity of home directories for this interval, in seconds. Entries which have been verified within the last *cache-interval* seconds will not be verified again, since the operation could be expensive, and the entries are most likely still valid. After the interval has expired, **hlfsd** will re-verify the validity of the user's home directory, and reset the cache time-counter. The default value for *cache-interval* is 300 seconds (5 minutes).

−f
: Force fast startup. This option tells **hlfsd** to skip startup-time consistency checks such as existence of mount directory, alternate spool directory, symlink to be hidden under the mount directory, their permissions and validity.

−g *group*
: Set the special group HLFS_GID to *group*. Programs such as **from** or **comsat**, which access the mailboxes of other users) must be setgid HLFS_GID to work properly. The default group is "hlfs". If no group is provided, and there is no group "hlfs", this feature is disabled.

−h
: Help. Print a brief help message, and exit.

−i *reload-interval*
: Map-reloading interval. Each *reload-interval* seconds, **hlfsd** will reload the password map. **Hlfsd** needs the password map for the UIDs and home directory pathnames. **Hlfsd** schedules a SIGALRM to reload the password maps. A SIGHUP sent to **hlfsd** will force it to reload the maps immediately. The default value for *reload-interval* is 900 seconds (15 minutes.)

−l *logfile*
: Specify a log file to which **hlfsd** will record events. If *logfile* is the string **syslog** then the log messages will be sent to the system log daemon by *syslog*(3), using the LOG_DAEMON facility. This is also the default.

−n
: No verify. **Hlfsd** will not verify the validity of the symbolic link it will be returning, or that the user's home directory contains sufficient disk-space for spooling. This can speed up **hlfsd** at the cost of possibly returning symbolic links to home directories which are not currently accessible or

are full. By default, **hlfsd** validates the symbolic-link in the background. The **−n** option overrides the meaning of the **−c** option, since no caching is necessary.

−o *mount-options*
Mount options. Mount options which **hlfsd** will use to mount itself on top of *dirname.* By default, *mount-options* is set to "ro". If the system supports symbolic-link caching, default options are set to "ro,nocache".

−p Print PID. Outputs the process-id of **hlfsd** to standard output where it can be saved into a file.

−v Version. Displays version information to standard error.

−x *log-options*
Specify run-time logging options. The options are a comma separated list chosen from: fatal, error, user, warn, info, map, stats, all.

−C Force **hlfsd** to run on systems that cannot turn off the NFS attribute-cache. Use of this option on those systems is discouraged, as it may result in loss or mis-delivery of mail. The option is ignored on systems that can turn off the attribute-cache.

−D *log-options*
Select from a variety of debugging options. Prefixing an option with the string **no** reverses the effect of that option. Options are cumulative. The most useful option is **all**. Since this option is only used for debugging other options are not documented here. A fuller description is available in the program source. A SIGUSR1 sent to **hlfsd** will cause it to dump its internal password map to the file **/usr/tmp/hlfsd.dump.XXXXXX**.

−P *password-file*
Read the user-name, user-id, and home directory information from the file *password-file.* Normally, **hlfsd** will use *getpwent*(3) to read the password database. This option allows you to override the default database, and is useful if you want to map users' mail files to a directory other than their home directory. Only the username, uid, and home-directory fields of the file *password-file* are read and checked. All other fields are ignored. The file *password-file* must otherwise be compliant with Unix System 7 colon-delimited format *passwd*(5).

FILES

/hlfs directory under which **hlfsd** mounts itself and manages the symbolic link **home**.

.hlfsdir
default sub-directory in the user's home directory, to which the **home** symbolic link returned by **hlfsd** points.

/var/hlfs
directory to which **home** symbolic link returned by **hlfsd** points if it is unable to verify the that user's home directory is accessible.

SEE ALSO

mail(1), getgrent(3), **getpwent**(3), **passwd**(5), **amd**(8), **cron(8), mount**(8), **sendmail**(8), **umount**(8)

HLFSD: Delivering Email to Your $HOME, in *Proc. LISA-VII, The 7th Usenix System Administration Conference*, November 1993.

‘‘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

AUTHORS

Erez Zadok <ezk@cs.sunysb.edu>, Computer Science Department, Stony Brook University, Stony Brook, New York, USA. and Alexander Dupuy <dupuy@smarts.com>, System Management ARTS, White Plains, New York, USA.

Other authors and contributors to am-utils are listed in the **AUTHORS** file distributed with am-utils.

NAME

hostapd − IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator

SYNOPSIS

hostapd [-hdBKtv] [-P <PID file>] <configuration file(s)>

DESCRIPTION

This manual page documents briefly 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. The current version supports Linux (Host AP, madwifi, Prism54 drivers) and FreeBSD (net80211).

hostapd is designed to be a "daemon" program that runs in the background and acts as the backend component controlling authentication. **hostapd** supports separate frontend programs and an example text-based frontend, **hostapd_cli**, is included with **hostapd**.

OPTIONS

A summary of options is included below. For a complete description, run **hostapd** from the command line.

−h Show usage.

−d Show more debug messages.

−dd Show even more debug messages.

−B Run daemon in the background.

−P <PID file>
Path to PID file.

−K Include key data in debug messages.

−t Include timestamps in some debug messages.

−v Show hostapd version.

SEE ALSO

hostapd_cli(1).

AUTHOR

hostapd 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

hostapd — authenticator for IEEE 802.11 networks

SYNOPSIS

hostapd [**−BdhKtv**] *config-file ...*

DESCRIPTION

The **hostapd** utility is an authenticator for IEEE 802.11 networks. It provides full support for WPA/IEEE 802.11i and can also act as an IEEE 802.1X Authenticator with a suitable backend Authentication Server (typically FreeRADIUS). The **hostapd** utility implements the authentication protocols that piggyback on top of the normal IEEE 802.11 protocol mechanisms. To use **hostapd** as an authenticator, the underlying device must support some basic functionality such as the ability to set security information in the 802.11 management frames. Beware that not all devices have this support.

The **hostapd** utility is designed to be a "daemon" program that runs in the background and acts as the backend component controlling the wireless connection. It supports separate frontend programs such as the text-based frontend, `hostapd_cli`(8).

The following arguments must be specified on the command line:

config-file
: Use the settings in the specified configuration file; the name of the specified wireless interface is contained in this file. See `hostapd.conf`(5) for a description of the configuration file syntax.

  Changes to the configuration file can be reloaded by sending a `SIGHUP` to the **hostapd** processor or with the `hostapd_cli`(8) utility, using "`hostapd_cli reconfigure`".

OPTIONS

The options are as follows:

−d Enable debugging messages. If this option is supplied twice, more verbose messages are displayed.

−h Show help text.

−t Include timestamps in debugging output.

−v Display version information on the terminal and exit.

−B Detach from the controlling terminal and run as a daemon process in the background.

−K Include key information in debugging output.

SEE ALSO

`ath`(4), `ipw`(4), `iwi`(4), `ral`(4), `wi`(4), `hostapd.conf`(5), `hostapd_cli`(8), `ifconfig`(8)

HISTORY

The **hostapd** utility first appeared in NetBSD 4.0.

AUTHORS

The **hostapd** utility was written by Jouni Malinen ⟨jkmaline@cc.hut.fi⟩. This manual page is derived from the `README` file included in the **hostapd** distribution.

NAME

hostapd_cli — text-based frontend program for interacting with hostapd(8)

SYNOPSIS

hostapd_cli [*commands*]

DESCRIPTION

The **hostapd_cli** utility is a text-based frontend program for interacting with hostapd(8). It is used to query the current status.

The **hostapd_cli** utility can show the current authentication status, dot11 and dot1x MIBs, etc.

The **hostapd_cli** utility supports two modes: interactive and command line. Both modes share the same command set.

Interactive mode is started when **hostapd_cli** is executed without any parameters on the command line. Commands are then entered from the controlling terminal in response to the **hostapd_cli** prompt. In command line mode, the same commands are entered as command line arguments.

COMMANDS

The following commands may be supplied on the command line or at a prompt when operating interactively.

mib Report MIB variables (dot1x, dot11) for the current interface.

sta *addr*
Report the MIB variables for the associated station with MAC address *addr*.

all_sta
Report the MIB variables for all associated stations.

help Show usage help.

interface [*ifname*]
Show available interfaces and/or set the current interface when multiple are available.

level *debug_level*
Change the debugging level in hostapd(8). Larger numbers generate more messages.

license
Display the full license for **hostapd_cli**.

quit Exit **hostapd_cli**.

SEE ALSO

hostapd.conf(5), hostapd(8)

HISTORY

The **hostapd_cli** utility first appeared in NetBSD 4.0.

AUTHORS

The **hostapd_cli** utility was written by Jouni Malinen ⟨jkmaline@cc.hut.fi⟩. This manual page is derived from the README file included in the **hostapd** distribution.

NAME

hpcboot — load and boot kernel from Windows CE

SYNOPSIS

hpcboot.exe

DESCRIPTION

hpcboot is a program that runs on Windows CE. It loads and executes the specified NetBSD kernel. **hpcboot** supports hpcarm, hpcmips, and hpcsh ports.

Click on the "Boot" button to start the boot process with selected options. Click on the "Cancel" button to exit **hpcboot**.

"**Kernel**" Tab

On this tab you can select the kernel to boot and options to pass to the kernel.

Directory
: In this combobox you specify the "current" directory. The kernel and miniroot image pathnames are taken to be relative to this directory.

  hpcboot can load kernel and miniroot from FAT and UFS filesystems, and via HTTP.

Kernel
: In this text field you specify the name of the kernel to load. Kernels compressed with gzip(1) are supported.

Model
: Select your H/PC model in this combobox.

Root File System
: This group of controls lets you specify the desired root file system type. You can select wd(4), sd(4), md(4), and NFS root.

  If you select md(4) memory disk root file system, you should specify the path name of the file system image in the text field below. Miniroot images compressed with gzip(1) are supported.

Kernel Boot Flags
: This group of controls is used to pass boot flags to the kernel.

"**Option**" Tab

On this tab you can specify miscellaneous options that mostly control the **hpcboot** program itself.

Auto Boot
: If this option is selected **hpcboot** will automatically boot NetBSD after the specified timeout.

Reverse Video
: Tells kernel if it should use the framebuffer in reverse video mode.

Pause Before Boot
: If selected, a warning dialog will be presented *before* anything is done, right after the "Boot" button is pressed.

Load Debug Info
: This option currently does nothing.

Safety Message
: If selected, a warning dialog will be presented *after* the kernel has been loaded and prepared to be started. This will be your last chance to cancel the boot.

Extra Kernel Options
In this text field you can specify additional options to pass to the kernel.

"**Console**" **Tab**

This tab gets its name from the big text area that **hpcboot** uses as the "console" to report its progress.

Save To File
If checked, the progress log will be sent to the specified file instead.

"Checkboxes Anonymous"
The row of 8 checkboxes controls debugging options for **hpcboot** itself. They control the bits of an internal variable, the leftmost checkbox being the 7th bit.

"Buttons Anonymous"
The buttons "a" to "d" control 4 "hooks" a developer might want to use during **hpcboot** development.

SEE ALSO

kloader(4), boot(8)

HISTORY

The **hpcboot** utility first appeared in NetBSD 1.6.

BUGS

hpcboot reads the entire kernel image at once, and requires enough free area on the main memory.

NAME

hprop — propagate the KDC database

SYNOPSIS

hprop [**−m** *file* | **−-master-key=**file] [**−d** *file* | **−-database=**file] [**−-source=***heimdal*|*mit-dump*|*krb4-dump*|*kaserver*] [**−r** *string* | **−-v4-realm=***string*] [**−c** *cell* | **−-cell=***cell*] [**−S** | **−-kaspecials**] [**−k** *keytab* | **−-keytab=***keytab*] [**−R** *string* | **−-v5-realm=***string*] [**−D** | **−-decrypt**] [**−E** | **−-encrypt**] [**−n** | **−-stdout**] [**−v** | **−-verbose**] [**−-version**] [**−h** | **−-help**] [*host*[:*port*]] *. . .*

DESCRIPTION

hprop takes a principal database in a specified format and converts it into a stream of Heimdal database records. This stream can either be written to standard out, or (more commonly) be propagated to a hpropd(8) server running on a different machine.

If propagating, it connects to all *hosts* specified on the command by opening a TCP connection to port 754 (service hprop) and sends the database in encrypted form.

Supported options:

−m *file*, **−-master-key=**file
: Where to find the master key to encrypt or decrypt keys with.

−d *file*, **−-database=**file
: The database to be propagated.

−-source=*heimdal*|*mit-dump*|*krb4-dump*|*kaserver*
: Specifies the type of the source database. Alternatives include:

heimdal	a Heimdal database
mit-dump	a MIT Kerberos 5 dump file
krb4-dump	a Kerberos 4 dump file
kaserver	an AFS kaserver database

−k *keytab*, **−-keytab=***keytab*
: The keytab to use for fetching the key to be used for authenticating to the propagation daemon(s). The key kadmin/hprop is used from this keytab. The default is to fetch the key from the KDC database.

−R *string*, **−-v5-realm=***string*
: Local realm override.

−D, **−-decrypt**
: The encryption keys in the database can either be in clear, or encrypted with a master key. This option transmits the database with unencrypted keys.

−E, **−-encrypt**
: This option transmits the database with encrypted keys.

−n, **−-stdout**
: Dump the database on stdout, in a format that can be fed to hpropd.

The following options are only valid if **hprop** is compiled with support for Kerberos 4 (kaserver).

−r *string*, **−-v4-realm=***string*
: v4 realm to use.

−c *cell*, **--cell=***cell*
The AFS cell name, used if reading a kaserver database.

−S, **--kaspecials**
Also dump the principals marked as special in the kaserver database.

−K, **--ka-db**
Deprecated, identical to '--source=kaserver'.

EXAMPLES

The following will propagate a database to another machine (which should run hpropd(8)):

```
$ hprop slave-1 slave-2
```

Convert a Kerberos 4 dump-file for use with a Heimdal KDC:

```
$ hprop -n --source=krb4-dump -d /var/kerberos/principal.dump --master-key=/.k |
```

SEE ALSO

hpropd(8)

NAME

hpropd — receive a propagated database

SYNOPSIS

hpropd [**−d** *file* | **−-database=***file*] [**−n** | **−-stdin**] [**−-print**] [**−i** | **−-no-inetd**] [**−k** *keytab* | **−-keytab=***keytab*] [**−4** | **−-v4dump**]

DESCRIPTION

hpropd receives a database sent by **hprop**. and writes it as a local database.

By default, **hpropd** expects to be started from **inetd** if stdin is a socket and expects to receive the dumped database over stdin otherwise. If the database is sent over the network, it is authenticated and encrypted. Only connections authenticated with the principal **kadmin**/**hprop** are accepted.

Options supported:

−d *file*, **−-database=***file*
database

−n, **−-stdin**
read from stdin

−-print
print dump to stdout

−i, **−-no-inetd**
not started from inetd

−k *keytab*, **−-keytab=***keytab*
keytab to use for authentication

−4, **−-v4dump**
create v4 type DB

SEE ALSO

hprop(8)

NAME

ian — refuse-based internet access node - ftp/http file system

SYNOPSIS

ian [**−v**] [**−d** *cache-directory*] *mount_point*

DESCRIPTION

The **ian** utility is a refuse-based virtual file system, and can be used to access files on remote machines via the ftp and http protocols. The **ian** utility makes use of the virtdir(3) virtual directory routines, and the libfetch remote access routines from FreeBSD.

The URL is provided as a path below the mountpoint, and is retrieved from the remote location using the associated scheme. The file is cached in the local cache directory, which can be manipulated using the *d* command line option. The default location is /tmp.

The **ian** utility makes use of the virtdir(3) virtual directory routines.

The refuse(3) library is used to provide the file system features.

The mandatory parameter is the local mount point.

The **ian** utility was designed to be a BSD-licensed, refuse-based implementation of the functionality provided by the older alex utility.

SEE ALSO

librefuse(3), puffs(3), virtdir(3).

HISTORY

The **ian** utility first appeared in NetBSD 5.0.

NAME

icfs — refuse-based virtual file system to display a case-insensitive interface to a file system

SYNOPSIS

icfs [**-v**] *directory mount_point*

DESCRIPTION

The **icfs** utility can be used to mount an existing directory on a new mount point. The **icfs** utility makes use of the virtdir(3) virtual directory routines. Underneath those virtual directories, the individual directory entries will be displayed as an exact mirror of the original directory, except that any capital letters in the original entry's name will be substituted with an entry name consisting entirely of lower-case letters.

The refuse(3) library is used to provide the file system features.

SEE ALSO

librefuse(3), puffs(3), virtdir(3).

HISTORY

The **icfs** utility first appeared in NetBSD 5.0.

NAME

id3fs — refuse-based virtual file system to display mp3 tree by id3 tags

SYNOPSIS

id3fs [**−v**] *file ...* **−p** *music-directory mount_point*

DESCRIPTION

The **id3fs** utility can be used to mount a number of directory tree under the mount point, using id3 tags to make the artists, genres and years appear as virtual directories. The **id3fs** utility makes use of the virtdir(3) virtual directory routines. Underneath those virtual directories, the individual trackes will be displayed by category, allowing a different view of the id3-tagged files from the standard one.

The refuse(3) library is used to provide the file system features.

The mandatory parameter is the local mount point.

The id3 package, found in pkgsrc/audio/id3, is necessary to run this package.

Two shel lscripts are also included, which must be used to build up an id3 database in advance. First id3info.sh must be run, followed by id3db.sh.

SEE ALSO

librefuse(3), puffs(3), virtdir(3).

HISTORY

The **id3fs** utility first appeared in NetBSD 5.0.

NAME

identd — TCP/IP Ident protocol server

SYNOPSIS

identd [**−46beIilNnr**] [**−a** *address*] [**−c** *charset*] [**−F** *format*] [**−f** *username*]
[**−g** *uid*] [**−L** *username*] [**−m** *filter*] [**−o** *osname*] [**−P** *address*]
[**−p** *portno*] [**−t** *seconds*] [**−u** *uid*]

DESCRIPTION

identd is a TCP/IP server which implements the user identification protocol as specified in RFC 1413.

identd operates by looking up specific TCP/IP connections and returning information which may or may not be associated with the process owning the connection.

The following options are available:

−4 Bind to IPv4 addresses only (valid with flag **−b**).

−6 Bind to IPv6 addresses only (valid with flag **−b**).

−a *address* Bind to the specified *address*. This may be an IPv4 or IPv6 address or even a hostname. If a hostname is specified then **identd** will resolve it to an address (or addresses) and will bind this address (valid with flag **−b**).

−b Run in the background (as daemon).

−c *charset* Specify an optional character set designator to be included in replies. *charset* should be a valid charset set as described in the MIME RFC in upper case characters.

−e Return “UNKNOWN-ERROR” instead of the usual “NO-USER” or “INVALID-PORT” error replies.

−F *format* Specify the format to display info. The allowed format specifiers are:

```
%u      print user name
%U      print user number
%g      print (primary) group name
%G      print (primary) group number
%l      print list of all groups by name
%L      print list of all groups by number
```

The lists of groups (%l, %L) are comma-separated, and start with the primary group which is not repeated. Any other characters (preceded by %, and those not preceded by it) are printed literally.

−f *username* Specify a fall back *username*. If the lookup fails then this username will be returned. This can be useful for when running this service on a NAT host and not using the forward/proxy functionality.

−g *gid* Specify the group id number or name which the server should switch to after binding itself to the TCP/IP port.

−I Same as **−i** but without the restriction that the username in `.ident` must not match an existing user.

−i If the `.ident` file exists in the home directory of the identified user, return the username found in that file instead of the real username. If the username found in `.ident` is that of an existing user, then the real username will be returned.

−L *username* Specify a "lie" *username*. **identd** will return this name for all valid ident requests.

−l Use syslogd(8) for logging purposes.

−m *filter* Enables forwarding of ident queries. The *filter* argument specifies which packet filter should be used to lookup the connections, currently 'pf' and 'ipfilter' are supported packet filters. Note that **identd** changes the ident queries to use the local port on the NAT host instead of the local port on the forwarding host. This is needed because otherwise we can't do a lookup on the proxy host. On the proxy host, "proxy mode" should be enabled with the **−P** flag or "lying mode" with the **−L** flag.

−N Enable .noident files. If this file exists in the home directory of the identified user then return "HIDDEN-USER" instead of the normal USERID response.

−n Return numeric user IDs instead of usernames.

−o *osname* Return *osname* instead of the default "UNIX".

−P *address* Specify a proxy server which will be used to receive proxied ident queries from. See also the **−m** flag how this operates.

−p *portno* Specify an alternative port number under which the server should run. The default is port 113 (valid with flag **−b**).

−r Return a random name of alphanumeric characters. If the **−n** flag is also enabled then a random number will be returned.

−t *seconds* Specify a timeout for the service. The default timeout is 30 seconds.

−u *uid* Specify the user id number or name to which the server should switch after binding itself to the TCP/IP port.

FILES

/etc/inetd.conf

EXAMPLES

identd operates from inetd(8) or as standalone daemon. Put the following lines into inetd.conf(5) to enable **identd** as an IPv4 and IPv6 service via inetd:

```
ident stream tcp nowait nobody /usr/libexec/identd identd -l
```

```
ident stream tcp6 nowait nobody /usr/libexec/identd identd -l
```

To run **identd** as standalone daemon, use the **−b** flag.

SEE ALSO

inetd.conf(5), inetd(8)

AUTHORS

This implementation of **identd** is written by Peter Postma ⟨peter@NetBSD.org⟩.

CAVEATS

Since **identd** should typically not be run as a privileged user or group, .ident files for use when running with the **−I** or **−i** flags will need to be world accessible. The same applies for .noident files when running with the **−N** flag.

When forwarding is enabled with the **−m** flag then **identd** will need access to either /etc/pf (pf) or /etc/ipnat (ipfilter). Since it's not a good idea to run **identd** under root, you'll need to adjust group owner/permissions to the device(s) and run **identd** under that group.

NAME

ifconfig — configure network interface parameters

SYNOPSIS

ifconfig *interface address_family* [*address* [*dest_address*]] [*parameters*]
ifconfig [**−hLmvz**] *interface* [*protocol_family*]
ifconfig −a [**−bdhLmsuvz**] [*protocol_family*]
ifconfig −l [**−bdsu**]
ifconfig −s *interface*
ifconfig −C

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters.

Available operands for **ifconfig**:

address
For the DARPA-Internet family, the address is either a host name present in the host name data base, hosts(5), or a DARPA Internet address expressed in the Internet standard "dot notation". For the Xerox Network Systems(tm) family, addresses are *net:a.b.c.d.e.f*, where *net* is the assigned network number (in decimal), and each of the six bytes of the host number, *a* through *f*, are specified in hexadecimal. The host number may be omitted on Ethernet interfaces, which use the hardware physical address, and on interfaces other than the first. For the ISO family, addresses are specified as a long hexadecimal string, as in the Xerox family. However, two consecutive dots imply a zero byte, and the dots are optional, if the user wishes to (carefully) count out long strings of digits in network byte order.

address_family
Specifies the *address_family* which affects interpretation of the remaining parameters. Since an interface can receive transmissions in differing protocols with different naming schemes, specifying the address family is recommended. The address or protocol families currently supported are "inet", "inet6", "atalk", "iso", and "ns".

interface
The *interface* parameter is a string of the form "name unit", for example, "en0"

The following parameters may be set with **ifconfig**:

advbase *n*
If the driver is a carp(4) pseudo-device, set the base advertisement interval to *n* seconds. This ia an 8-bit number; the default value is 1 second.

advskew *n*
If the driver is a carp(4) pseudo-device, skew the advertisement interval by *n*. This is an 8-bit number; the default value is 0.

Taken together the **advbase** indicate how frequently, in seconds, the host will advertise the fact that it considers itself the master of the virtual host. The formula is **advbase** + (**advskew** / 256). If the master does not advertise within three times this interval, this host will begin advertising as master.

alias
Establish an additional network address for this interface. This is sometimes useful when changing network numbers, and one wishes to accept packets addressed to the old interface.

-alias	Remove the specified network address alias.
arp	Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and Ethernet addresses.
-arp	Disable the use of the Address Resolution Protocol.
anycast	(inet6 only) Set the IPv6 anycast address bit.
-anycast	(inet6 only) Clear the IPv6 anycast address bit.
broadcast *mask*	(Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1’s.
carpdev *iface*	If the driver is a carp(4) pseudo-device, attach it to *iface*. If not specified, the kernel will attempt to select an interface with a subnet matching that of the carp interface.
debug	Enable driver dependent debugging code; usually, this turns on extra console error logging.
-debug	Disable driver dependent debugging code.
delete	Remove the network address specified. This would be used if you incorrectly specified an alias, or it was no longer needed. If you have incorrectly set an NS address having the side effect of specifying the host portion, removing all NS addresses will allow you to respecify the host portion. **delete** does not work for IPv6 addresses. Use **-alias** with explicit IPv6 address instead.
dest_address	Specify the address of the correspondent on the other end of a point to point link.
down	Mark an interface ‘‘down’’. When an interface is marked ‘‘down’’, the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.
ipdst	This is used to specify an Internet host who is willing to receive ip packets encapsulating NS packets bound for a remote network. An apparent point to point link is constructed, and the address specified will be taken as the NS address and network of the destination. IP encapsulation of CLNP packets is done differently.
media *type*	Set the media type of the interface to *type*. Some interfaces support the mutually exclusive use of one of several different physical media connectors. For example, a 10Mb/s Ethernet interface might support the use of either AUI or twisted pair connectors. Setting the media type to “10base5” or “AUI” would change the currently active connector to the AUI port. Setting it to “10baseT” or “UTP” would activate twisted pair. Refer to the interfaces’ driver specific man page for a complete list of the available types.
mediaopt *opts*	Set the specified media options on the interface. *opts* is a comma delimited list of options to apply to the interface. Refer to the interfaces’ driver specific man page for a complete list of available options.
-mediaopt *opts*	Disable the specified media options on the interface.
mode *mode*	If the driver supports the media selection system, set the specified operating mode on the interface to *mode*. For IEEE 802.11 wireless interfaces that support multiple operating modes this directive is used to select between 802.11a (“11a”), 802.11b (“11b”), and 802.11g (“11g”) operating modes.

instance *minst* Set the media instance to *minst*. This is useful for devices which have multiple physical layer interfaces (PHYs). Setting the instance on such devices may not be strictly required by the network interface driver as the driver may take care of this automatically; see the driver's manual page for more information.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (routed(8)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

mtu *n* Set the maximum transmission unit of the interface to *n*. Most interfaces don't support this option.

netmask *mask* (inet, inet6, and ISO) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table networks(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

For INET and INET6 addresses, the netmask can also be given with slash-notation after the address (e.g 192.168.17.3/24).

nsellength *n* (ISO only) This specifies a trailing number of bytes for a received NSAP used for local identification, the remaining leading part of which is taken to be the NET (Network Entity Title). The default value is 1, which is conformant to US GOSIP. When an ISO address is set in an ifconfig command, it is really the NSAP which is being specified. For example, in US GOSIP, 20 hex digits should be specified in the ISO NSAP to be assigned to the interface. There is some evidence that a number different from 1 may be useful for AFI 37 type addresses.

state *state* Explicitly force the carp(4) pseudo-device to enter this state. Valid states are *init*, *backup*, and *master*.

frag *threshold* (IEEE 802.11 devices only) Configure the fragmentation threshold for IEEE 802.11-based wireless network interfaces.

ssid *id* (IEEE 802.11 devices only) Configure the Service Set Identifier (a.k.a. the network name) for IEEE 802.11-based wireless network interfaces. The *id* can either be any text string up to 32 characters in length, or a series of up to 64 hexadecimal digits preceded by "0x". Setting *id* to the empty string allows the interface to connect to any available access point.

nwid *id* Synonym for "ssid".

hidessid (IEEE 802.11 devices only) When operating as an access point, do not broadcast the SSID in beacon frames or respond to probe request frames unless they are directed to the ap (i.e., they include the ap's SSID). By default, the SSID is included in beacon frames and undirected probe request frames are answered.

−hidessid (IEEE 802.11 devices only) When operating as an access point, broadcast the SSID in beacon frames and answer and respond to undirected probe request frames (default).

nwkey *key* (IEEE 802.11 devices only) Enable WEP encryption for IEEE 802.11-based wireless network interfaces with the *key*. The *key* can either be a string, a series of hexadecimal digits preceded by "0x", or a set of keys in the form *n:k1,k2,k3,k4*, where *n* specifies which of keys will be used for all transmitted packets, and four keys, *k1* through *k4*, are configured as WEP keys. Note that the order must be match within same network if multiple keys are used. For IEEE 802.11 wireless network, the length of each key is restricted to 40 bits, i.e. 5-character string or 10 hexadecimal digits, while the WaveLAN/IEEE Gold cards accept the 104 bits (13 characters) key.

nwkey persist (IEEE 802.11 devices only) Enable WEP encryption for IEEE 802.11-based wireless network interfaces with the persistent key written in the network card.

nwkey persist:*key*
(IEEE 802.11 devices only) Write the *key* to the persistent memory of the network card, and enable WEP encryption for IEEE 802.11-based wireless network interfaces with the *key*.

−nwkey (IEEE 802.11 devices only) Disable WEP encryption for IEEE 802.11-based wireless network interfaces.

apbridge (IEEE 802.11 devices only) When operating as an access point, pass packets between wireless clients directly (default).

−apbridge (IEEE 802.11 devices only) When operating as an access point, pass packets through the system so that they can be forwarded using some other mechanism. Disabling the internal bridging is useful when traffic is to be processed with packet filtering.

pass *passphrase*
If the driver is a carp(4) pseudo-device, set the authentication key to *passphrase*. There is no passphrase by default

powersave (IEEE 802.11 devices only) Enable 802.11 power saving mode.

−powersave (IEEE 802.11 devices only) Disable 802.11 power saving mode.

powersavesleep *duration*
(IEEE 802.11 devices only) Set the receiver sleep duration in milliseconds for 802.11 power saving mode.

bssid *bssid* (IEEE 802.11 devices only) Set the desired BSSID for IEEE 802.11-based wireless network interfaces.

−bssid (IEEE 802.11 devices only) Unset the desired BSSID for IEEE 802.11-based wireless network interfaces. The interface will automatically select a BSSID in this mode, which is the default.

chan *chan* (IEEE 802.11 devices only) Select the channel (radio frequency) to be used for IEEE 802.11-based wireless network interfaces.

−chan (IEEE 802.11 devices only) Unset the desired channel to be used for IEEE 802.11-based wireless network interfaces. It doesn't affect the channel to be created for IBSS or hostap mode.

list scan (IEEE 802.11 devices only) Display the access points and/or ad-hoc neighbors located in the vicinity. The **−v** flag may be used to display long SSIDs. **−v** also causes received information elements to be displayed symbolicaly. Only the superuser can use this command.

tunnel *src_addr*[,*src_port*]
dest_addr[,*dest_port*] (IP tunnel devices only) Configure the physical source and destination address for IP tunnel interfaces, including gif(4). The arguments *src_addr* and *dest_addr* are interpreted as the outer source/destination for

the encapsulating IPv4/IPv6 header.

On a `gre`(4) interface in UDP mode, the arguments *src_port* and *dest_port* are interpreted as the outer source/destination port for the encapsulating UDP header.

deletetunnel Unconfigure the physical source and destination address for IP tunnel interfaces previously configured with **tunnel**.

create Create the specified network pseudo-device.

destroy Destroy the specified network pseudo-device.

pltime *n* (inet6 only) Set preferred lifetime for the address.

prefixlen *n* (inet and inet6 only) Effect is similar to **netmask**. but you can specify by prefix length by digits.

deprecated (inet6 only) Set the IPv6 deprecated address bit.

−deprecated (inet6 only) Clear the IPv6 deprecated address bit.

tentative (inet6 only) Set the IPv6 tentative address bit.

−tentative (inet6 only) Clear the IPv6 tentative address bit.

eui64 (inet6 only) Fill interface index (lowermost 64bit of an IPv6 address) automatically.

link[0-2] Enable special processing of the link level of the interface. These three options are interface specific in actual effect, however, they are in general used to select special modes of operation. An example of this is to enable SLIP compression, or to select the connector type for some ethernet cards. Refer to the man page for the specific driver for more information.

−link[0-2] Disable special processing at the link level with the specified interface.

up Mark an interface ‘‘up’’. This may be used to enable an interface after an ‘‘ifconfig down.’’ It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

vhid *n* If the driver is a `carp`(4) pseudo-device, set the virtual host ID to *n*. Acceptable values are 1 to 255.

vlan *vid* If the interface is a `vlan`(4) pseudo-interface, set the VLAN identifier to *vid*. These are the first 12 bits (0-4095) from a 16-bit integer used to create an 802.1Q VLAN header for packets sent from the `vlan`(4) interface. Note that **vlan** and **vlanif** must be set at the same time.

vlanif *iface* If the interface is a `vlan`(4) pseudo-interface, associate the physical interface *iface* with it. Packets transmitted through the `vlan`(4) interface will be diverted to the specified physical interface *iface* with 802.1Q VLAN encapsulation. Packets with 802.1Q encapsulation received by the physical interface with the correct VLAN tag will be diverted to the associated `vlan`(4) pseudo-interface. The VLAN interface is assigned a copy of the physical interface's flags and Ethernet address. If the `vlan`(4) interface already has a physical interface associated with it, this command will fail. To change the association to another physical interface, the existing association must be cleared first. Note that **vlanif** and **vlan** must be set at the same time.

agrport *iface* Add *iface* to the `agr`(4) interface.

-agrport *iface*	Remove *iface* from the `agr`(4) interface.
vltime *n*	(inet6 only) Set valid lifetime for the address.
ip4csum	Shorthand of "ip4csum-tx ip4csum-rx"
-ip4csum	Shorthand of "-ip4csum-tx -ip4csum-rx"
tcp4csum	Shorthand of "tcp4csum-tx tcp4csum-rx"
-tcp4csum	Shorthand of "-tcp4csum-tx -tcp4csum-rx"
udp4csum	Shorthand of "udp4csum-tx udp4csum-rx"
-udp4csum	Shorthand of "-udp4csum-tx -udp4csum-rx"
tcp6csum	Shorthand of "tcp6csum-tx tcp6csum-rx"
-tcp6csum	Shorthand of "-tcp6csum-tx -tcp6csum-rx"
udp6csum	Shorthand of "udp6csum-tx udp6csum-rx"
-udp6csum	Shorthand of "-udp6csum-tx -udp6csum-rx"
ip4csum-tx	Enable hardware-assisted IPv4 header checksums for the out-bound direction.
-ip4csum-tx	Disable hardware-assisted IPv4 header checksums for the out-bound direction.
ip4csum-rx	Enable hardware-assisted IPv4 header checksums for the in-bound direction.
-ip4csum-rx	Disable hardware-assisted IPv4 header checksums for the in-bound direction.
tcp4csum-tx	Enable hardware-assisted TCP/IPv4 checksums for the out-bound direction.
-tcp4csum-tx	Disable hardware-assisted TCP/IPv4 checksums for the out-bound direction.
tcp4csum-rx	Enable hardware-assisted TCP/IPv4 checksums for the in-bound direction.
-tcp4csum-rx	Disable hardware-assisted TCP/IPv4 checksums for the in-bound direction.
udp4csum-tx	Enable hardware-assisted UDP/IPv4 checksums for the out-bound direction.
-udp4csum-tx	Disable hardware-assisted UDP/IPv4 checksums for the out-bound direction.
udp4csum-rx	Enable hardware-assisted UDP/IPv4 checksums for the in-bound direction.
-udp4csum-rx	Disable hardware-assisted UDP/IPv4 checksums for the in-bound direction.
tcp6csum-tx	Enable hardware-assisted TCP/IPv6 checksums for the out-bound direction.
-tcp6csum-tx	Disable hardware-assisted TCP/IPv6 checksums for the out-bound direction.
tcp6csum-rx	Enable hardware-assisted TCP/IPv6 checksums for the in-bound direction.
-tcp6csum-rx	Disable hardware-assisted TCP/IPv6 checksums for the in-bound direction.
udp6csum-tx	Enable hardware-assisted UDP/IPv6 checksums for the out-bound direction.
-udp6csum-tx	Disable hardware-assisted UDP/IPv6 checksums for the out-bound direction.
udp6csum-rx	Enable hardware-assisted UDP/IPv6 checksums for the in-bound direction.
-udp6csum-rx	Disable hardware-assisted UDP/IPv6 checksums for the in-bound direction.
tso4	Enable hardware-assisted TCP/IPv4 segmentation on interfaces that support it.

-tso4	Disable hardware-assisted TCP/IPv4 segmentation on interfaces that support it.
tso6	Enable hardware-assisted TCP/IPv6 segmentation on interfaces that support it.
-tso6	Disable hardware-assisted TCP/IPv6 segmentation on interfaces that support it.

ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family.

If the **−s** flag is passed before an interface name, **ifconfig** will attempt to query the interface for its media status. If the interface supports reporting media status, and it reports that it does not appear to be connected to a network, **ifconfig** will exit with status of 1 (false); otherwise, it will exit with a zero (true) exit status. Not all interface drivers support media status reporting.

If the **−m** flag is passed before an interface name, **ifconfig** will display all of the supported media for the specified interface. If the **−L** flag is supplied, address lifetime is displayed for IPv6 addresses, as time offset string.

Optionally, the **−a** flag may be used instead of an interface name. This flag instructs **ifconfig** to display information about all interfaces in the system. **−d** limits this to interfaces that are down, **−u** limits this to interfaces that are up, **−b** limits this to broadcast interfaces, and **−s** omits interfaces which appear not to be connected to a network.

The **−l** flag may be used to list all available interfaces on the system, with no other additional information. Use of this flag is mutually exclusive with all other flags and commands, except for **−d** (only list interfaces that are down) , **−u** (only list interfaces that are up) , **−s** (only list interfaces that may be connected) , **−b** (only list broadcast interfaces) .

The **−C** flag may be used to list all of the interface cloners available on the system, with no additional information. Use of this flag is mutually exclusive with all other flags and commands.

The **−v** flag prints statistics on packets sent and received on the given interface. If **−h** is used in conjunction with **−v**, the byte statistics will be printed in "human-readable" format. The **−z** flag is identical to the **−v** flag except that it zeros the interface input and output statistics after printing them.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

netstat(1), agr(4), carp(4), ifmedia(4), netintro(4), vlan(4), ifconfig.if(5), rc(8), routed(8)

HISTORY

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

NAME

ifmcstat — dump multicast group management statistics per interface

SYNOPSIS

ifmcstat

DESCRIPTION

The **ifmcstat** command dumps multicast group information in the kernel.

There are no command-line options.

NAME

ifwatchd — watch for addresses added to or deleted from interfaces and call up/down-scripts for them

SYNOPSIS

ifwatchd [**-hiqv**] [**-A** *arrival-script*] [**-c** *carrier-script*]
[**-D** *departure-script*] [**-d** *down-script*] [**-u** *up-script*]
[**-n** *no-carrier-script*] *ifname(s)*

DESCRIPTION

ifwatchd is used to monitor dynamic interfaces (for example PPP interfaces) for address changes, and to monitor static interfaces for carrier changes. Sometimes these interfaces are accompanied by a daemon program, which can take care of running any necessary scripts (like `pppd`(8) or `isdnd`(8)), but sometimes the interfaces run completely autonomously (like `pppoe`(4)).

ifwatchd provides a generic way to watch these types of changes. It works by monitoring the routing socket and interpreting `RTM_NEWADDR` (address added), `RTM_DELADDR` (address deleted) and `RTM_IFINFO` (carrier detect or loss of carrier) messages. It does not need special privileges to do this. The scripts called for up or down events are run with the same user id as **ifwatchd** is run.

The following options are available:

-A *arrival-script*
Specify the command to invoke on arrival of new interfaces (like PCMCIA cards).

-c *carrier-script*
Specify the command to invoke when the carrier status transitions from no carrier to carrier.

-D *departure-script*
Specify the command to invoke when an interface departs (for example a PCMCIA card is removed.)

-d *down-script*
Specify the command to invoke on "interface down" events (or: deletion of an address from an interface).

-h Show the synopsis.

-i Inhibit a call to the up-script on startup for all watched interfaces already marked up. If this option is not given, **ifwatchd** will check all watched interfaces on startup whether they are already marked up and, if they are, call the up-script with appropriate parameters. Additionally, if the interface is up and has a link, **ifwatchd** will run the carrier script.

Since ifwatchd typically is started late in the system boot sequence, some of the monitored interfaces may already have come up when it finally starts, but their up-scripts have not been called. By default **ifwatchd** calls them on startup to account for this (and make the scripts easier.)

-n *no-carrier-script*
Specify the command to invoke when the carrier status transitions from carrier to no carrier.

-q Be quiet and don't log non-error messages to syslog.

-u *up-script*
Specify the command to invoke on "interface up" events (or: addition of an address to an interface).

-v Run in verbose debug mode and do not detach from the controlling terminal. Output verbose progress messages and flag errors ignored during normal operation. *You do not want to use this option in* `/etc/rc.conf`!

ifname(s)
The name of the interface to watch. Multiple interfaces may be specified. Events for other interfaces are ignored.

EXAMPLES

```
# ifwatchd -u /etc/ppp/ip-up -d /etc/ppp/ip-down pppoe0
```

If your pppoe0 interface is your main connection to the internet, the typical use of the up/down scripts is to add and remove a default route. This is an example for an up script doing this:

```
#! /bin/sh
/sbin/route add default $5
```

As described below the fifth command line parameter will contain the peer address of the pppoe link. The corresponding ip-down script is:

```
#! /bin/sh
/sbin/route delete default $5
```

Note that this is not a good idea if you have pppoe0 configured to connect only on demand (via the link1 flag), but works well for all permanent connected cases. Use

```
! /sbin/route add default -iface 0.0.0.1
```

in your `/etc/ifconfig.pppoe0` file in the on-demand case.

The next example is for dhclient users.

```
# ifwatchd -i -c /etc/dhcp/carrier-detect tlp0
```

With the above command, the carrier-detect script will be invoked when a carrier is detected on the interface *tlp0*. Note that the *-i* flag prevents any action based on the initial state. A script like the following should work for most users, although it will not work for machines with multiple interfaces running **dhclient**.

```
#! /bin/sh
# Arguments:  ifname tty speed address destination
# If there is a dhclient already running, kill it.
# (This step could be put in a distinct no-carrier script,
# if desired.)
if [ -f /var/run/dhclient.pid ]; then
       /bin/kill ‘/bin/cat /var/run/dhclient.pid‘
fi
# Start dhclient again on this interface
/sbin/dhclient $1
```

PARAMETERS PASSED TO SCRIPTS

The invoked scripts get passed these parameters:

ifname	The name of the interface this change is for (this allows to share the same script for multiple interfaces watched and dispatching on the interface name in the script).
tty	Dummy parameter for compatibility with pppd(8) which will always be */dev/null*.
speed	Dummy parameter for compatibility with pppd(8) which will always be *9600*.
address	The new address if this is an up event, or the no longer valid old address if this is a down event.

The format of the address depends on the address family, for IPv4 it is the usual dotted quad notation, for IPv6 the colon separated standard notation.

destination For point to point interfaces, this is the remote address of the interface. For other interfaces it is the broadcast address.

ERRORS

The program logs to the syslog daemon as facility "daemon". For detailed debugging use the **-v** (verbose) option.

SEE ALSO

`pppoe`(4), `route`(4), `ifconfig.if`(5), `rc.d`(8), `route`(8)

HISTORY

The **ifwatchd** utility appeared in NetBSD 1.6.

AUTHORS

The program was written by Martin Husemann ⟨martin@NetBSD.org⟩.

CAVEATS

Due to the nature of the program a lot of stupid errors can not easily be caught in advance without removing the provided facility for advanced uses. For example typing errors in the interface name can not be detected by checking against the list of installed interfaces, because it is possible for a pcmcia card with the name given to be inserted later.

NAME

inetd, **inetd.conf** — internet "super-server"

SYNOPSIS

inetd [**-d**] [**-l**] [*configuration file*]

DESCRIPTION

inetd should be run at boot time by `/etc/rc` (see `rc`(8)). It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. After the program is finished, it continues to listen on the socket (except in some cases which will be described below). Essentially, **inetd** allows running one daemon to invoke several others, reducing load on the system.

The options available for **inetd**:

-d Turns on debugging.

-l Turns on libwrap connection logging.

Upon execution, **inetd** reads its configuration information from a configuration file which, by default, is `/etc/inetd.conf`. The path given for this configuration file must be absolute, unless the **-d** option is also given on the command line. There must be an entry for each field of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a "#" at the beginning of a line. There must be an entry for each field (except for one special case, described below). The fields of the configuration file are as follows:

```
[addr:]service-name
socket-type
protocol[,sndbuf=size][,rcvbuf=size]
wait/nowait[:max]
user[:group]
server-program
server program arguments
```

To specify an *Sun-RPC* based service, the entry would contain these fields.

```
service-name/version
socket-type
rpc/protocol[,sndbuf=size][,rcvbuf=size]
wait/nowait[:max]
user[:group]
server-program
server program arguments
```

For Internet services, the first field of the line may also have a host address specifier prefixed to it, separated from the service name by a colon. If this is done, the string before the colon in the first field indicates what local address **inetd** should use when listening for that service, or the single character "*" to indicate `INADDR_ANY`, meaning 'all local addresses'. To avoid repeating an address that occurs frequently, a line with a host address specifier and colon, but no further fields, causes the host address specifier to be remembered and used for all further lines with no explicit host specifier (until another such line or the end of the file). A line

```
*:
```

is implicitly provided at the top of the file; thus, traditional configuration files (which have no host address specifiers) will be interpreted in the traditional manner, with all services listened for on all local addresses.

The *service-name* entry is the name of a valid service in the file `/etc/services`. For "internal" services (discussed below), the service name *must* be the official name of the service (that is, the first entry in `/etc/services`). When used to specify a *Sun-RPC* based service, this field is a valid RPC service name in the file `/etc/rpc`. The part on the right of the "/" is the RPC version number. This can simply be a single numeric argument or a range of versions. A range is bounded by the low version to the high version − "rusers/1-3".

The *socket-type* should be one of "stream", "dgram", "raw", "rdm", or "seqpacket", depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket.

The *protocol* must be a valid protocol as given in `/etc/protocols`. Examples might be "tcp" and "udp". Rpc based services are specified with the "rpc/tcp" or "rpc/udp" service type. "tcp" and "udp" will be recognized as "TCP or UDP over default IP version". It is currently IPv4, but in the future it will be IPv6. If you need to specify IPv4 or IPv6 explicitly, use something like "tcp4" or "udp6". If you would like to enable special support for `faithd`(8), prepend a keyword "faith" into *protocol*, like "faith/tcp6".

In addition to the protocol, the configuration file may specify the send and receive socket buffer sizes for the listening socket. This is especially useful for TCP as the window scale factor, which is based on the receive socket buffer size, is advertised when the connection handshake occurs, thus the socket buffer size for the server must be set on the listen socket. By increasing the socket buffer sizes, better TCP performance may be realized in some situations. The socket buffer sizes are specified by appending their values to the protocol specification as follows:

```
tcp,rcvbuf=16384
tcp,sndbuf=64k
tcp,rcvbuf=64k,sndbuf=1m
```

A literal value may be specified, or modified using 'k' to indicate kilobytes or 'm' to indicate megabytes. Socket buffer sizes may be specified for all services and protocols except for tcpmux services.

The *wait/nowait* entry is used to tell **inetd** if it should wait for the server program to return, or continue processing connections on the socket. If a datagram server connects to its peer, freeing the socket so **inetd** can receive further messages on the socket, it is said to be a "multi-threaded" server, and should use the "nowait" entry. For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be "single-threaded" and should use a "wait" entry. `comsat`(8) (`biff`(1)) and `talkd`(8) are both examples of the latter type of datagram server. `tftpd`(8) is an exception; it is a datagram server that establishes pseudo-connections. It must be listed as "wait" in order to avoid a race; the server reads the first packet, creates a new socket, and then forks and exits to allow **inetd** to check for new service requests to spawn new servers. The optional "max" suffix (separated from "wait" or "nowait" by a dot or a colon) specifies the maximum number of server instances that may be spawned from **inetd** within an interval of 60 seconds. When omitted, "max" defaults to 40. If it reaches this maximum spawn rate, **inetd** will log the problem (via the syslogger using the LOG_DAEMON facility and LOG_ERR level) and stop handling the specific service for ten minutes.

Stream servers are usually marked as "nowait" but if a single server process is to handle multiple connections, it may be marked as "wait". The master socket will then be passed as fd 0 to the server, which will then need to accept the incoming connection. The server should eventually time out and exit when no more connections are active. **inetd** will continue to listen on the master socket for connections, so the server should not close it when it exits. `identd`(8) is usually the only stream server marked as wait.

The *user* entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root. Optionally, a group can be specified by appending a colon to the user name, followed by the group name (it is possible to use a dot (".") in lieu of a colon, however this feature is provided only for backward compatibility). This allows for servers to run with a different (primary) group id than specified in the password file. If a group is specified and *user* is not root, the supplementary groups associated with that user will still be set.

The *server-program* entry should contain the pathname of the program which is to be executed by **inetd** when a request is found on its socket. If **inetd** provides this service internally, this entry should be "internal".

The *server program arguments* should be just as arguments normally are, starting with argv[0], which is the name of the program. If the service is provided internally, the word "internal" should take the place of this entry. It is possible to quote an argument using either single or double quotes. This allows you to have, e.g., spaces in paths and parameters.

Internal Services

inetd provides several "trivial" services internally by use of routines within itself. These services are "echo", "discard", "chargen" (character generator), "daytime" (human readable time), and "time" (machine readable time, in the form of the number of seconds since midnight, January 1, 1900 GMT). For details of these services, consult the appropriate RFC.

TCP services without official port numbers can be handled with the RFC1078-based tcpmux internal service. TCPmux listens on port 1 for requests. When a connection is made from a foreign host, the service name requested is passed to TCPmux, which performs a lookup in the service name table provided by /etc/inetd.conf and returns the proper entry for the service. TCPmux returns a negative reply if the service doesn't exist, otherwise the invoked server is expected to return the positive reply if the service type in /etc/inetd.conf file has the prefix "tcpmux/". If the service type has the prefix "tcpmux/+", TCP-mux will return the positive reply for the process; this is for compatibility with older server code, and also allows you to invoke programs that use stdin/stdout without putting any special server code in them. Services that use TCPmux are "nowait" because they do not have a well-known port number and hence cannot listen for new requests.

inetd rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread. **inetd** creates a file */var/run/inetd.pid* that contains its process identifier.

libwrap

Support for TCP wrappers is included with **inetd** to provide internal tcpd-like access control functionality. An external tcpd program is not needed. You do not need to change the /etc/inetd.conf server-program entry to enable this capability. **inetd** uses /etc/hosts.allow and /etc/hosts.deny for access control facility configurations, as described in hosts_access(5).

Nota Bene: TCP wrappers do not affect/restrict UDP or internal services.

IPsec

The implementation includes a tiny hack to support IPsec policy settings for each socket. A special form of the comment line, starting with "#@", is used as a policy specifier. The content of the above comment line will be treated as a IPsec policy string, as described in ipsec_set_policy(3). Multiple IPsec policy strings may be specified by using a semicolon as a separator. If conflicting policy strings are found in a single line, the last string will take effect. A #@ line affects all of the following lines in /etc/inetd.conf, so you may want to reset the IPsec policy by using a comment line containing only #@ (with no policy string).

If an invalid IPsec policy string appears in /etc/inetd.conf, **inetd** logs an error message using syslog(3) and terminates itself.

IPv6 TCP/UDP behavior

If you wish to run a server for IPv4 and IPv6 traffic, you'll need to run two separate process for the same server program, specified as two separate lines on /etc/inetd.conf, for "tcp4" and "tcp6". "tcp" means TCP on top of currently-default IP version, which is, at this moment, IPv4.

Under various combination of IPv4/v6 daemon settings, **inetd** will behave as follows:

- If you have only one server on "tcp4", IPv4 traffic will be routed to the server. IPv6 traffic will not be accepted.
- If you have two servers on "tcp4" and "tcp6", IPv4 traffic will be routed to the server on "tcp4", and IPv6 traffic will go to server on "tcp6".
- If you have only one server on "tcp6", only IPv6 traffic will be routed to the server. The kernel may route to the server IPv4 traffic as well, under certain configuration. See `ip6`(4) for details.

FILES

`/etc/inetd.conf`	configuration file for all **inetd** provided services
`/etc/services`	service name to protocol and port number mappings.
`/etc/protocols`	protocol name to protocol number mappings
`/etc/rpc`	Sun-RPC service name to service number mappings.
`/etc/hosts.allow`	explicit remote host access list.
`/etc/hosts.deny`	explicit remote host denial of service list.

SEE ALSO

`hosts_access`(5), `hosts_options`(5), `protocols`(5), `rpc`(5), `services`(5), `comsat`(8), `fingerd`(8), `ftpd`(8), `rexecd`(8), `rlogind`(8), `rshd`(8), `telnetd`(8), `tftpd`(8)

J. Postel, *Echo Protocol*, RFC, 862, May 1983.

J. Postel, *Discard Protocol*, RFC, 863, May 1983.

J. Postel, *Character Generator Protocol*, RFC, 864, May 1983.

J. Postel, *Daytime Protocol*, RFC, 867, May 1983.

J. Postel and K. Harrenstien, *Time Protocol*, RFC, 868, May 1983.

M. Lottor, *TCP port service Multiplexer (TCPMUX)*, RFC, 1078, November 1988.

HISTORY

The **inetd** command appeared in 4.3BSD. Support for *Sun-RPC* based services is modeled after that provided by SunOS 4.1. Support for specifying the socket buffer sizes was added in NetBSD 1.4. In November 1996, libwrap support was added to provide internal tcpd-like access control functionality; libwrap is based on Wietse Venema's tcp_wrappers. IPv6 support and IPsec hack was made by KAME project, in 1999.

BUGS

Host address specifiers, while they make conceptual sense for RPC services, do not work entirely correctly. This is largely because the portmapper interface does not provide a way to register different ports for the same service on different local addresses. Provided you never have more than one entry for a given RPC service, everything should work correctly (Note that default host address specifiers do apply to RPC lines with no explicit specifier.)

"tcpmux" on IPv6 is not tested enough.

SECURITY CONSIDERATIONS

Enabling the "echo", "discard", and "chargen" built-in trivial services is not recommended because remote users may abuse these to cause a denial of network service to or from the local host.

NAME

init — process control initialization

SYNOPSIS

init

DESCRIPTION

The **init** program is the last stage of the boot process (after the kernel loads and initializes all the devices). It normally begins multi-user operation.

The following table describes the state machine used by **init**:

1. Single user shell. **init** may be passed **−s** from the boot program to prevent the system from going multi-user and to instead execute a single user shell without starting the normal daemons. The system is then quiescent for maintenance work and may later be made to go to state 2 (multi-user) by exiting the single-user shell (with ˆD).

2. Multi-user boot (default operation). Executes `/etc/rc` (see `rc`(8)). If this was the first state entered (as opposed to entering here after state 1), then `/etc/rc` will be invoked with its first argument being 'autoboot'. If `/etc/rc` exits with a non-zero (error) exit code, commence single user operation by giving the super-user a shell on the console by going to state 1 (single user). Otherwise, proceed to state 3.

   If value of the "init.root" sysctl node is not equal to `/` at this point, the `/etc/rc` process will be run inside a `chroot`(2) indicated by sysctl with the same error handling as above.

3. Set up ttys as specified in `ttys`(5). See below for more information. On completion, continue to state 4. If we did chroot in state 2, each `getty`(8) process will be run in the same `chroot`(2) path as in 2 (that is, the value of "init.root" sysctl is not re-read).

4. Multi-user operation. Depending upon the signal received, change state appropriately; on `SIGTERM`, go to state 7; on `SIGHUP`, go to state 5; on `SIGTSTP`, go to state 6.

5. Clean-up mode; re-read `ttys`(5), killing off the controlling processes on lines that are now 'off', and starting processes that are newly 'on'. On completion, go to state 4.

6. 'Boring' mode; no new sessions. Signals as per state 4.

7. Shutdown mode. Send `SIGHUP` to all controlling processes, reap the processes for 30 seconds, and then go to state 1 (single user); warning if not all the processes died.

If the 'console' entry in the `ttys`(5) file is marked "insecure", then **init** will require that the superuser password be entered before the system will start a single-user shell. The password check is skipped if the 'console' is marked as "secure".

It should be noted that while **init** has the ability to start multi-user operation inside a `chroot`(2) environment, the **init** process itself will always run in the "original root directory". This also implies that single-user mode is always started in the original root, giving the possibility to create multi-user sessions in different root directories over time. The "init.root" sysctl node is fabricated by **init** at startup and re-created any time it's found to be missing. Type of the node is string capable of holding full pathname, and is only accessible by the superuser (unless explicitly destroyed and re-created with different specification).

In multi-user operation, **init** maintains processes for the terminal ports found in the file `ttys`(5). **init** reads this file, and executes the command found in the second field. This command is usually `getty`(8); it opens and initializes the tty line and executes the `login`(1) program. The `login`(1) program, when a valid user logs in, executes a shell for that user. When this shell dies, either because the user logged out or an abnormal termination occurred (a signal), the **init** program wakes up, deletes the user from the `utmp`(5) and `utmpx`(5) files of current users and records the logout in the `wtmp`(5) and `wtmpx`(5) files. The cycle is then restarted by **init** executing a new getty(8) for the line.

Line status (on, off, secure, getty, or window information) may be changed in the ttys(5) file without a reboot by sending the signal SIGHUP to **init** with the command "kill −s HUP 1". This is referenced in the table above as state 5. On receipt of this signal, **init** re-reads the ttys(5) file. When a line is turned off in ttys(5), **init** will send a SIGHUP signal to the controlling process for the session associated with the line. For any lines that were previously turned off in the ttys(5) file and are now on, **init** executes a new getty(8) to enable a new login. If the getty or window field for a line is changed, the change takes effect at the end of the current login session (e.g., the next time **init** starts a process on the line). If a line is commented out or deleted from ttys(5), **init** will not do anything at all to that line. However, it will complain that the relationship between lines in the ttys(5) file and records in the utmp(5) file is out of sync, so this practice is not recommended.

init will terminate multi-user operations and resume single-user mode if sent a terminate (TERM) signal, for example, "kill −s TERM 1". If there are processes outstanding that are deadlocked (because of hardware or software failure), **init** will not wait for them all to die (which might take forever), but will time out after 30 seconds and print a warning message.

init will cease creating new getty(8)'s and allow the system to slowly die away, if it is sent a terminal stop (TSTP) signal, i.e. "kill −s TSTP 1". A later hangup will resume full multi-user operations, or a terminate will start a single user shell. This hook is used by reboot(8) and halt(8).

The role of **init** is so critical that if it dies, the system will reboot itself automatically. If, at bootstrap time, the **init** process cannot be located, or exits during its initialisation, the system will panic with the message "panic: init died (signal %d, exit %d)".

If /dev/console does not exist, **init** will cd to /dev and run "MAKEDEV -MM init". MAKEDEV(8) will use mount_tmpfs(8) or mount_mfs(8) to create a memory file system mounted over /dev that contains the standard devices considered necessary to boot the system.

FILES

/dev/console	System console device.
/dev/tty*	Terminal ports found in ttys(5).
/var/run/utmp{,x}	Record of current users on the system.
/var/log/wtmp{,x}	Record of all logins and logouts.
/etc/ttys	The terminal initialization information file.
/etc/rc	System startup commands.

DIAGNOSTICS

getty repeating too quickly on port %s, sleeping A process being started to service a line is exiting quickly each time it is started. This is often caused by a ringing or noisy terminal line. *Init will sleep for 10 seconds, then continue trying to start the process*.

some processes would not die; ps axl advised. A process is hung and could not be killed when the system was shutting down. This condition is usually caused by a process that is stuck in a device driver because of a persistent device error condition.

SEE ALSO

config(1), kill(1), login(1), sh(1), options(4), ttys(5), MAKEDEV(8), MAKEDEV.local(8), getty(8), halt(8), mount_mfs(8), mount_tmpfs(8), rc(8), reboot(8), shutdown(8), sysctl(8), secmodel_bsd44(9)

HISTORY

A **init** command appeared in Version 6 AT&T UNIX.

NAME

installboot — install a bootstrap on an FFS filesystem partition

SYNOPSIS

/usr/mdec/installboot [**-l** *newcommandline*] *bootblock device*

DESCRIPTION

installboot copies the bootblock to a bootable partition. The bootstrap is written into the bootblock area on the partition, right in front of the superblock, and hence limited in size to 8192 bytes.

The bootstrap resides in the first few blocks on the partition (as specified by Commodore-Amiga Inc.) The bootstrap is loaded into memory by the ROM from bootable devices: RDB devices, where the partition is marked as bootable, or (not on the DraCo) floppy disks in Amiga format (880K/1760k).

In the presence of more than one bootable partition/floppy disk, the partition is chosen by the bootpriority (from the RDB), which can be overridden by the operator from the boot menu (on Amiga machines, hold down the outer mouse buttons during boot; on DraCo machines, press the left mouse button when prompted).

On RDB devices, the whole bootblock is loaded by the ROM. The number of boot blocks in the RDB partition entry must be correct.

On floppy disks, the ROM always loads the first two blocks (1024 bytes), and the bootblock allocates memory and loads the whole bootblock on startup.

After receiving control, the bootblock uses the stand-alone filesystem code in "libsa.a" to load the kernel from the filesystem on the partition it was started from. The code for the boot program can be found in `/usr/mdec/bootxx_fd` (floppy disk code) or `/usr/mdec/bootxx_ffs` (generic RDB disk code) .

The arguments are:

-l *newcommandline*
: Specify a different command line to replace the default.

bootblock The file containing the bootblock (normally /usr/mdec/bootxx_ffs for RDB devices).

device The name of the character special device specifying the partition on which the bootstrap is to be installed.

EXAMPLES

The following command will install the boot program in the bootblock area on "sd0a":

```
installboot /usr/mdec/bootxx_ffs /dev/rsd0a
```

SEE ALSO

dd(1), boot(8)

HISTORY

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

BUGS

If **installboot** is accidentally used on the whole disk partition, the RDB will be overwritten, making your system unusable after the next reboot.

Some third-party accelerator boards are not autoconfiguring. You won't be able to use their memory when booting from the bootblock after a cold start.

Some third-party disk controllers don't support bootblock booting.

DraCo ROMs don't support bootblock booting from floppy disks.

Most 68060 boards, unlike the DraCo, don't set the SysBase->AttnFlags bit for the 68060 CPU (a patch program which is called during AmigaOS startup does this). You need to add options BB060STUPIDROM to your kernel to boot on such a machine.

There is currently no easy way to edit the RDB from within NetBSD. Therefore, you have to use HDTOOLBOX or a similar tool to set the partition to bootable, "use custom bootblocks" and the number of bootblocks to 16 (for bootxx_ffs) or 2 (for bootxx_fd), at least the first time you install the bootblock.

As normal dd is used to install the bootblock, you can only install onto your currently used root (or any other mounted) partition from single-user mode, or while otherwise running in insecure mode.

NAME

installboot — install a bootstrap on an FFS filesystem partition

SYNOPSIS

/usr/mdec/installboot [**-Nmtuv**] *device*

DESCRIPTION

installboot prepares the (physically) first partition on a device for boot-strapping from the TOS-ROM. The bootstrap is written into the bootblock area on the partition, right in front of the disk pack label, and hence limited in size to `LABELOFFSET` bytes. A disk pack label should be created (see `disklabel`(8)) before installing the bootstrap.

The bootstrap is split into three parts: a small first-stage program that resides in the (physically) first 512 bytes on the device (as specified by Atari Corp.), a second-stage program that immediately follows the first-stage program, and a third-stage program that resides on the root filesystem. The first-stage program is loaded into memory by the ROM. After receiving control, it loads the second-stage program and the disk label. The second-stage boot program uses the stand-alone filesystem code in "libsa.a" to load the third-stage boot program from the root-filesystem on the device. The third-stage boot program then loads the kernel. The prototype code for the first-stage boot program can be found in `/usr/mdec/std/fdboot` (floppy disk code), `/usr/mdec/std/sdboot` (SCSI disk code) and `/usr/mdec/std/wdboot` (IDE disk code). The second-stage boot program is stored in `/usr/mdec/std/bootxx`. and the third-stage boot program is stored in `/usr/mdec/std/boot.atari`. The boot code for Milan machines is different from the other machines and the files for the Milan can be found in the directory `/usr/mdec/milan`. Note that the Milan uses the SCSI disk code for both SCSI and IDE disks.

For backwards compatibility with the vendor specific AHDI disk label, a special first-stage boot program is provided in `/usr/mdec/std/xxboot.ahdi`. Together with the general second-stage boot program, it is installed in the AHDI partition where the NetBSD disk label lives. Furthermore, the AHDI specifications require an additional bootstrap, which is written into the AHDI root sector (disk block zero). The prototype code for this AHDI compliant bootstrap can be found in `/usr/mdec/std/sdb00t.ahdi` and `/usr/mdec/std/wdb00t.ahdi`, or the equivalents in `/usr/mdec/milan`.

Perform the following steps to make a file system bootable:

1. Copy the secondary bootstrap (either `/usr/mdec/std/boot.atari` or `/usr/mdec/milan/boot.atari`) to the root directory of the target file system.
2. Use **installboot** to install the primary and secondary bootstrap programs (from `/usr/mdec/std` or `/usr/mdec/milan`) into the *filesystem*.

The options are as follows:

-N Do not actually write anything on the disk.

-m Use Milan boot code.

-t Number of tracks per cylinder (IDE disk).

-u Number of sectors per track (IDE disk).

-v Verbose mode.

The arguments are:

device The name of the device on which the bootstrap is to be installed.

EXAMPLES

The following command will install the first-stage and second-stage boot programs in the bootblock area on "sd0c":

```
installboot sd0
```

SEE ALSO

`bootpref`(8), `disklabel`(8)

HISTORY

The **`installboot`** command first appeared in NetBSD 1.1

BUGS

`installboot` knows too much about kernel internal details, forcing it to check the running kernel's release and revision.

Because neither the floppy disk driver nor `disklabel`(8) are capable of creating a disk pack label on a floppy disk, **`installboot`** has to create a fictitious label, that is not used by the kernel.

Except for installation of the bootcode on floppy, **`installboot`** automatically sets the boot preference in NVRAM to NetBSD.

NAME

installboot — install a bootstrap on a UFS disk

SYNOPSIS

/usr/mdec/installboot -n | **-v** *ufsboot bootxx rawdev*

DESCRIPTION

installboot is used to install a "first-stage" boot program into the boot area of a UFS disk partition, and initialize the table of block numbers the *bootxx* program uses to load the second-stage boot program.

The options are as follows:

-n Do not actually write anything on the disk.

-v Be verbose, printing out the block numbers that *bootxx* will use to load *ufsboot*.

The arguments are:

ufsboot the name of the second-stage boot program in the file system where the first-stage boot program is to be installed.

bootxx the name of the prototype file for the first stage boot program.

rawdev the name of the raw device in which the first-stage boot program is to be installed. This should correspond to the block device on which the file system containing *ufsboot* is mounted.

SEE ALSO

disklabel(8), init(8)

BUGS

installboot requires simultaneous access to the mounted file system and the raw device, but that is not allowed with the kernel securelevel variable set to a value greater than zero (the default), so **installboot** only works in single-user mode (or insecure mode - see init(8)).

NAME

installboot — install disk bootstrap software

SYNOPSIS

installboot [**-fnv**] [**-B** *s2bno*] [**-b** *s1bno*] [**-m** *machine*] [**-o** *options*]
[**-t** *fstype*] *filesystem primary* [*secondary*]
installboot -c [**-fnv**] [**-m** *machine*] [**-o** *options*] [**-t** *fstype*] *filesystem*
installboot -e [**-fnv**] [**-m** *machine*] [**-o** *options*] *bootstrap*

DESCRIPTION

The **installboot** utility installs and removes NetBSD disk bootstrap software into a file system. **installboot** can install *primary* into *filesystem*, or disable an existing bootstrap in *filesystem*.

One some architectures the options of an existing installed bootstrap, or those of a bootstrap file can be changed.

Generally, NetBSD disk bootstrap software consists of two parts: a "primary" bootstrap program usually written into the disklabel area of the file system by **installboot**, and a "secondary" bootstrap program that usually resides as an ordinary file in the file system.

When booting, the primary bootstrap program is loaded and invoked by the machine's PROM or BIOS. After receiving control of the system it loads and runs the secondary bootstrap program, which in turn loads and runs the kernel. The secondary bootstrap may allow control over various boot parameters passed to the kernel.

Perform the following steps to make a file system bootable:

1. Copy the secondary bootstrap (usually `/usr/mdec/boot.`**MACHINE** or `/usr/mdec/boot`) to the root directory of the target file system.
2. Use **installboot** to install the primary bootstrap program (usually `/usr/mdec/bootxx_`**FSTYPE**) into *filesystem*.

   The following platforms do not require this step if the primary bootstrap already exists and the secondary bootstrap file is just being updated: **alpha**, **amd64**, **amiga**, **i386**, **pmax**, **sparc64**, and **vax**.

   The following platform does not require the first step since a single bootstrap file is used. The single bootstrap is installed like the primary bootstrap on other platforms: **next68k**.

The options and arguments recognized by **installboot** are as follows:

-B *s2bno* When hard-coding the blocks of *secondary* into *primary*, start from block *s2bno* instead of trying to determine the block numbers occupied by *secondary* by examining *filesystem*. If this option is supplied, *secondary* should refer to an actual secondary bootstrap (rather than the file name of the one present in *filesystem*) so that its size can be determined.

-b *s1bno* Install *primary* at block number *s1bno* instead of the default location for the machine and file system type. [**alpha**, **pmax**, **vax**]

-c Clear (remove) any existing bootstrap instead of installing one.

-e Edit the options of an existing bootstrap. This can be use to change the options in bootxx_xxxfs files, raw disk partitions, and the `pxeboot_ia32.bin` file. [**amd64**, **i386**]

-f Forces **installboot** to ignore some errors.

-m *machine*
Use *machine* as the target machine type. The default machine is determined from uname(3) and then MACHINE. The following machines are currently supported by **installboot**:

alpha, **amd64**, **amiga**, **ews4800mips**, **hp300**, **hp700**, **i386**, **landisk**, **macppc**, **news68k**, **newsmips**, **next68k**, **pmax**, **sparc**, **sparc64**, **sun2**, **sun3**, **vax**, **x68k**

-n Do not write to *filesystem*.

-o *options*
Machine specific **installboot** options, comma separated.

Supported options are (with the machines for they are valid in brackets):

alphasum [**alpha**] Recalculate and restore the Alpha checksum. This is the default for NetBSD/alpha.

append [**alpha**, **pmax**, **vax**] Append *primary* to the end of *filesystem*, which must be a regular file in this case.

command=<boot command>
[**amiga**] Modify the default boot command line.

console=<console name>
[**amd64**, **i386**] Set the console device, <console name> must be one of: pc, com0, com1, com2, com3, com0kbd, com1kbd, com2kbd or com3kbd.

ioaddr=<ioaddr>
[**amd64**, **i386**] Set the IO address to be used for the console serial port. Defaults to the IO address used by the system BIOS for the specified port.

keymap=<keymap>
[**amd64**, **i386**] Set a boot time keyboard translation map. Each character in <keymap> will be replaced by the one following it. For example, an argument of “zyz” would swap the lowercase letters ‘y’ and ‘z’.

password=<password>
[**amd64**, **i386**] Set the password which must be entered before the boot menu can be accessed.

resetvideo [**amd64**, **i386**] Reset the video before booting.

speed=<baud rate>
[**amd64**, **i386**] Set the baud rate for the serial console. If a value of zero is specified, then the current baud rate (set by the BIOS) will be used.

sunsum [**alpha**, **pmax**, **vax**] Recalculate and restore the Sun and NetBSD/sparc compatible checksum. *Note*: The existing NetBSD/sparc disklabel should use no more than 4 partitions.

timeout=<seconds>
[**amd64**, **i386**] Set the timeout before the automatic boot begins to the given number of seconds.

-t *fstype* Use *fstype* as the type of *filesystem*. The default operation is to attempt to auto-detect this setting. The following file system types are currently supported by **installboot**:

ffs BSD Fast File System.

raid Mirrored RAIDframe File System.

raw 'Raw' image. Note: if a platform needs to hard-code the block offset of the secondary bootstrap, it cannot be searched for on this file system type, and must be provided with **-B** *s2bno*.

-v Verbose operation.

filesystem The path name of the device or file system image that **installboot** is to operate on. It is not necessary for *filesystem* to be a currently mounted file system.

primary The path name of the "primary" boot block to install.

secondary The path name of the "secondary" boot block, relative to the top of *filesystem*. Most systems require *secondary* to be in the "root" directory of the file system, so the leading "/" is not necessary on *secondary*.

Only certain combinations of platform (**-m** *machine*) and file system type (**-t** *fstype*) require that the name of the secondary bootstrap is supplied as *secondary*, so that information such as the disk block numbers occupied by the secondary bootstrap can be stored in the primary bootstrap. These are:

Platform	**File systems**
macppc	ffs, raw
news68k	ffs, raw
newsmips	ffs, raw
sparc	ffs, raid, raw
sun2	ffs, raw
sun3	ffs, raw

installboot exits 0 on success, and >0 if an error occurs.

ENVIRONMENT

installboot uses the following environment variables:

`MACHINE` Default value for *machine*, overriding the result from `uname`(3).

FILES

Most NetBSD ports will contain variations of the following files:

`/usr/mdec/bootxx_`**FSTYPE** Primary bootstrap for file system type **FSTYPE**. Installed into the bootstrap area of the file system by **installboot**.

`/usr/mdec/bootxx_fat16` Primary bootstrap for MS-DOS **FAT16** file systems. This differs from **bootxx_msdos** in that it doesn't require the filesystem to have been initialised with any `reserved sectors`. It also uses the information in the `Boot Parameter Block` to get the media and filesytem properties.

`/usr/mdec/bootxx_ffsv1` Primary bootstrap for **FFSv1** file systems (the "traditional" NetBSD file system).

`/usr/mdec/bootxx_ffsv2` Primary bootstrap for **FFSv2** file systems.

`/usr/mdec/bootxx_lfsv1` Primary bootstrap for **LFSv1** file systems.

`/usr/mdec/bootxx_lfsv2`	Primary bootstrap for **LFSv2** file systems (the default LFS version).
`/usr/mdec/bootxx_msdos`	Primary bootstrap for MS-DOS **FAT** file systems.
`/usr/mdec/bootxx_ustarfs`	Primary bootstrap for **TARFS** boot images. This is used by various install media.
`/usr/mdec/boot.`**MACHINE**	Secondary bootstrap for machine type **MACHINE**. This should be installed into the file system before **`installboot`** is run.
`/usr/mdec/boot`	Synonym for `/usr/mdec/boot.`**MACHINE**
`/boot.`**MACHINE**	Installed copy of secondary bootstrap for machine type **MACHINE**.
`/boot`	Installed copy of secondary bootstrap. Searched for by the primary bootstrap if `/boot.`**MACHINE** is not found.

NetBSD/macppc files

`/usr/mdec/bootxx`	NetBSD/macppc primary bootstrap.
`/usr/mdec/ofwboot`	NetBSD/macppc secondary bootstrap.
`/ofwboot`	Installed copy of NetBSD/macppc secondary bootstrap.

NetBSD/next68k files

`/usr/mdec/boot`	NetBSD/next68k bootstrap.

NetBSD/sparc64 files

`/usr/mdec/bootblk`	NetBSD/sparc64 primary bootstrap.
`/usr/mdec/ofwboot`	NetBSD/sparc64 secondary bootstrap.
`/ofwboot`	Installed copy of NetBSD/sparc64 secondary bootstrap.

EXAMPLES

common

Verbosely install the Berkeley Fast File System primary bootstrap on to disk 'sd0':

```
installboot -v /dev/rsd0c /usr/mdec/bootxx_ffs
```

Note: the "whole disk" partition (c on some ports, d on others) is used here, since the a partition probably is already opened (mounted as `/`), so **`installboot`** would not be able to access it.

Remove the primary bootstrap from disk 'sd1':

```
installboot -c /dev/rsd1c
```

NetBSD/amiga

Modify the command line to change the default from "netbsd -ASn2" to "netbsd -S":

```
installboot -m amiga -o command="netbsd -S" /dev/rsd0a
/usr/mdec/bootxx_ffs
```

NetBSD/ews4800mips

Install the System V Boot File System primary bootstrap on to disk 'sd0', with the secondary bootstrap '`/boot`' already present in the SysVBFS partition on the disk:

```
installboot /dev/rsd0c /usr/mdec/bootxx_bfs
```

NetBSD/i386 and NetBSD/amd64

Install new boot blocks on an existing mounted root file system on 'wd0', setting the timeout to five seconds, after copying a new secondary bootstrap:

```
cp /usr/mdec/boot /boot
installboot -v -o timeout=5 /dev/rwd0a /usr/mdec/bootxx_ffsv1
```

Create a bootable CD-ROM with an ISO9660 file system for an i386 system with a serial console:

```
mkdir cdrom
cp sys/arch/i386/compile/mykernel/netbsd cdrom/netbsd
cp /usr/mdec/boot cdrom/boot
cp /usr/mdec/bootxx_cd9660 bootxx
installboot -o console=com0,speed=19200 -m i386 -e bootxx
makefs -t cd9660 -o 'bootimage=i386;bootxx,no-emul-boot' boot.iso
      cdrom
```

Create a bootable floppy disk with an FFSv1 file system for a small custom kernel (note: bigger kernels needing multiple disks are handled with the ustarfs file system):

```
newfs -s 1440k /dev/rfd0a
```

Note: Ignore the warnings that `newfs`(8) displays; it can not write a disklabel, which is not a problem for a floppy disk.

```
mount /dev/fd0a /mnt
cp /usr/mdec/boot /mnt/boot
gzip -9 < sys/arch/i386/compile/mykernel/netbsd > /mnt/netbsd.gz
umount /mnt
installboot -v /dev/rfd0a /usr/mdec/bootxx_ffsv1
```

Create a bootable FAT file system on 'wd1a', which should have the same offset and size as a FAT primary partition in the Master Boot Record (MBR):

```
newfs_msdos -r 16 /dev/rwd1a
```

Notes: The **-r** *16* is to reserve space for the primary bootstrap. `newfs_msdos`(8) will display an "MBR type" such as '`1`', '`4`', or '`6`'; the MBR partition type of the appropriate primary partition should be changed to this value.

```
mount -t msdos /dev/wd1a /mnt
cp /usr/mdec/boot /mnt/boot
cp path/to/kernel /mnt/netbsd
umount /mnt
installboot -t raw /dev/rwd1a /usr/mdec/bootxx_msdos
```

Make the existing FAT16 filesystem on 'sd0e' bootable. This can be used to make USB memory bootable provided it has 512 byte sectors and that the manufacturer correctly initialised the file system.

```
mount -t msdos /dev/sd0e /mnt
cp /usr/mdec/boot /mnt/boot
cp path/to/kernel /mnt/netbsd
umount /mnt
installboot /dev/rsd0e /usr/mdec/bootxx_fat16
```

It may also be necessary to use **fdisk** to make the device itself bootable.

NetBSD/macppc

Note the **installboot** utility is only required for macppc machines with OpenFirmware version 2 to boot. OpenFirmware 3 cannot load bootblocks specified in the Apple partition map.

Install the Berkeley Fast File System primary bootstrap on to disk 'wd0':

```
installboot /dev/rwd0c /usr/mdec/bootxx /ofwboot
```

The secondary NetBSD/macppc bootstrap is located in `/usr/mdec/ofwboot`.

The primary bootstrap requires the raw `ofwboot` for the secondary bootstrap, not `ofwboot.xcf`, which is used for the OpenFirmware to load kernels.

NetBSD/next68k

Install the bootstrap on to disk 'sd0':

```
installboot /dev/rsd0c /usr/mdec/boot
```

NetBSD/pmax

Install the Berkeley Fast File System primary bootstrap on to disk 'sd0':

```
installboot /dev/rsd0c /usr/mdec/bootxx_ffs
```

NetBSD/pmax requires that this file system starts at block 0 of the disk.

Install the ISO 9660 primary bootstrap in the file `/tmp/cd-image`:

```
installboot -m pmax /tmp/cd-image /usr/mdec/bootxx_cd9660
```

Make an ISO 9660 filesystem in the file `/tmp/cd-image` and install the ISO 9660 primary bootstrap in the filesystem, where the source directory for the ISO 9660 filesystem contains a kernel, the primary bootstrap `bootxx_cd9660` and the secondary bootstrap `boot.pmax`:

```
mkisofs -o /tmp/cd-image -a -l -v iso-source-dir
...
48 51 iso-source-dir/bootxx_cd9660
...
installboot -b `expr 48 \* 4` /tmp/cd-image /usr/mdec/bootxx_cd9660
```

NetBSD/sparc

Install the Berkeley Fast File System primary bootstrap on to disk 'sd0', with the secondary bootstrap '`/boot`' already present:

```
installboot /dev/rsd0c /usr/mdec/bootxx /boot
```

NetBSD/sparc64

Install the Berkeley Fast File System primary bootstrap on to disk 'wd0':

```
installboot /dev/rwd0c /usr/mdec/bootblk
```

The secondary NetBSD/sparc64 bootstrap is located in `/usr/mdec/ofwboot`.

NetBSD/sun2 and NetBSD/sun3

Install the Berkeley Fast File System primary bootstrap on to disk 'sd0', with the secondary bootstrap '`/boot`' already present:

```
installboot /dev/rsd0c /usr/mdec/bootxx /boot
```

SEE ALSO

`uname`(3), `boot`(8), `disklabel`(8), `fdisk`(8) `pxeboot`(8)

HISTORY

This implementation of **installboot** appeared in NetBSD 1.6.

AUTHORS

The machine independent portion of this implementation of **installboot** was written by Luke Mewburn. The following people contributed to the various machine dependent back-ends: Simon Burge (pmax), Chris Demetriou (alpha), Matthew Fredette (sun2, sun3), Matthew Green (sparc64), Ross Harvey (alpha), Michael Hitch (amiga), Paul Kranenburg (sparc), David Laight (i386), Christian Limpach (next68k), Luke Mewburn (macppc), Matt Thomas (vax), Izumi Tsutsui (news68k, newsmips), and UCHIYAMA Yasushi (ews4800mips).

BUGS

There are not currently primary bootstraps to support all file systems types which are capable of being the root file system.

If a disk has been converted from **FFS** to **RAID** without the contents of the disk erased, then the original **FFS** installation may be auto-detected instead of the **RAID** installation. In this case, the **-t** *raid* option must be provided.

NetBSD/alpha

The NetBSD/alpha primary bootstrap program can only load the secondary bootstrap program from file systems starting at the beginning (block 0) of disks. Similarly, the secondary bootstrap program can only load kernels from file systems starting at the beginning of disks.

The size of primary bootstrap programs is restricted to 7.5KB, even though some file systems (e.g., ISO 9660) are able to accommodate larger ones.

NetBSD/hp300

The disk must have a boot partition large enough to hold the bootstrap code. Currently the primary bootstrap must be a LIF format file.

NetBSD/i386 and NetBSD/amd64

The bootstrap must be installed in the NetBSD partition that starts at the beginning of the mbr partition. If that is a valid filesystem and contains the `/boot` program then it will be used as the root filesystem, otherwise the 'a' partition will be booted.

The size of primary bootstrap programs is restricted to 8KB, even though some file systems (e.g., ISO 9660) are able to accommodate larger ones.

NetBSD/macppc

Due to restrictions in **installboot** and the secondary bootstrap implementation, file systems where kernels exist must start at the beginning of disks.

Currently, **installboot** doesn't recognize an existing Apple partition map on the disk and always writes a faked map to make disks bootable.

The NetBSD/macppc bootstrap program can't load kernels from **FFSv2** partitions.

NetBSD/next68k

The size of bootstrap programs is restricted to the free space before the file system at the beginning of the disk minus 8KB.

NetBSD/pmax

The NetBSD/pmax secondary bootstrap program can only load kernels from file systems starting at the beginning of disks.

The size of primary bootstrap programs is restricted to 7.5KB, even though some file systems (e.g., ISO 9660) are able to accommodate larger ones.

NetBSD/sun2 and NetBSD/sun3

The NetBSD/sun2 and NetBSD/sun3 secondary bootstrap program can only load kernels from file systems starting at the beginning of disks.

NetBSD/vax

The NetBSD/vax secondary bootstrap program can only load kernels from file systems starting at the beginning of disks.

The size of primary bootstrap programs is restricted to 7.5KB, even though some file systems (e.g., ISO 9660) are able to accommodate larger ones.

NAME

iopctl — a program to control IOP devices

SYNOPSIS

iopctl [**-f** *ctldev*] *command* [*tid*]

DESCRIPTION

The **iopctl** command can be used to interrogate and control I2O devices.

The following options are available:

-f *ctldev* Specify the control device to use. The default is `/dev/iop0`.

The following commands are available:

reconfig Reconfigure the IOP: ask all bus ports to rescan their busses, and attach or detach devices to and from the system as necessary.

showddmid *tid*
Retrieve and display the DDM (device driver module) identity parameter group from the specified target.

showdevid *tid*
Retrieve and display the device identity parameter group from the specified target.

showlct Display the driver's private copy of the logical configuration table. This copy of the LCT matches the current device configuration, but is not necessarily the latest available version of the LCT.

showstatus
Display the current status of the IOP.

showtidmap
Display the device to TID map.

FILES

`/dev/iop`*u* control device for IOP unit *u*

SEE ALSO

`ioctl`(2), `iop`(4), `iopsp`(4), `ld`(4)

The `sysutils/i2ocfg` package.

HISTORY

The **iopctl** command first appeared in NetBSD 1.5.3.

NAME

iostat — report I/O statistics

SYNOPSIS

iostat [**−CdDITx**] [**−c** *count*] [**−M** *core*] [**−N** *system*] [**−w** *wait*] [*drives*]

DESCRIPTION

iostat displays kernel I/O statistics on terminal, disk and CPU operations. By default, **iostat** displays one line of statistics averaged over the machine's run time. The use of **−c** presents successive lines averaged over the *wait* period. The **−I** option causes **iostat** to print raw, unaveraged values.

Only the last disk option specified (**−d**, **−D**, or **−x**) is used.

The options are as follows:

−c *count*	Repeat the display *count* times. Unless the **−I** flag is in effect, 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.
−C	Show CPU statistics. This is enabled by default unless the **−d**, **−D**, **−T**, or **−x** flags are used.
−d	Show disk statistics. This is the default. Displays kilobytes per transfer, number of transfers, and megabytes transferred. Use of this flag disables display of CPU and tty statistics.
−D	Show alternative disk statistics. Displays kilobytes transferred, number of transfers, and time spent in transfers. Use of this flag disables the default display.
−I	Show the running total values, rather than an average.
−M *core*	Extract values associated with the name list from the specified core instead of the default "/dev/mem".
−N *system*	Extract the name list from the specified system instead of the default "/netbsd".
−T	Show tty statistics. This is enabled by default unless the **−C**, **−d**, or **−D** flags are used.
−w *wait*	Pause *wait* seconds between each display. If no repeat *count* is specified, the default is infinity.
−x	Show extended disk statistics. Each disk is displayed on a line of its own with all available statistics. This option overrides all other display options, and all disks are displayed unless specific disks are provided as arguments. Additionally, separate read and write statistics are displayed.

iostat displays its information in the following format:

tty

tin	characters read from terminals
tout	characters written to terminals

disks Disk operations. The header of the field is the disk name and unit number. If more than four disk drives are configured in the system, **iostat** displays only the first four drives. To force **iostat** to display specific drives, their names may be supplied on the command line.

KB/t	Kilobytes transferred per disk transfer
t/s	transfers per second

MB/s	Megabytes transferred per second

The alternative display format, (selected with **-D**), presents the following values.

KB	Kilobytes transferred
xfr	Disk transfers
time	Seconds spent in disk activity

cpu

us	% of CPU time in user mode
ni	% of CPU time in user mode running niced processes
sy	% of CPU time in system mode
id	% of CPU time in idle mode

FILES

`/netbsd`	Default kernel namelist.
`/dev/mem`	Default memory file.

SEE ALSO

`fstat`(1), `netstat`(1), `nfsstat`(1), `ps`(1), `systat`(1), `vmstat`(1), `pstat`(8)

The sections starting with ‘‘Interpreting system activity’’ in *Installing and Operating 4.3BSD*.

HISTORY

iostat appeared in Version 6 AT&T UNIX. The **-x** option was added in NetBSD 1.4.

NAME

ipf − alters packet filtering lists for IP packet input and output

SYNOPSIS

ipf [**−6AcdDEInoPrsvVyzZ**] [**−l** <block|pass|nomatch>] [**−T** <optionlist>] [**−F** <i|o|a|s|S>] **−f** <*filename*> [**−f** <*filename*> [...]]

DESCRIPTION

ipf opens the filenames listed (treating "−" as stdin) and parses the file for a set of rules which are to be added or removed from the packet filter rule set.

Each rule processed by **ipf** is added to the kernel's internal lists if there are no parsing problems. Rules are added to the end of the internal lists, matching the order in which they appear when given to **ipf**.

OPTIONS

−6 This option is required to parse IPv6 rules and to have them loaded.

−A Set the list to make changes to the active list (default).

−c <language>
This option causes **ipf** to generate output files for a compiler that supports **language**. *At present, the only target language supported is* **C (-cc) for which two files - ip_rules.c and ip_rules.h are generated in the CURRENT DIRECTORY when ipf is being run. These files can be used with the IPFILTER_COMPILED kernel option to build filter rules staticly into the kernel.**

−d Turn debug mode on. Causes a hexdump of filter rules to be generated as it processes each one.

−D Disable the filter (if enabled). Not effective for loadable kernel versions.

−E Enable the filter (if disabled). Not effective for loadable kernel versions.

−F <i|o|a>
This option specifies which filter list to flush. The parameter should either be "i" (input), "o" (output) or "a" (remove all filter rules). Either a single letter or an entire word starting with the appropriate letter maybe used. This option maybe before, or after, any other with the order on the command line being that used to execute options.

−F <s|S>
To flush entries from the state table, the **-F** option is used in conjunction with either "s" (removes state information about any non-fully established connections) or "S" (deletes the entire state table). Only one of the two options may be given. A fully established connection will show up in **ipfstat -s** output as 5/5, with deviations either way indicating it is not fully established any more.

−F<5|6|7|8|9|10|11>
For the TCP states that represent the closing of a connection has begun, be it only one side or the complete connection, it is possible to flush those states directly using the number corresponding to that state. The numbers relate to the states as follows: 5 = close-wait, 6 = fin-wait-1, 7 = closing, 8 = last-ack, 9 = fin-wait-2, 10 = time-wait, 11 = closed.

−F<number>
If the argument supplied to **-F** is greater than 30, then state table entries that have been idle for more than this many seconds will be flushed.

−f <filename>
This option specifies which files **ipf** should use to get input from for modifying the packet filter rule lists.

−I Set the list to make changes to the inactive list.

−l <pass|block|nomatch>
Use of the **-l** flag toggles default logging of packets. Valid arguments to this option are **pass**, **block** and **nomatch**. When an option is set, any packet which exits filtering and matches the set category is logged. This is most useful for causing all packets which don't match any of the loaded rules to be logged.

−n This flag (no-change) prevents **ipf** from actually making any ioctl calls or doing anything which would alter the currently running kernel.

−o Force rules by default to be added/deleted to/from the output list, rather than the (default) input list.

−P Add rules as temporary entries in the authentication rule table.

−r Remove matching filter rules rather than add them to the internal lists

−s Swap the active filter list in use to be the "other" one.

−T <optionlist>
This option allows run-time changing of IPFilter kernel variables. Some variables require IPFilter to be in a disabled state (**-D**) for changing, others do not. 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. Some examples follow.

```
# Print out all IPFilter kernel tunable parameters
ipf -T list
# Display the current TCP idle timeout and then set it to 3600
ipf -D -T fr_tcpidletimeout,fr_tcpidletimeout=3600 -E
# Display current values for fr_pass and fr_chksrc, then set fr_chksrc to 1.
ipf -T fr_pass,fr_chksrc,fr_chksrc=1
```

−v Turn verbose mode on. Displays information relating to rule processing.

−V Show version information. This will display the version information compiled into the ipf binary and retrieve it from the kernel code (if running/present). If it is present in the kernel, information about its current state will be displayed (whether logging is active, default filtering, etc).

−y Manually resync the in-kernel interface list maintained by IP Filter with the current interface status list.

−z For each rule in the input file, reset the statistics for it to zero and display the statistics prior to them being zeroed.

−Z Zero global statistics held in the kernel for filtering only (this doesn’t affect fragment or state statistics).

FILES

/dev/ipauth
/dev/ipl
/dev/ipstate

SEE ALSO

ipftest(1), mkfilters(1), ipf(4), ipl(4), ipf(5), ipf.conf(5), ipf6.conf(5), ipfstat(8), ipmon(8), ipnat(8)

DIAGNOSTICS

Needs to be run as root for the packet filtering lists to actually be affected inside the kernel.

BUGS

If you find any, please send email to me at darrenr@pobox.com

NAME

ipfs − saves and restores information for NAT and state tables.

SYNOPSIS

ipfs [-nv] -l

ipfs [-nv] -u

ipfs [-nv] [**−d** <*dirname*>] -R

ipfs [-nv] [**−d** <*dirname*>] -W

ipfs [-nNSv] [**−f** <*filename*>] -r

ipfs [-nNSv] [**−f** <*filename*>] -w

ipfs [-nNSv] **−f** <*filename*> **−i** <if1>,<if2>

DESCRIPTION

ipfs allows state information created for NAT entries and rules using *keep state* to be locked (modification prevented) and then saved to disk, allowing for the system to experience a reboot, followed by the restoration of that information, resulting in connections not being interrupted.

OPTIONS

−d Change the default directory used with **−R** and **−W** options for saving state information.

−n Don't actually take any action that would affect information stored in the kernel or on disk.

−v Provides a verbose description of what's being done.

−i <ifname1>,<ifname2>
Change all instances of interface name ifname1 in the state save file to ifname2. Useful if you're restoring state information after a hardware reconfiguration or change.

−N Operate on NAT information.

−S Operate on filtering state information.

−u Unlock state tables in the kernel.

−l Lock state tables in the kernel.

−r Read information in from the specified file and load it into the kernel. This requires the state tables to have already been locked and does not change the lock once complete.

−w Write information out to the specified file and from the kernel. This requires the state tables to have already been locked and does not change the lock once complete.

−R Restores all saved state information, if any, from two files, *ipstate.ipf* and *ipnat.ipf*, stored in the */var/db/ipf* directory unless otherwise specified by the **−d** option. The state tables are locked at the beginning of this operation and unlocked once complete.

−W Saves in-kernel state information, if any, out to two files, *ipstate.ipf* and *ipnat.ipf*, stored in the */var/db/ipf* directory unless otherwise specified by the **−d** option. The state tables are locked at the beginning of this operation and unlocked once complete.

FILES

/var/db/ipf/ipstate.ipf
/var/db/ipf/ipnat.ipf
/dev/ipl
/dev/ipstate
/dev/ipnat

SEE ALSO

ipf(8), ipl(4), ipmon(8), ipnat(8)

DIAGNOSTICS

Perhaps the -W and -R operations should set the locking but rather than undo it, restore it to what it was previously. Fragment table information is currently not saved.

BUGS

If you find any, please send email to me at darrenr@pobox.com

NAME

ipfstat − reports on packet filter statistics and filter list

SYNOPSIS

ipfstat [**−6aAdfghIilnoRsv**]
ipfstat -t [**−6C**] [**−D** <addrport>] [**−P** <protocol>] [**−S** <addrport>] [**−T** <refresh time>]

DESCRIPTION

ipfstat examines /dev/kmem using the symbols **_fr_flags**, **_frstats**, **_filterin**, and **_filterout**. To run and work, it needs to be able to read both /dev/kmem and the kernel itself. The kernel name defaults to **/netbsd**.

The default behaviour of **ipfstat** is to retrieve and display the accumulated statistics which have been accumulated over time as the kernel has put packets through the filter.

OPTIONS

−6 Display filter lists and states for IPv6, if available.

−a Display the accounting filter list and show bytes counted against each rule.

−A Display packet authentication statistics.

−C This option is only valid in combination with **−t**. Display "closed" states as well in the top. Normally, a TCP connection is not displayed when it reaches the CLOSE_WAIT protocol state. With this option enabled, all state entries are displayed.

−d Produce debugging output when displaying data.

−D <addrport>
This option is only valid in combination with **−t**. Limit the state top display to show only state entries whose destination IP address and port match the addrport argument. The addrport specification is of the form ipaddress[,port]. The ipaddress and port should be either numerical or the string "any" (specifying any IP address resp. any port). If the **−D** option is not specified, it defaults to "**−D** any,any".

−f Show fragment state information (statistics) and held state information (in the kernel) if any is present.

−g Show groups currently configured (both active and inactive).

−h Show per-rule the number of times each one scores a "hit". For use in combination with **−i**.

−i Display the filter list used for the input side of the kernel IP processing.

−I Swap between retrieving "inactive"/"active" filter list details. For use in combination with **−i**.

−n Show the "rule number" for each rule as it is printed.

−o Display the filter list used for the output side of the kernel IP processing.

−P <protocol>
This option is only valid in combination with **−t**. Limit the state top display to show only state entries that match a specific protocol. The argument can be a protocol name (as defined in **/etc/protocols**) or a protocol number. If this option is not specified, state entries for any protocol are specified.

−R Don’t try to resolve addresses to hostnames and ports to services while printing statistics.

−s Show packet/flow state information (statistics only).

−sl Show held state information (in the kernel) if any is present (no statistics).

−S <addrport>
This option is only valid in combination with **−t**. Limit the state top display to show only state entries whose source IP address and port match the addrport argument. The addrport specification is of the form ipaddress[,port]. The ipaddress and port should be either numerical or the string "any" (specifying any IP address resp. any port). If the **−S** option is not specified, it defaults to "**−S** any,any".

−t Show the state table in a way similar to the way **top(1)** shows the process table. States can be sorted using a number of different ways. This option requires **curses(3)** and needs to be compiled in. It may not be available on all operating systems. See below, for more information on the keys that can be used while ipfstat is in top mode.

−T <refreshtime>
This option is only valid in combination with **−t**. Specifies how often the state top display should be updated. The refresh time is the number of seconds between an update. Any positive integer can be used. The default (and minimal update time) is 1.

−v Turn verbose mode on. Displays more debugging information. When used with either **-i** or **-o**, counters associated with the rule, such as the number of times it has been matched and the number of bytes from such packets is displayed. For "keep state" rules, a count of the number of state sessions active against the rule is also displayed.

SYNOPSIS

The role of **ipfstat** is to display current kernel statistics gathered as a result of applying the filters in place (if any) to packets going in and out of the kernel. This is the default operation when no command line parameters are present.

When supplied with either **−i** or **−o**, it will retrieve and display the appropriate list of filter rules currently installed and in use by the kernel.

One of the statistics that **ipfstat** shows is **ticks**. This number indicates how long the filter has been enabled. The number is incremented every half−second.

STATE TOP

Using the **−t** option **ipfstat** will enter the state top mode. In this mode the state table is displayed similar to the way **top** displays the process table. The **−C**, **−D**, **−P**, **−S** and **−T** command line options can be used to restrict the state entries that will be shown and to specify the frequency of display updates.

In state top mode, the following keys can be used to influence the displayed information:

b show packets/bytes from backward direction.

f show packets/bytes from forward direction. (default)

l redraw the screen.

q quit the program.

s switch between different sorting criterion.

r reverse the sorting criterion.

States can be sorted by protocol number, by number of IP packets, by number of bytes and by time-to-live of the state entry. The default is to sort by the number of bytes. States are sorted in descending order, but you can use the **r** key to sort them in ascending order.

STATE TOP LIMITATIONS

It is currently not possible to interactively change the source, destination and protocol filters or the refresh frequency. This must be done from the command line.

The screen must have at least 80 columns. This is however not checked. When running state top in IPv6 mode, the screen must be much wider to display the very long IPv6 addresses.

Only the first X-5 entries that match the sort and filter criteria are displayed (where X is the number of rows on the display. The only way to see more entries is to resize the screen.

FILES

/dev/kmem
/dev/ipl
/dev/ipstate
/netbsd

SEE ALSO

ipf(8)

BUGS

none known.

NAME

ipmon − monitors /dev/ipl for logged packets

SYNOPSIS

ipmon [**−abBDFhnpstvxX**] [**−N <device>**] [**−L <facility>**] [**−o [NSI]**] [**−O [NSI]**] [**−P <pidfile>**] [**−S <device>**] [**−f <device>**] [**<filename>**]

DESCRIPTION

ipmon opens **/dev/ipl** for reading and awaits data to be saved from the packet filter. The binary data read from the device is reprinted in human readable for, however, IP#'s are not mapped back to hostnames, nor are ports mapped back to service names. The output goes to standard output by default or a filename, if given on the command line. Should the **−s** option be used, output is instead sent to **syslogd(8)**. Messages sent via syslog have the day, month and year removed from the message, but the time (including microseconds), as recorded in the log, is still included.

Messages generated by ipmon consist of whitespace separated fields. Fields common to all messages are:

1. The date of packet receipt. This is suppressed when the message is sent to syslog.

2. The time of packet receipt. This is in the form HH:MM:SS.F, for hours, minutes seconds, and fractions of a second (which can be several digits long).

3. The name of the interface the packet was processed on, e.g., **we1**.

4. The group and rule number of the rule, e.g., **@0:17**. These can be viewed with **ipfstat -n**.

5. The action: **p** for passed, **b** for blocked, for a short packet, **n** did not match any rules or **L** for a log rule.

6. The addresses. This is actually three fields: the source address and port (separated by a comma), the **->** symbol, and the destination address and port. E.g.: **209.53.17.22,80 -> 198.73.220.17,1722**.

7. **PR** followed by the protocol name or number, e.g., **PR tcp**.

8. **len** followed by the header length and total length of the packet, e.g., **len 20 40**.

If the packet is a TCP packet, there will be an additional field starting with a hyphen followed by letters corresponding to any flags that were set. See the ipf.conf manual page for a list of letters and their flags.

If the packet is an ICMP packet, there will be two fields at the end, the first always being 'icmp', and the next being the ICMP message and submessage type, separated by a slash, e.g., **icmp 3/3** for a port unreachable message.

In order for **ipmon** to properly work, the kernel option **IPFILTER_LOG** must be turned on in your kernel. Please see **options(4)** for more details.

ipmon reopens its log file(s) and rereads its configuration file when it receives a SIGHUP signal.

OPTIONS

−a Open all of the device logfiles for reading log entries from. All entries are displayed to the same output 'device' (stderr or syslog).

−b For rules which log the body of a packet, generate hex output representing the packet contents after the headers.

−B <binarylogfilename>
Enable logging of the raw, unformatted binary data to the specified <*binarylogfilename*> file. This can be read, later, using **ipmon** with the **-f** option.

−D Cause ipmon to turn itself into a daemon. Using subshells or backgrounding of ipmon is not required to turn it into an orphan so it can run indefinitely.

−f <device>
specify an alternative device/file from which to read the log information for normal IP Filter log records.

−F Flush the current packet log buffer. The number of bytes flushed is displayed, even should the result be zero.

−L <facility>
Using this option allows you to change the default syslog facility that ipmon uses for syslog messages. The default is local0.

−n IP addresses and port numbers will be mapped, where possible, back into hostnames and service names.

−N <device>
Set the logfile to be opened for reading NAT log records from to <device>.

−o Specify which log files to actually read data from. N - NAT logfile, S - State logfile, I - normal IP Filter logfile. The **-a** option is equivalent to using **-o NSI**.

−O Specify which log files you do not wish to read from. This is most sensibly used with the **-a**. Letters available as parameters to this are the same as for **-o**.

−p Cause the port number in log messages to always be printed as a number and never attempt to look it up as from */etc/services*, etc.

−P <pidfile>
Write the pid of the ipmon process to a file. By default this is *//etc/opt/ipf/ipmon.pid* (Solaris), */var/run/ipmon.pid* (44BSD or later) or */etc/ipmon.pid* for all others.

−s Packet information read in will be sent through syslogd rather than saved to a file. The default facility when compiled and installed is **local0**. The following levels are used:

LOG_INFO − packets logged using the "log" keyword as the action rather than pass or block.

LOG_NOTICE − packets logged which are also passed

LOG_WARNING − packets logged which are also blocked

LOG_ERR − packets which have been logged and which can be considered "short".

−S <device>
Set the logfile to be opened for reading state log records from to <device>.

−t read the input file/device in a manner akin to tail(1).

−v show tcp window, ack and sequence fields.

−x show the packet data in hex.

−X show the log header record data in hex.

DIAGNOSTICS

ipmon expects data that it reads to be consistent with how it should be saved and will abort if it fails an assertion which detects an anomaly in the recorded data.

FILES

/dev/ipl
/dev/ipnat
/dev/ipstate
/etc/services

SEE ALSO

ipl(4), ipf(8), ipfstat(8), ipnat(8)

BUGS

If you find any, please send email to me at darrenr@pobox.com

NAME

ipnat − user interface to the NAT subsystem

SYNOPSIS

ipnat [**−dhlnrsvCF**] [**−M core**] [**−N system**] **−f** **<***filename***>**

DESCRIPTION

ipnat opens the filename given (treating "−" as stdin) and parses the file for a set of rules which are to be added or removed from the IP NAT.

Each rule processed by **ipnat** is added to the kernels internal lists if there are no parsing problems. Rules are added to the end of the internal lists, matching the order in which they appear when given to **ipnat**.

Note that **ipf(8)** must be enabled (with **ipf -E**) before NAT is configured, as the same kernel facilities are used for NAT functionality. In addition, packet forwarding must be enabled. These details may be handled automatically when **ipnat** is run by **rc** at normal system startup. See **options(4)**, **sysctl(8)**, and **rc.conf(5)** for more information.

OPTIONS

−C delete all entries in the current NAT rule listing (NAT rules)

−d Enable printing of some extra debugging information.

−F delete all active entries in the current NAT translation table (currently active NAT mappings)

−h Print number of hits for each MAP/Redirect filter.

−l Show the list of current NAT table entry mappings.

−n This flag (no-change) prevents **ipf** from actually making any ioctl calls or doing anything which would alter the currently running kernel.

−r Remove matching NAT rules rather than add them to the internal lists.

−s Retrieve and display NAT statistics.

−v Turn verbose mode on. Displays information relating to rule processing and active rules/table entries.

FILES

/dev/ipnat
/usr/share/examples/ipf Directory with examples.

DIAGNOSTICS

ioctl(SIOCGNATS): Input/output error Ensure that the necessary kernel functionality is present and **ipf** enabled with **ipf -E**.

SEE ALSO

ipnat(5), rc.conf(5), ipf(8), ipfstat(8)

NAME

ippool − user interface to the IPFilter pools

SYNOPSIS

ippool -a [-dnv] [-m <name>] [-o <role>] -i <ipaddr>[/<netmask>]
ippool -A [-dnv] [-m <name>] [-o <role>] [-S <seed>] [-t <type>]
ippool -f <file> [-dnuv]
ippool -F [-dv] [-o <role>] [-t <type>]
ippool -l [-dv] [-m <name>] [-t <type>]
ippool -r [-dnv] [-m <name>] [-o <role>] -i <ipaddr>[/<netmask>]
ippool -R [-dnv] [-m <name>] [-o <role>] [-t <type>]
ippool -s [-dtv] [-M <core>] [-N <namelist>]

DESCRIPTION

Ippool is used to manage information stored in the IP pools subsystem of IPFilter. Configuration file information may be parsed and loaded into the kernel, currently configured pools removed or changed as well as inspected.

The command line options used are broken into two sections: the global options and the instance specific options.

GLOBAL OPTIONS

−d Toggle debugging of processing the configuration file.

−n This flag (no-change) prevents **ippool** from actually making any ioctl calls or doing anything which would alter the currently running kernel.

−v Turn verbose mode on.

COMMAND OPTIONS

-a Add a new data node to an existing pool in the kernel.

-A Add a new (empty) pool to the kernel.

-f <file>
Read in IP pool configuration information from the file and load it into the kernel.

-F Flush loaded pools from the kernel.

-l Display a list of pools currently loaded into the kernel.

-r Remove an existing data node from a pool in the kernel.

-R Remove an existing pool from within the kernel.

-s Display IP pool statistical information.

OPTIONS

-i <ipaddr>[/<netmask>]
Sets the IP address for the operation being undertaken with an all-one’s mask or, optionally, a specific netmask given in either the dotted-quad notation or a single integer.

-m <name>
Sets the pool name for the current operation.

-M <core>
Specify an alternative path to /dev/kmem to retrieve statistical information from.

-N <namelist>
Specify an alternative path to lookup symbol name information from when retrieving statistical information.

-o <role>
Sets the role with which this pool is to be used. Currently only **ipf, auth** and **count** are accepted as arguments to this option.

-S <seed>
Sets the hashing seed to the number specified. Only for use with **hash** type pools.

-t <type>
Sets the type of pool being defined. Myst be one of **tree, hash, group-map.**

-u When parsing a configuration file, rather than load new pool data into the kernel, unload it.

FILES

/dev/iplookup
/etc/ippool.conf

SEE ALSO

ippool(5), ipf(8), ipfstat(8)

NAME

iprop-log — maintain the iprop log file

SYNOPSIS

iprop-log [**--version**] [**-h** | **--help**] *command*

iprop-log truncate [**-c** *file* | **--config-file=***file*] [**-r** *string* | **--realm=***string*] [**-h** | **--help**]

iprop-log dump [**-c** *file* | **--config-file=***file*] [**-r** *string* | **--realm=***string*] [**-h** | **--help**]

iprop-log replay [**--start-version=***version-number*] [**--end-version=***version-number*] [**-c** *file* | **--config-file=***file*] [**-r** *string* | **--realm=***string*] [**-h** | **--help**]

DESCRIPTION

Supported options:

--version

-h, **--help**

command can be one of the following:

truncate

-c *file*, **--config-file=***file*
configuration file

-r *string*, **--realm=***string*
realm

Truncates the log. Sets the new logs version number for the to the last entry of the old log. If the log is truncted by emptying the file, the log will start over at the first version (0).

dump

-c *file*, **--config-file=***file*
configuration file

-r *string*, **--realm=***string*
realm

Print out all entires in the log to standard output.

replay

--start-version=*version-number*
start replay with this version

--end-version=*version-number*
end replay with this version

-c *file*, **--config-file=***file*
configuration file

-r *string*, **--realm=***string*
realm

Replay the changes from specified entries (or all if none is specified) in the transaction log to the database.

last-version

−c *file*, **−-config-file=***file*
configuration file

−r *string*, **−-realm=***string*
realm

prints the version of the last log entry.

SEE ALSO

iprop(8)

NAME

iprop, **ipropd-master**, **ipropd-slave** — propagate changes to a Heimdal Kerberos master KDC to slave KDCs

SYNOPSIS

ipropd-master [**−c** *string* | **−-config-file=***string*] [**−r** *string* | **−-realm=***string*] [**−k** *kspec* | **−-keytab=***kspec*] [**−d** *file* | **−-database=***file*] [**−-slave-stats-file=***file*] [**−-time-missing=***time*] [**−-time-gone=***time*] [**−-detach**] [**−-version**] [**−-help**]

ipropd-slave [**−c** *string* | **−-config-file=***string*] [**−r** *string* | **−-realm=***string*] [**−k** *kspec* | **−-keytab=***kspec*] [**−-time-lost=***time*] [**−-detach**] [**−-version**] [**−-help**] *master*

DESCRIPTION

ipropd-master is used to propagate changes to a Heimdal Kerberos database from the master Kerberos server on which it runs to slave Kerberos servers running **ipropd-slave**.

The slaves are specified by the contents of the slaves file in the KDC's database directory, e.g. /var/heimdal/slaves. This has principals one per-line of the form

iprop/*slave@REALM*

where *slave* is the hostname of the slave server in the given *REALM*, e.g.

iprop/kerberos-1.example.com@EXAMPLE.COM

On a slave, the argument *master* specifies the hostname of the master server from which to receive updates.

In contrast to hprop(8), which sends the whole database to the slaves regularly, **iprop** normally sends only the changes as they happen on the master. The master keeps track of all the changes by assigning a version number to every change to the database. The slaves know which was the latest version they saw, and in this way it can be determined if they are in sync or not. A log of all the changes is kept on the master. When a slave is at an older version than the oldest one in the log, the whole database has to be sent.

The changes are propagated over a secure channel (on port 2121 by default). This should normally be defined as "iprop/tcp" in /etc/services or another source of the services database. The master and slaves must each have access to a keytab with keys for the **iprop** service principal on the local host.

There is a keep-alive feature logged in the master's slave-stats file (e.g. /var/heimdal/slave-stats).

Supported options for **ipropd-master**:

−c *string*, **−-config-file=***string*

−r *string*, **−-realm=***string*

−k *kspec*, **−-keytab=***kspec*
 keytab to get authentication from

−d *file*, **−-database=***file*
 Database (default per KDC)

−-slave-stats-file=*file*
 file for slave status information

−-time-missing=*time*
 time before slave is polled for presence (default 2 min)

--time-gone=*time*
  time of inactivity after which a slave is considered gone (default 5 min)

--detach
  detach from console

--version

--help

Supported options for **ipropd-slave**:

-c *string*, **--config-file=***string*

-r *string*, **--realm=***string*

-k *kspec*, **--keytab=***kspec*
  keytab to get authentication from

--time-lost=*time*
  time before server is considered lost (default 5 min)

--detach
  detach from console

--version

--help

Time arguments for the relevant options above may be specified in forms like 5 min, 300 s, or simply a number of seconds.

FILES

`slaves`, `slave-stats` in the database directory.

SEE ALSO

`hpropd`(8), `hprop`(8), `krb5.conf`(8), `kdc`(8), `iprop-log`(8).

NAME

ipscan − user interface to the IPFilter content scanning

SYNOPSIS

ipscan [**−dlnrsv**] [ ] **−f** <*filename*>

DESCRIPTION

ipscan opens the filename given (treating "−" as stdin) and parses the file to build up a content scanning configuration to load into the kernel. Currently only the first 16 bytes of a connection can be compared.

OPTIONS

−d Toggle debugging of processing the configuration file.

−l Show the list of currently configured content scanning entries.

−n This flag (no-change) prevents **ipscan** from actually making any ioctl calls or doing anything which would alter the currently running kernel.

−r Remove commands from kernel configuration as they are read from the configuration file rather than adding new ones.

−s Retrieve and display content scanning statistics

−v Turn verbose mode on.

FILES

/dev/ipscan /etc/ipscan.conf

SEE ALSO

ipscan(5), ipf(8)

NAME

ipwctl — configure Intel PRO/Wireless 2100 network adapter

SYNOPSIS

ipwctl [**-i**] *iface*
ipwctl [**-i**] *iface* **-r**

DESCRIPTION

The **ipwctl** utility controls the operation of Intel PRO/Wireless 2100 networking devices via ipw(4) driver.

You should not use this program to configure IEEE 802.11 parameters. Use ifconfig(8) instead.

OPTIONS

The options are as follows:

-i *iface*
Display adapter's internal statistics.

-i *iface* **-r**
Display the radio switch state (on or off).

FILES

The firmware, loaded automatically, is not shipped with NetBSD; install the pkgsrc/sysutils/ipw-firmware package. The original archive is available from Intel at: http://ipw2100.sourceforge.net/firmware.php?fid=3.

ipw2100-1.2.fw	BSS mode (connection to an access point) firmware
ipw2100-1.2-i.fw	IBSS mode (point-to-point connection) firmware
ipw2100-1.2-p.fw	Monitor mode firmware

SEE ALSO

ipw(4), ifconfig(8), pkgsrc/sysutils/ipw-firmware

AUTHORS

The **ipwctl** utility and this man page were written by Damien Bergamini ⟨damien.bergamini@free.fr⟩.

NAME

irdaattach — attach serial lines to IrDA frame driver

SYNOPSIS

irdaattach [**−d** *dongle*] [**−fHhlmnp**] *ttyname*

DESCRIPTION

irdaattach is used to assign a tty line to an IrDA frame level driver. The following operands are supported by **irdaattach**:

−d *dongle* Sets the dongle type. The following dongles are supported:

none	No dongle
tekram	Tekram IR-210B
jeteye	Extended Systems JetEye
actisys	ACTiSYS IR-220L
actisys+	ACTiSYS IR-220L+
litelink	Parallax LiteLink
girbil	Greenwich GIrBIL

The default is `none`.

−f Print the name of the IrDA frame device that should be used to access the frames.

−H Turn on DTR/CTS flow control. By default, no flow control is done.

−h Turn on RTS/CTS flow control. By default, no flow control is done.

−l Turn on the CLOCAL flag, making it possible to run SLIP on a cable without modem control signals (e.g. DTR, DSR, DCD).

−m Maintain modem control signals after closing the line. Specifically, this disables HUPCL.

−n Do not detach from invoking tty.

−p Print process id to file.

ttyname Specifies the name of the tty device. *Ttyname* should be a string of the form `ttyXX`, or `/dev/ttyXX`.

Only the super-user may attach a network interface.

The frame driver is detached by killing the **irdaattach** process.

EXAMPLES

```
      irdaattach tty00
      ircomm −Y −d ‘irdaattach −p −f /dev/tty02‘
```

DIAGNOSTICS

Messages indicating that the specified interface does not exist, the requested address is unknown, or that the user is not privileged but tried to alter an interface's configuration.

SEE ALSO

`daemon`(3), `irframe`(4), `irframetty`(4), `slattach`(8)

HISTORY

The **irdaattach** command appeared in NetBSD 1.6.

NAME

iscsi-harness — test harness for iscsi-target

SYNOPSIS

iscsi-harness [**−V**] [**−a** *authentication type*] [**−d** *digest type*]
[**−h** *target hostname*] [**−n** *number of test iterations*]
[**−p** *port number of iSCSI target*] [**−t** *target name*]
[**−u** *user name*]

DESCRIPTION

iscsi-harness is the test harness for iscsi-target(8), the iSCSI service. **iscsi-harness** acts as an initiator, performing a set of tests for a number of iterations. to show network-related information, It connects to the iscsi-target(8) service, and performs a discovery session, then sends data of various sizes to the target, and reads data of various sizes from the target.

Options and operands available for **iscsi-harness**:

−a *authentication type*
The authentication type can be one of *chap*, *Kerberos*, *SRP*, or *none*. The default value is *none*.

−d *digest type*
The digest type can be one of *header*, *data*, *both*, or *all*, or *none*. The default value is *none*.

−p *iSCSI target port number*
This option allows the **iscsi-harness** utility to connect to a specific port number upon which the iscsi-target(8) process is listening. The default port number is 3260.

−V **iscsi-harness** will print the utility name and version number, and the address for bug reports, and then exit.

FILES

/etc/iscsi/targets	the list of exported storage
/var/run/iscsi-target.pid	the PID of the currently running **iscsi-harness**

SEE ALSO

targets(5), iscsi-target(8)

HISTORY

The **iscsi-harness** utility first appeared in NetBSD 4.0.

NAME

iscsi-target — service remote iSCSI requests

SYNOPSIS

iscsi-target [**-46DV**] [**-b** *block length*] [**-f** *configuration file*]
[**-p** *port number*] [**-s** *maximum number of sessions*]
[**-t** *target name*] [**-v** *verbose arg*]

DESCRIPTION

iscsi-target is the server for iSCSI requests from iSCSI initiators. **iscsi-target** listens for discovery and login requests on the required port, and responds to those requests appropriately.

Options and operands available for **iscsi-target**:

-4 **iscsi-target** will listen for IPv4 connections, and respond back using IPv4. This is the default address family.

-6 **iscsi-target** will listen for IPv6 connections, and respond back using IPv6.

-b *blocksize*
Specify the underlying block size for iSCSI storage which will be served. The possible sizes are: 512, 1024, 2048, and 4096 bytes, with the default being 512 bytes.

-D When this option is specified, **iscsi-target** will not detach itself from the controlling tty, and will not become a daemon. This can be useful for debugging purposes.

-f *configfile*
Use the named file as the configuration file. The default file can be found in */etc/iscsi/targets*. See targets(5) for more information.

-p *port number*
Use the port number provided as the argument as the port on which to listen for iSCSI service requests from initiators.

-s *maximum number of sessions*
Allow the maximum number of sessions to be initiated when connecting to the target.

-t *filename*
The target name (as it appears to the iSCSI initiator) can be specified using this flag.

-V **iscsi-target** will print the utility name and version number, and the address for bug reports, and then exit.

-v *argument*
The amount of information shown can be varied by using this command. Possible values of *argument* are *net* to show network-related information, *iscsi* to show iSCSI protocol-related information, *scsi* to show SCSI protocol information, and *all* to show information from all of the above arguments.

FILES

/etc/iscsi/targets	the list of exported storage
/var/run/iscsi-target.pid	the PID of the currently running **iscsi-target**

SEE ALSO

targets(5)

HISTORY

The **iscsi-target** utility first appeared in NetBSD 4.0.

NAME

iscsifs — refuse-based iSCSI initiator

SYNOPSIS

iscsifs [**−46bcfvV**] [**−a** *authentication-type*] [**−d** *digest-type*]
[**−h** *target-hostname*] [**−p** *target-port-number*] [**−t** *target-number*]
[**−u** *username*] *mount_point*

DESCRIPTION

The **iscsifs** utility can be used to access an iSCSI target, such as iscsi-target(8), to access block storage which has been exported. The **iscsifs** utility makes use of the virtdir(3) virtual directory routines. Information pertaining to the target is displayed underneath the mount point, along with the device corresponding to the storage which the target exports.

The various arguments are as follows:

−4 Use an IPv4 connection to the target

−6 Use an IPv6 connection to the target

−a *authentication-type*
Use the specified authentication type when communicating with the target. The possible values are chap, kerberos, srp or none. The default value is none.

−b Show the storage as a block device

−c Show the storage as a character device

−d *digest-type*
Use the specified digest type when communicating with the target. The possible values are header, data, both, all or none. The default value is none.

−f Show the storage as a regular file

−h *hostname*
Connect to the iSCSI target running on the host specified as the argument.

−p *port-number*
Connect to the iSCSI target running on the port specified as the argument. The default value is 3260.

−t *target*
Connect to the number of the iSCSI target running as the argument.

−u *username*
Use the specified user's credentials when logging in to the iSCSI target. There is no default.

−v Be verbose in operation

−V Print out the version number and then exit.

The refuse(3) library is used to provide the file system features.

The mandatory parameter is the local mount point.

This iSCSI initiator presents a view of the targets underneath the mount point. Firstly, it creates a directory tree with the hostname of the target, and, in that directory, a virtual directory is created for each target name exported by the iSCSI target program. Within that virtual target directory, symbolic links exist for the hostname (for convenience), a textual representation of the IP address, the iSCSI target product name, the iSCSI target IQN, the iSCSI target vendor and version number. One other directory entry is presented in the virtual target directory, relating to the storage presented by the iSCSI target. This can be in the form of a regular file, which is also the default, a block device or a character device.

SEE ALSO

`librefuse`(3), `puffs`(3), `virtdir`(3), `iscsi-target`(8).

HISTORY

The **iscsifs** utility first appeared in NetBSD 5.0.

AUTHORS

The **iscsifs** utility was written by Alistair Crooks ⟨agc@NetBSD.org⟩.

NAME

isdnd — isdn4bsd ISDN connection management daemon

SYNOPSIS

isdnd [**-c** *configfile*] [**-d** *debuglevel*] [**-f**] [**-F**] [**-l**] [**-L** *logfile*] [**-P**] [**-r** *device*] [**-s** *facility*] [**-t** *terminaltype*] [**-u** *charging unit length*] [**-m**]

DESCRIPTION

isdnd is the isdn4bsd package demon which manages all ISDN related connection and disconnection of ISDN devices supported by the package.

The options are as follows:

-c Use *configfile* as the name of the runtime configuration filename for **isdnd** instead of the default file `/etc/isdn/isdnd.rc`.

-d If debugging support is compiled into **isdnd** this option is used to specify the debugging level, or better which kind of debugging messages are displayed. The debugging level is the sum of the following values:

0x001	general debugging.
0x002	rates calculation.
0x004	timing calculations.
0x008	state transitions.
0x010	retry handling.
0x020	dialing.
0x040	process handling.
0x080	isdn4bsd kernel i/o calls.
0x100	controller and channel busy/free messages.
0x200	isdnd.rc configuration file processing.
0x400	outgoing call budget handling.

The value can be specified in any number base supported by the `sscanf`(3) library routine.

In addition, this option accepts also the character ‘n’ as an argument to disable displaying debug messages on the full-screen display.

-f Specifying this option causes **isdnd** to enter the full-screen mode of operation. When operating in this mode, entering the control character *Control-L* causes the display to be refreshed and entering *Carriage-Return* or *Enter* will pop-up a command window. Because the **isdnd** daemon will not listen to messages while the command window is active, this command window will disappear automatically after 5 seconds without any command key press.

While the command window is active, *Tab* or *Space* advances to the next menu item. To execute a command, press *Return* or *Enter* for the highlighted menu item, or enter the number corresponding to the item to be executed or enter the capitalized character in the menu item description.

-l If this option is set, logging is not done via the `syslogd`(8) facility but instead is appended to a file.

-L Specifies the name of the logfile which is used when the option **-l** is set. See also the keyword *rotatesuffix* in the system section of `isdnd.rc`(5).

-P This option prints out the parsed and verified isdnd configuration in the same format as the isdnd.rc file. This output can be used as an isdnd.rc file. This feature is especially useful when debugging an isdnd.rc file to see what the default settings of options are when they are not set in the isdnd.rc input file.

The **isdnd** exits after the printout is done.

−F This option prevents **isdnd** to detach from the controlling tty and become a daemon.

−r In conjunction with the **−t** option, *device* specifies a terminal device which becomes the controlling tty for **isdnd** and on which the full-screen mode output is displayed.

−s This option may be used to specify the logging facility in case `syslog`(3) logging is configured and another facility than the default LOCAL0 facility shall be used. The facility is to be specified as an integer in the range 0-11 or 16-23 (see the file `/usr/include/syslog.h`).

−t In conjunction with the **−f** and **−r** options, *terminaltype* specifies a terminal type or termcap entry name (such as vt220) for the device used for **isdnd** full-screen output. This is useful if an unused (no getty running) tty line is used for full-screen output for which no `TERM` environment variable exists.

−u Specifies the length of a charging unit in case the config file entry keyword *unitlengthsrc* is set to *cmdl*.

−m If the ISDN daemon is compiled with local or remote monitoring support, this option disables all monitoring access. It overrides the config file option *monitor-allowed*.

INTERACTION WITH THE KERNEL

isdnd communicates with the kernel part of isdn4bsd by receiving status and event messages (via `read`(2) from device `/dev/isdn`) and by transmitting commands and responses (via `ioctl`(2) on device `/dev/isdn`).

The messages and message parameters are documented in the include file */usr/include/machine/i4b_ioctl.h*.

Supported command and response messages (ioctl's) to the kernel are:

I4B_CDID_REQ
: Request a unique Call Description IDentifier (cdid) which identifies uniquely a single interaction of the local D channel with the exchange.

I4B_CONNECT_REQ
: Actively request a call setup to a remote ISDN subscriber.

I4B_CONNECT_RESP
: Respond to an incoming call, either accept, reject or ignore it.

I4B_DISCONNECT_REQ
: Actively terminate a connection.

I4B_CTRL_INFO_REQ
: Request information about an installed ISDN controller card.

I4B_DIALOUT_RESP
: Give information about call setup to driver who requested dialing out.

I4B_TIMEOUT_UPD
: Update the kernels timeout value(s) in case of dynamically calculated shorthold mode timing changes.

I4B_UPDOWN_IND
: Inform the kernel userland drivers about interface soft up/down status changes.

I4B_CTRL_DOWNLOAD
: Download firmware to active card(s).

I4B_ACTIVE_DIAGNOSTIC
: Return diagnostic information from active cards.

Supported status and event messages from the kernel are:

MSG_CONNECT_IND
: An incoming call from a remote ISDN user is indicated.

MSG_CONNECT_ACTIVE_IND
: After an incoming call has been accepted locally or an outgoing call has been accepted by a remote, the exchange signaled an active connection and the corresponding B-channel is switched through.

MSG_DISCONNECT_IND
: A call was terminated.

MSG_DIALOUT_IND
: A userland interface driver requests the daemon to dial out (typically a network interface when a packet arrives in its send queue).

MSG_IDLE_TIMEOUT_IND
: A call was terminated by the isdn4bsd kernel driver because a B-channel idle timeout occurred.

MSG_ACCT_IND
: Accounting information from a network driver.

MSG_CHARGING_IND
: Charging information from the kernel.

OUTGOING CALLS

Currently the only possibility to trigger an outgoing call is that an isdn4bsd network driver (*isdn<n>*) sends a *MSG_DIALOUT_IND* to the **isdnd** daemon.

The daemon requests a new CDID from the kernel by using the *I4B_CDID_REQ* ioctl message, this CDID is now used in all interactions with the kernel to identify this single call until a disconnect occurs.

After getting the CDID, the daemon looks up several additional information in its entry section of the configuration corresponding to that connection and issues a *I4B_CONNECT_REQ* ioctl message to the kernel. The kernel now dials the remote side and if the remote side accepts the call, the kernel sends a *MSG_CONNECT_ACTIVE_IND* to the daemon.

The call is terminated by either the local site timing out or the remote side hanging up the connection or the local side actively sending a *I4B_DISCONNECT_REQ* ioctl message, both events are signaled to the **isdnd** by the kernel sending the *I4B_DISCONNECT_IND* message and the CDID corresponding to the call is no longer valid.

INCOMING CALLS

Incoming calls are signaled to **isdnd** by the kernel transmitting the *MSG_CONNECT_IND* message to the daemon.

With the information contained in this message, **isdnd** searches the entry section of its configuration database and if a match is found, it accepts or rejects the call or, if no match is found, it ignores the call - all by issuing a *I4B_CONNECT_RESP* ioctl message with the appropriate parameters to the kernel.

In case the daemon decided to accept the call, the kernel signals this by sending a *MSG_CONNECT_ACTIVE_IND* message to the daemon.

The call is terminated by either the local site timing out or the remote side hanging up the connection or the local side actively sending a *I4B_DISCONNECT_REQ* ioctl message, both events are signaled to **isdnd** by the kernel sending the *I4B_DISCONNECT_IND* message and the CDID corresponding to the call is no longer valid.

SIGNALS

Sending a HUP signal to **isdnd** causes all open connections to be terminated and the configuration file is reread. In case aliasfile handling was enabled, the aliasfile is also reread.

Sending a USR1 signal to **isdnd** causes the accounting file and the logfile (if logging to a file is used instead of logging via the `syslog`(3) facility) to be closed and reopened to make logfile rotation possible.

ENVIRONMENT

The following environment variables affect the execution of **isdnd**:

`TERM`	The terminal type when running in full-screen display mode. See `environ`(7) for more information.

FILES

`/dev/isdn`	The device-file used to communicate with the kernel ISDN driver subsystem.
`/var/log/messages`	A record of the actions in case of syslogd logging support.
`/var/log/isdnd.acct`	The default accounting information filename (if accounting is configured).
`/var/log/isdnd.log`	The default logging filename (if logging to a file is configured).
`/var/run/isdnd.pid`	The process id of the ISDN daemon (also known as "lockfile" to isdnd, preventing multiple invocations of it).
`/etc/isdn`	The directory where isdnd expects some supplementary data files and programs for telephone answering support.
`/etc/isdn/isdnd.rc`	The default runtime configuration file.
`/etc/isdn/isdnd.rates`	The default unit charging rates specification file.
`/etc/isdn/isdntel.alias`	The default table (if aliasing is enabled) to convert phone number to caller's name.

EXAMPLES

For a first try, the following command should be used to start **isdnd** in foreground mode for better debugging the configuration setup:

```
isdnd -d0xf9 -F
```

This will start isdnd with reasonable debugging settings and produce output on the current terminal. **isdnd** can then be terminated by entering *Control-C*.

Another example, the command:

```
isdnd -d0xf9 -f -r /dev/ttyv3 -t vt100
```

will start **isdnd** with reasonable debugging messages enabled, full-screen mode of operation, full-screen display redirected to /dev/ttyv03 and using a termcap entry for vt100 on this display.

DIAGNOSTICS

Exit status is 0 on success, 1 on error.

SEE ALSO

`ippp`(4), `irip`(4), `isdnd.rates`(5), `isdnd.rc`(5), `isdntel`(8), `isdntrace`(8), `syslogd`(8)

AUTHORS

The **isdnd** daemon and this manual page were written by Hellmuth Michaelis ⟨hm@kts.org⟩.

NAME

isdnmonitor — isdn4bsd / isdnd remote monitoring tool

SYNOPSIS

isdnmonitor [**−c**] [**−d** *debuglevel*] [**−f** *filename*] [**−h** *hostspec*] [**−l** *pathname*] [**−p** *portspec*]

DESCRIPTION

isdnmonitor is used to remotely monitor the operation of the ISDN demon, isdnd(8), which manages all ISDN related connection and disconnection of ISDN devices supported by the isdn4bsd package.

The options are as follows:

−c Switch to (curses-) fullscreen mode of operation. In this mode, **isdnmonitor** behaves nearly exactly as isdnd(8) in fullscreen mode. In fullscreen mode, entering the control character *Control-L* causes the display to be refreshed and entering *Carriage-Return* or *Enter* will pop-up a command window. Because **isdnmonitor** will not listen to messages while the command window is active, this command window will disappear automatically after 5 seconds without any command key press.

While the command window is active, *Tab* or *Space* advances to the next menu item. To execute a command, press *Return* or *Enter* for the highlighted menu item, or enter the number corresponding to the item to be executed or enter the capitalized character in the menu item description.

−d If debugging support is compiled into **isdnmonitor** this option is used to specify the debugging level.

In addition, this option accepts also the character 'n' as an argument to disable displaying debug messages on the full-screen display.

−f Specifying this option causes **isdnmonitor** to write its normal output and - if enabled - debugging output to a file which name is specified as the argument.

−l is used to specify a Unix local domain socket name to be used for communication between isdnd(8) and **isdnmonitor**

−h is used to specify a hostname or a dotted-quad IP address of a machine where an isdnd(8) is running which should be monitored.

−p This option may be used to specify a remote port number in conjunction with the **−h** option.

ENVIRONMENT

The following environment variables affect the execution of **isdnmonitor**:

TERM The terminal type when running in full-screen display mode. See environ(7) for more information.

EXAMPLES

For a first try, the following command should be used to start **isdnmonitor** to monitor a locally running isdnd:

```
isdnmonitor -h localhost
```

DIAGNOSTICS

Exit status is 0 on success, 1 on error.

SEE ALSO

`isdnd`(8)

AUTHORS

The **`isdnmonitor`** utility was written by Martin Husemann and Hellmuth Michaelis. This manual page was written by Hellmuth Michaelis ⟨hm@kts.org⟩.

NAME

isdntel — isdn4bsd telephone answering management utility

SYNOPSIS

isdntel [**-a** *aliasfile*] [**-d** *spooldir*] [**-p** *playcommand*] [**-t** *timeout*]

DESCRIPTION

isdntel is used to provide an "answering machine" functionality for incoming telephone voice messages.

The following options are supported:

-a Use *aliasfile* as the pathname for an aliasfile containing aliases for phone numbers. The default path is `/etc/isdn/isdntel.alias`. The format of an alias entry is the number string followed by one or more spaces or tabs. The rest of the line is taken as the alias string. Comments are introduced by a leading blank, tab or "#" character.

-d Use *spooldir* as the directory where the incoming voice messages are stored by the "answ" script called by isdnd(8). This defaults to the directory `/var/isdn`. The format of a voice message filename is:

```
YYMMDDhhmmss-dest_number-source_number-length_in_secs
```

-p Use *playcommand* as the command string to execute for playing a voice message to some audio output facility. The characters *%s* are replaced by the currently selected filename. The default string is *cat %s | alaw2ulaw >/dev/audio*

-t The value for *timeout* specifies the time in seconds the program rereads the spool directory when there is no keyboard activity.

The screen output should be obvious. If in doubt, consult the source.

SEE ALSO

isdntel(4), isdnd.rc(5), isdnd(8)

AUTHORS

The **isdntel** utility and this manual page were written by Hellmuth Michaelis ⟨hm@kts.org⟩.

NAME

isdntelctl — control isdn4bsd telephone sound format conversion

SYNOPSIS

isdntelctl [**-c**] [**-g**] [**-u** *unit*] [**-A**] [**-U**] [**-N**]

DESCRIPTION

isdntelctl is part of the isdn4bsd package and is used to configure the sound format conversion facilities of the /dev/isdntel interfaces.

The following options are available:

-c Clear the telephone input queue.

-g Get the sound format currently in use.

-u Set the `/dev/isdntel` unit number. The default value is zero to access device `/dev/isdntel0`.

-A Do A-law (ISDN line) -> mu-law (userland) conversion.

-U Do mu-law (ISDN line) -> A-law (userland) conversion.

-N Set sound conversion to do no format conversion.

The telephony data stream comes out of the line in a bit-reversed format, so the `isdntel`(4) driver does the bit-reversion process in any case.

Additionally, the user can specify to do A-law to mu-law, mu-law to A-law or no conversion at all in the isdntel driver by using the **isdntelctl** utility.

FILES

`/dev/isdntel<n>`

EXAMPLES

The command

```
isdntelctl -g
```

displays the currently used sound format for device `/dev/isdntel0`.

SEE ALSO

`isdntel`(4), `isdnd.rc`(5), `isdnd`(8)

STANDARDS

A-law and mu-law are specified in ITU Recommendation G.711.

AUTHORS

The **isdntelctl** utility and this man page were written by Hellmuth Michaelis ⟨hm@kts.org⟩.

NAME

isdntrace — isdn4bsd ISDN protocol trace utility

SYNOPSIS

isdntrace [**−a**] [**−b**] [**−d**] [**−f** *filename*] [**−h**] [**−i**] [**−l**] [**−n** *number*] [**−o**] [**−p** *filename*] [**−r**] [**−u** *number*] [**−x**] [**−B**] [**−F**] [**−P**] [**−R** *unit*] [**−T** *unit*]

DESCRIPTION

isdntrace is part of the isdn4bsd package and is used to provide the user with a mnemonic display of the layers 1, 2 and 3 protocol activities on the D channel and hex dump of the B channel(s) activities.

Together with two passive supported cards and an easy to build cable it can also be used to monitor the complete traffic on a S0 bus providing S0 bus analyzer features.

The **isdntrace** utility is only available for passive supported cards.

Note
All filenames, user specified or default, get a date and time stamp string added in the form -yymmdd-hhmmss: a hyphen, two digits year, month, day, a hyphen and two digits hour, minutes and seconds. Trace files no longer get overwritten. In case a new filename is needed within a second, the filename-generating mechanism sleeps one second.
In case the program is sent a USR1 signal, a new user specified or default filename with a new date and time stamp is generated and opened.

The following options can be used:

−a Run **isdntrace** in analyzer mode by using two passive cards and a custom cable which can be build as described in the file *cable.txt* in the isdn4bsd source distribution. One card acts as a receiver for the transmitting direction on the S0 bus while the other card acts as a receiver for the receiving direction on the S0 bus. Complete traffic monitoring is possible using this setup.

−b switch B channel tracing on (default off).

−d switch D channel tracing off (default on).

−f Use *filename* as the name of a file into which to write tracing output (default filename is isdntrace<n> where n is the number of the unit to trace).

−h switch display of header off (default on).

−i print layer 1 (I.430) INFO signals to monitor layer 1 activity (default off).

−l switch displaying of Layer 2 (Q.921) frames off (default on).

−n This option takes a numeric argument specifying the minimum frame size in octets a frame must have to be displayed. (default 0)

−o switch off writing trace output to a file (default on).

−p Use *filename* as the name of a file used for the -B and -P options (default filename is isdntracebin<n> where n is the number of the unit to trace).

−r Switch off printing a raw hexadecimal dump of the packets preceding the decoded protocol information (default on).

−u Use *number* as the unit number of the controller card to trace (default 0).

-x Switch on printing of packets with a non-Q.931 protocol discriminator. (default off).

-B Write undecoded binary trace data to a file for later or remote analyzing (default off).

-F This option can only be used when option -P (playback from binary data file) is used. The -F option causes playback not to stop at end of file but rather to wait for additional data to be available from the input file.

This option is useful when trace data is accumulated in binary format (to save disk space) but a monitoring functionality is desired. (default off).

-P Read undecoded binary trace data from file instead from device (default off).

-R Use *unit* as the receiving interface unit number in analyze mode.

-T Use *unit* as the transmitting interface unit number in analyze mode.

When the USR1 signal is sent to a **isdntrace** process, the currently used logfiles are reopened, so that logfile rotation becomes possible.

The trace output should be obvious. It is very handy to have the following standard texts available when tracing ISDN protocols:

I.430	ISDN BRI layer 1 protocol description.
Q.921	ISDN D-channel layer 2 protocol description.
Q.931	ISDN D-channel layer 3 protocol description.
1TR6	German-specific ISDN layer 3 protocol description. (NOTICE: decoding of the 1TR6 protocol is included but not supported since i dont have any longer access to a 1TR6 based ISDN installation.)

isdntrace automatically detects the layer 3 protocol being used by looking at the Protocol Discriminator (see: Q.931/1993 pp. 53).

FILES

/dev/isdntrc<n>
The device file(s) used to get the trace messages for ISDN card unit <n> out of the kernel.

EXAMPLES

The command:

```
isdntrace -f /var/tmp/isdn.trace
```

will start D channel tracing on passive controller 0 with all except B channel tracing enabled and logs everything into the output file /var/tmp/isdn.trace-yymmdd-hhmmss (where yymmdd and hhmmss are replaced by the current date and time values).

SEE ALSO

isdnd(8)

STANDARDS

ITU Recommendations I.430, Q.920, Q.921, Q.930, Q.931

FTZ Richtlinie 1TR3, Band III

ITU Recommendation Q.932 (03/93), Q.950 (03/93)

ETSI Recommendation ETS 300 179 (10/92), ETS 300 180 (10/92)

ETSI Recommendation ETS 300 181 (04/93), ETS 300 182 (04/93)

ITU Recommendation X.208, X.209

AUTHORS

The **isdntrace** utility was written by Gary Jennejohn and Hellmuth Michaelis.

This manual page was written by Hellmuth Michaelis ⟨hm@kts.org⟩.

NAME

iteconfig — modify console attributes at run time

SYNOPSIS

iteconfig [**-i**] [**-f** *file*] [**-v** *volume*] [**-p** *pitch*] [**-t** *msec*] [**-w** *width*]
[**-h** *height*] [**-d** *depth*] [**-x** *offset*] [**-y** *offset*] [*color ...*]

DESCRIPTION

iteconfig is used to modify or examine the attributes of the console bell and bitmapped console display. The console bell's volume, pitch, and count may be specified, as well as the bitmapped display's width, height, horizontal and vertical offset, pixel depth, and color map.

The following flags are interpreted by **iteconfig**:

-i	After processing all other arguments, print information about the console's state.
-f	Open and use the terminal named by *file* rather than the default console `/dev/ttye0`.
-v	Set the volume of the console bell to *volume*, which must be between 0 and 63, inclusive.
-p	Set the pitch of the console bell to *pitch*, which must be between 10 and 1399.
-t	Set the duration of the beep to *msec* milliseconds which must be between 1 and 5000 (5 seconds).
-w	Set the width of the console display to *width* pixel columns. *Width* must be a positive integer.
-h	Set the height of the console display to *height* pixel rows. *Height* must be a positive integer.
-d	Set the number of bitplanes the console view should use to *depth*. For example, if *depth* is 3 then 8 colors will be used.
-x	Set the horizontal offset of the console view on the monitor to *offset* pixel columns. The horizontal offset may be a positive or a negative integer, positive being an offset to the right, negative to the left.
-y	Set the vertical offset of the console view on the monitor to *offset* pixel rows. The vertical offset may be a positive or a negative integer, positive being an offset down, negative up.

Any additional arguments will be interpreted as colors and will be used to supply the color values for the console view's color map, starting with the first entry in the map. (See the **COLOR SPECIFICATION** section of this manual page for information on how to specify colors.) If more colors are supplied than are usable by the console view, a warning is printed and the extra colors are ignored.

COLOR SPECIFICATION

Colors are hexadecimal numbers which have one of the following formats:

0xRRGGBB	*RR*, *GG*, and *BB* are taken to be eight-bit values specifying the intensities of the red, green and blue components, respectively, of the color to be used. For example, `0xff0000` is bright red, `0xffffff` is white, and `0x008080` is dark cyan.
0xGG	*GG* is taken to be an eight-bit value specifying the intensity of grey to be used. A value of `0x00` is black, a value of `0xff` is white, and a value of `0x80` is a grey approximately half way in between.
0xM	*M* is taken to be the one-bit monochrome value to be used. A value of `0x1` is black, and a value of `0x0` is white.

BUGS

The **iteconfig** command is only available on the amiga and atari ports.

NAME

iwictl — configure Intel(R) PRO/Wireless 2200BG/2915ABG network adapters

SYNOPSIS

iwictl [**-i**] *iface*
iwictl [**-i**] *iface* **-r**

DESCRIPTION

The **iwictl** utility controls the operation of Intel(R) PRO/Wireless 2200BG/2915ABG networking devices via the `iwi`(4) driver.

You should not use this program to configure IEEE 802.11 parameters. Use `ifconfig`(8) instead.

OPTIONS

The options are as follows:

-i *iface*
Displays adapter's internal statistics. Firmware binary images can be found in the `pkgsrc/sysutils/iwi-firmware` package.

-i *iface* **-r**
Displays the radio transmitter state (on or off).

FILES

The firmware is not shipped with NetBSD. It can be obtained via installing the `pkgsrc/sysutils/iwi-firmware` package and is loaded automatically by the kernel once installed.

SEE ALSO

`iwi`(4), `ifconfig`(8) `pkgsrc/sysutils/iwi-firmware`

AUTHORS

The **iwictl** utility and this man page were written by Damien Bergamini ⟨damien.bergamini@free.fr⟩.

CAVEATS

Monitor mode requires firmware version 1.0.4 (included in the `pkgsrc/sysutils/iwi-firmware` package version 2.3) or newer.

NAME

kadmin — Kerberos administration utility

SYNOPSIS

kadmin [**−p** *string* | **−−principal=***string*] [**−K** *string* | **−−keytab=***string*]
[**−c** *file* | **−−config-file=***file*] [**−k** *file* | **−−key-file=***file*]
[**−r** *realm* | **−−realm=***realm*] [**−a** *host* | **−−admin-server=***host*]
[**−s** *port number* | **−−server-port=***port number*] [**−l** | **−−local**]
[**−h** | **−−help**] [**−v** | **−−version**] [*command*]

DESCRIPTION

The **kadmin** program is used to make modifications to the Kerberos database, either remotely via the kadmind(8) daemon, or locally (with the **−l** option).

Supported options:

−p *string*, **−−principal=***string*
principal to authenticate as

−K *string*, **−−keytab=***string*
keytab for authentication principal

−c *file*, **−−config-file=***file*
location of config file

−k *file*, **−−key-file=***file*
location of master key file

−r *realm*, **−−realm=***realm*
realm to use

−a *host*, **−−admin-server=***host*
server to contact

−s *port number*, **−−server-port=***port number*
port to use

−l, **−−local**
local admin mode

If no *command* is given on the command line, **kadmin** will prompt for commands to process. Some of the commands that take one or more principals as argument (**delete**, **ext_keytab**, **get**, **modify**, and **passwd**) will accept a glob style wildcard, and perform the operation on all matching principals.

Commands include:

add [**−r** | **−−random-key**] [**−−random-password**] [**−p** *string* | **−−password=***string*] [**−−key=***string*] [**−−max-ticket-life=***lifetime*] [**−−max-renewable-life=***lifetime*] [**−−attributes=***attributes*] [**−−expiration-time=***time*] [**−−pw-expiration-time=***time*] *principal...*

Adds a new principal to the database. The options not passed on the command line will be promped for.

add_enctype [**−r** | **−−random-key**] *principal enctypes...*

Adds a new encryption type to the principal, only random key are supported.

delete *principal...*

Removes a principal.

del_enctype *principal enctypes...*

Removes some enctypes from a principal; this can be useful if the service belonging to the principal is known to not handle certain enctypes.

ext_keytab [**−k** *string* | **−-keytab=***string*] *principal...*

Creates a keytab with the keys of the specified principals.

get [**−l** | **−-long**] [**−s** | **−-short**] [**−t** | **−-terse**] [**−o** *string* | **−-column-info=***string*] *principal...*

Lists the matching principals, short prints the result as a table, while long format produces a more verbose output. Which columns to print can be selected with the **−o** option. The argument is a comma separated list of column names optionally appended with an equal sign ('=') and a column header. Which columns are printed by default differ slightly between short and long output.

The default terse output format is similar to **−s −o** *principal=*, just printing the names of matched principals.

Possible column names include: `principal`, `princ_expire_time`, `pw_expiration`, `last_pwd_change`, `max_life`, `max_rlife`, `mod_time`, `mod_name`, `attributes`, `kvno`, `mkvno`, `last_success`, `last_failed`, `fail_auth_count`, `policy`, and `keytypes`.

modify [**−a** *attributes* | **−-attributes=***attributes*] [**−-max-ticket-life=***lifetime*] [**−-max-renewable-life=***lifetime*] [**−-expiration-time=***time*] [**−-pw-expiration-time=***time*] [**−-kvno=***number*] *principal...*

Modifies certain attributes of a principal. If run without command line options, you will be prompted. With command line options, it will only change the ones specified.

Possible attributes are: `new-princ`, `support-desmd5`, `pwchange-service`, `disallow-svr`, `requires-pw-change`, `requires-hw-auth`, `requires-pre-auth`, `disallow-all-tix`, `disallow-dup-skey`, `disallow-proxiable`, `disallow-renewable`, `disallow-tgt-based`, `disallow-forwardable`, `disallow-postdated`

Attributes may be negated with a "-", e.g.,

kadmin -l modify -a -disallow-proxiable user

passwd [**−r** | **−-random-key**] [**−-random-password**] [**−p** *string* | **−-password=***string*] [**−-key=***string*] *principal...*

Changes the password of an existing principal.

password-quality *principal password*

Run the password quality check function locally. You can run this on the host that is configured to run the kadmind process to verify that your configuration file is correct. The verification is done locally, if kadmin is run in remote mode, no rpc call is done to the server.

privileges

Lists the operations you are allowed to perform. These include `add`, `add_enctype`, `change-password`, `delete`, `del_enctype`, `get`, `list`, and `modify`.

rename *from to*

Renames a principal. This is normally transparent, but since keys are salted with the principal name, they will have a non-standard salt, and clients which are unable to cope with this will fail. Kerberos 4 suffers from this.

check [*realm*]

Check database for strange configurations on important principals. If no realm is given, the default realm is used.

When running in local mode, the following commands can also be used:

dump [**−d** | **−-decrypt**] [*dump-file*]

Writes the database in "human readable" form to the specified file, or standard out. If the database is encrypted, the dump will also have encrypted keys, unless **−-decrypt** is used.

init [**−-realm-max-ticket-life=***string*]
[**−-realm-max-renewable-life=***string*] *realm*

Initializes the Kerberos database with entries for a new realm. It's possible to have more than one realm served by one server.

load *file*

Reads a previously dumped database, and re-creates that database from scratch.

merge *file*

Similar to **load** but just modifies the database with the entries in the dump file.

stash [**−e** *enctype* | **−-enctype=***enctype*] [**−k** *keyfile* |
−-key-file=*keyfile*] [**−-convert-file**] [**−-master-key-fd=***fd*]

Writes the Kerberos master key to a file used by the KDC.

SEE ALSO

`kadmind`(8), `kdc`(8)

NAME

kadmind — server for administrative access to Kerberos database

SYNOPSIS

kadmind [**−c** *file* | **−-config-file=***file*] [**−k** *file* | **−-key-file=***file*] [**−-keytab=***keytab*] [**−r** *realm* | **−-realm=***realm*] [**−d** | **−-debug**] [**−p** *port* | **−-ports=***port*]

DESCRIPTION

kadmind listens for requests for changes to the Kerberos database and performs these, subject to permissions. When starting, if stdin is a socket it assumes that it has been started by inetd(8), otherwise it behaves as a daemon, forking processes for each new connection. The **−-debug** option causes **kadmind** to accept exactly one connection, which is useful for debugging.

The kpasswdd(8) daemon is responsible for the Kerberos 5 password changing protocol (used by kpasswd(1))

This daemon should only be run on the master server, and not on any slaves.

Principals are always allowed to change their own password and list their own principal. Apart from that, doing any operation requires permission explicitly added in the ACL file /var/heimdal/kadmind.acl. The format of this file is:

principal rights [*principal-pattern*]

Where rights is any (comma separated) combination of:
- change-password or cpw
- list
- delete
- modify
- add
- get
- all

And the optional *principal-pattern* restricts the rights to operations on principals that match the glob-style pattern.

Supported options:

−c *file*, **−-config-file=***file*
location of config file

−k *file*, **−-key-file=***file*
location of master key file

−-keytab=*keytab*
what keytab to use

−r *realm*, **−-realm=***realm*
realm to use

−d, **−-debug**
enable debugging

−p *port*, **−-ports=***port*
ports to listen to. By default, if run as a daemon, it listens to port 749, but you can add any number of ports with this option. The port string is a whitespace separated list of port specifications, with the special string "+" representing the default port.

FILES

`/var/heimdal/kadmind.acl`

EXAMPLES

This will cause **kadmind** to listen to port 4711 in addition to any compiled in defaults:

```
kadmind --ports="+ 4711" &
```

This acl file will grant Joe all rights, and allow Mallory to view and add host principals.

```
joe/admin@EXAMPLE.COM      all
mallory/admin@EXAMPLE.COM  add,get  host/*@EXAMPLE.COM
```

SEE ALSO

kpasswd(1), kadmin(8), kdc(8), kpasswdd(8)

NAME

kcm — is a process based credential cache for Kerberos tickets.

SYNOPSIS

kcm [**--cache-name=***cachename*] [**-c** *file* | **--config-file=***file*] [**-g** *group* | **--group=***group*] [**--max-request=***size*] [**--disallow-getting-krbtgt**] [**--detach**] [**-h** | **--help**] [**-k** *principal* | **--system-principal=***principal*] [**-l** *time* | **--lifetime=***time*] [**-m** *mode* | **--mode=***mode*] [**-n** | **--no-name-constraints**] [**-r** *time* | **--renewable-life=***time*] [**-s** *path* | **--socket-path=***path*] [**--door-path=***path*] [**-S** *principal* | **--server=***principal*] [**-t** *keytab* | **--keytab=***keytab*] [**-u** *user* | **--user=***user*] [**-v** | **--version**]

DESCRIPTION

kcm is a process based credential cache. To use it, set the KRB5CCNAME enviroment variable to KCM:*uid* or add the stanza

```
[libdefaults]
        default_cc_name = KCM:%{uid}
```

to the /etc/krb5.conf configuration file and make sure **kcm** is started in the system startup files.

The **kcm** daemon can hold the credentials for all users in the system. Access control is done with Unix-like permissions. The daemon checks the access on all operations based on the uid and gid of the user. The tickets are renewed as long as is permitted by the KDC's policy.

The **kcm** daemon can also keep a SYSTEM credential that server processes can use to access services. One example of usage might be an nss_ldap module that quickly needs to get credentials and doesn't want to renew the ticket itself.

Supported options:

--cache-name=*cachename*
system cache name

-c *file*, **--config-file=***file*
location of config file

-g *group*, **--group=***group*
system cache group

--max-request=*size*
max size for a kcm-request

--disallow-getting-krbtgt
disallow extracting any krbtgt from the **kcm** daemon.

--detach
detach from console

-h, **--help**

-k *principal*, **--system-principal=***principal*
system principal name

−l *time*, **−-lifetime=***time*
lifetime of system tickets

−m *mode*, **−-mode=***mode*
octal mode of system cache

−n, **−-no-name-constraints**
disable credentials cache name constraints

−r *time*, **−-renewable-life=***time*
renewable lifetime of system tickets

−s *path*, **−-socket-path=***path*
path to kcm domain socket

−-door-path=*path*
path to kcm door socket

−S *principal*, **−-server=***principal*
server to get system ticket for

−t *keytab*, **−-keytab=***keytab*
system keytab name

−u *user*, **−-user=***user*
system cache owner

−v, **−-version**

NAME

kdc — Kerberos 5 server

SYNOPSIS

kdc [**−c** *file* | **−-config-file=***file*] [**−p** | **−-no-require-preauth**]
[**−-max-request=***size*] [**−H** | **−-enable-http**] [**−-no-524**] [**−-kerberos4**]
[**−-kerberos4-cross-realm**] [**−r** *string* | **−-v4-realm=***string*]
[**−K** | **−-kaserver**] [**−P** *portspec* | **−-ports=***portspec*] [**−-detach**]
[**−-disable-DES**] [**−-addresses=***list of addresses*]

DESCRIPTION

kdc serves requests for tickets. When it starts, it first checks the flags passed, any options that are not specified with a command line flag are taken from a config file, or from a default compiled-in value.

Options supported:

−c *file*, **−-config-file=***file*
Specifies the location of the config file, the default is `/var/heimdal/kdc.conf`. This is the only value that can't be specified in the config file.

−p, **−-no-require-preauth**
Turn off the requirement for pre-autentication in the initial AS-REQ for all principals. The use of pre-authentication makes it more difficult to do offline password attacks. You might want to turn it off if you have clients that don't support pre-authentication. Since the version 4 protocol doesn't support any pre-authentication, serving version 4 clients is just about the same as not requiring preathentication. The default is to require pre-authentication. Adding the require-preauth per principal is a more flexible way of handling this.

−-max-request=*size*
Gives an upper limit on the size of the requests that the kdc is willing to handle.

−H, **−-enable-http**
Makes the kdc listen on port 80 and handle requests encapsulated in HTTP.

−-no-524
don't respond to 524 requests

−-kerberos4
respond to Kerberos 4 requests

−-kerberos4-cross-realm
respond to Kerberos 4 requests from foreign realms. This is a known security hole and should not be enabled unless you understand the consequences and are willing to live with them.

−r *string*, **−-v4-realm=***string*
What realm this server should act as when dealing with version 4 requests. The database can contain any number of realms, but since the version 4 protocol doesn't contain a realm for the server, it must be explicitly specified. The default is whatever is returned by **krb_get_lrealm**(). This option is only availabe if the KDC has been compiled with version 4 support.

−K, **−-kaserver**
Enable kaserver emulation (in case it's compiled in).

−P *portspec*, **−-ports=***portspec*
Specifies the set of ports the KDC should listen on. It is given as a white-space separated list of services or port numbers.

−-addresses=*list of addresses*
The list of addresses to listen for requests on. By default, the kdc will listen on all the locally configured addresses. If only a subset is desired, or the automatic detection fails, this option might be used.

−-detach
detach from pty and run as a daemon.

−-disable-DES
disable add des encryption types, makes the kdc not use them.

All activities are logged to one or more destinations, see `krb5.conf`(5), and `krb5_openlog`(3). The entity used for logging is **kdc**.

CONFIGURATION FILE

The configuration file has the same syntax as `krb5.conf`(5), but will be read before `/etc/krb5.conf`, so it may override settings found there. Options specific to the KDC only are found in the "[kdc]" section. All the command-line options can preferably be added in the configuration file. The only difference is the pre-authentication flag, which has to be specified as:

```
require-preauth = no
```

(in fact you can specify the option as **−-require-preauth=no**).

And there are some configuration options which do not have command-line equivalents:

`enable-digest =` *boolean*
turn on support for digest processing in the KDC. The default is FALSE.

`check-ticket-addresses =` *boolean*
Check the addresses in the ticket when processing TGS requests. The default is TRUE.

`allow-null-ticket-addresses =` *boolean*
Permit tickets with no addresses. This option is only relevant when check-ticket-addresses is TRUE.

`allow-anonymous =` *boolean*
Permit anonymous tickets with no addresses.

`max-kdc-datagram-reply-length =` *number*
Maximum packet size the UDP rely that the KDC will transmit, instead the KDC sends back a reply telling the client to use TCP instead.

`transited-policy = always-check | allow-per-principal | always-honour-request`
This controls how KDC requests with the `disable-transited-check` flag are handled. It can be one of:

`always-check`
Always check transited encoding, this is the default.

`allow-per-principal`
Currently this is identical to `always-check`. In a future release, it will be possible to mark a principal as able to handle unchecked requests.

`always-honour-request`
Always do what the client asked. In a future release, it will be possible to force a check per principal.

encode_as_rep_as_tgs_rep = *boolean*
: Encode AS-Rep as TGS-Rep to be bug-compatible with old DCE code. The Heimdal clients allow both.

kdc_warn_pwexpire = *time*
: How long before password/principal expiration the KDC should start sending out warning messages.

The configuration file is only read when the **kdc** is started. If changes made to the configuration file are to take effect, the **kdc** needs to be restarted.

An example of a config file:

```
[kdc]
        require-preauth = no
        v4-realm = FOO.SE
```

BUGS

If the machine running the KDC has new addresses added to it, the KDC will have to be restarted to listen to them. The reason it doesn't just listen to wildcarded (like INADDR_ANY) addresses, is that the replies has to come from the same address they were sent to, and most OS:es doesn't pass this information to the application. If your normal mode of operation require that you add and remove addresses, the best option is probably to listen to a wildcarded TCP socket, and make sure your clients use TCP to connect. For instance, this will listen to IPv4 TCP port 88 only:

```
kdc --addresses=0.0.0.0 --ports="88/tcp"
```

There should be a way to specify protocol, port, and address triplets, not just addresses and protocol, port tuples.

SEE ALSO

kinit(1), krb5.conf(5)

NAME

kerberos — introduction to the Kerberos system

DESCRIPTION

Kerberos is a network authentication system. Its purpose is to securely authenticate users and services in an insecure network environment.

This is done with a Kerberos server acting as a trusted third party, keeping a database with secret keys for all users and services (collectively called *principals*).

Each principal belongs to exactly one *realm*, which is the administrative domain in Kerberos. A realm usually corresponds to an organisation, and the realm should normally be derived from that organisation's domain name. A realm is served by one or more Kerberos servers.

The authentication process involves exchange of 'tickets' and 'authenticators' which together prove the principal's identity.

When you login to the Kerberos system, either through the normal system login or with the `kinit`(1) program, you acquire a *ticket granting ticket* which allows you to get new tickets for other services, such as **telnet** or **ftp**, without giving your password.

For more information on how Kerberos works, and other general Kerberos questions see the Kerberos FAQ at `http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html`.

For setup instructions see the Heimdal Texinfo manual.

SEE ALSO

`ftp`(1), `kdestroy`(1), `kinit`(1), `klist`(1), `kpasswd`(1), `telnet`(1)

HISTORY

The Kerberos authentication system was developed in the late 1980's as part of the Athena Project at the Massachusetts Institute of Technology. Versions one through three never reached outside MIT, but version 4 was (and still is) quite popular, especially in the academic community, but is also used in commercial products like the AFS filesystem.

The problems with version 4 are that it has many limitations, the code was not too well written (since it had been developed over a long time), and it has a number of known security problems. To resolve many of these issues work on version five started, and resulted in IETF RFC 1510 in 1993. IETF RFC 1510 was obsoleted in 2005 with IETF RFC 4120, also known as Kerberos clarifications. With the arrival of IETF RFC 4120, the work on adding extensibility and internationalization have started (Kerberos extensions), and a new RFC will hopefully appear soon.

This manual page is part of the **Heimdal** Kerberos 5 distribution, which has been in development at the Royal Institute of Technology in Stockholm, Sweden, since about 1997.

NAME

kfd — receive forwarded tickets

SYNOPSIS

kfd [**−p** *port* | **−-port**=*port*] [**−i** | **−-inetd**] [**−R** *regpag* | **−-regpag**=*regpag*]
[**−h** | **−-help**] [**−-version**]

DESCRIPTION

This is the daemon for `kf`(1). Supported options:

−p *port*, **−-port**=*port*
port to listen to

−i, **−-inetd**
not started from inetd

−R *regpag*, **−-regpag=***regpag*
path to regpag binary

EXAMPLES

Put the following in `/etc/inetd.conf`:

```
kf      stream  tcp     nowait  root    /usr/heimdal/libexec/kfd      kfd
```

SEE ALSO

`kf`(1)

NAME

kgmon — generate a dump of the operating system's profile buffers

SYNOPSIS

kgmon [**-bdhpr**] [**-M** *core*] [**-N** *system*]

DESCRIPTION

kgmon is a tool used when profiling the operating system. When no arguments are supplied, **kgmon** indicates the state of operating system profiling as running, off, or not configured (see config(1)). If the **-p** flag is specified, **kgmon** extracts profile data from the operating system and produces a gmon.out file suitable for later analysis by gprof(1).

The options are as follows:

-b Resume the collection of profile data.

-d Enable debug output.

-h Stop the collection of profile data.

-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.

-p Dump the contents of the profile buffers into a gmon.out file.

-r Reset all the profile buffers. If the **-p** flag is also specified, the gmon.out file is generated before the buffers are reset.

If neither **-b** nor **-h** is specified, the state of profiling collection remains unchanged. For example, if the **-p** flag is specified and profile data is being collected, profiling will be momentarily suspended, the operating system profile buffers will be dumped, and profiling will be immediately resumed.

FILES

/netbsd	the default system
/dev/kmem	the default memory

DIAGNOSTICS

Users with only read permission on /dev/kmem cannot change the state of profiling collection. They can get a gmon.out file with the warning that the data may be inconsistent if profiling is in progress.

SEE ALSO

config(1), gprof(1)

HISTORY

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

NAME

kpasswdd — Kerberos 5 password changing server

SYNOPSIS

kpasswdd [**--addresses=***address*] [**--check-library=***library*] [**--check-function=***function*] [**-k** *kspec* | **--keytab=***kspec*] [**-r** *realm* | **--realm=***realm*] [**-p** *string* | **--port=***string*] [**--version**] [**--help**]

DESCRIPTION

kpasswdd serves request for password changes. It listens on UDP port 464 (service kpasswd) and processes requests when they arrive. It changes the database directly and should thus only run on the master KDC.

Supported options:

--addresses=*address*
: For each till the argument is given, add the address to what kpasswdd should listen too.

--check-library=*library*
: If your system has support for dynamic loading of shared libraries, you can use an external function to check password quality. This option specifies which library to load.

--check-function=*function*
: This is the function to call in the loaded library. The function should look like this:

const char * **passwd_check**(*krb5_context context*, *krb5_principal principal*, *krb5_data *password*)

context is an initialized context; *principal* is the one who tries to change passwords, and *password* is the new password. Note that the password (in *password->data*) is not zero terminated.

-k *kspec*, **--keytab=***kspec*
: Keytab to get authentication key from.

-r *realm*, **--realm=***realm*
: Default realm.

-p *string*, **--port=***string*
: Port to listen on (default service kpasswd - 464).

DIAGNOSTICS

If an error occurs, the error message is returned to the user and/or logged to syslog.

BUGS

The default password quality checks are too basic.

SEE ALSO

kpasswd(1), kdc(8)

NAME

kstash — store the KDC master password in a file

SYNOPSIS

kstash [**−e** *string* | **−-enctype=***string*] [**−k** *file* | **−-key-file=***file*] [**−-convert-file**] [**−-random-key**] [**−-master-key-fd=***fd*] [**−-random-key**] [**−h** | **−-help**] [**−-version**]

DESCRIPTION

kstash reads the Kerberos master key and stores it in a file that will be used by the KDC.

Supported options:

−e *string*, **−-enctype=***string*
: the encryption type to use, defaults to DES3-CBC-SHA1.

−k *file*, **−-key-file=***file*
: the name of the master key file.

−-convert-file
: don't ask for a new master key, just read an old master key file, and write it back in the new keyfile format.

−-random-key
: generate a random master key.

−-master-key-fd=*fd*
: filedescriptor to read passphrase from, if not specified the passphrase will be read from the terminal.

FILES

`/var/heimdal/m-key` is the default keyfile if no other keyfile is specified. The format of a Heimdal master key is the same as a keytab, so **ktutil** list can be used to list the content of the file.

SEE ALSO

kdc(8)

NAME

ktutil — manage Kerberos keytabs

SYNOPSIS

ktutil [**-k** *keytab* | **--keytab=***keytab*] [**-v** | **--verbose**] [**--version**]
[**-h** | **--help**] *command* [*args*]

DESCRIPTION

ktutil is a program for managing keytabs. Supported options:

-v, **--verbose**
Verbose output.

command can be one of the following:

add [**-p** *principal*] [**--principal=***principal*] [**-V** *kvno*] [**--kvno=***kvno*] [**-e** *enctype*] [**--enctype=***enctype*] [**-w** *password*] [**--password=***password*] [**-r**] [**--random**] [**-s**] [**--no-salt**] [**-H**] [**--hex**]
Adds a key to the keytab. Options that are not specified will be prompted for. This requires that you know the password or the hex key of the principal to add; if what you really want is to add a new principal to the keytab, you should consider the *get* command, which talks to the kadmin server.

change [**-r** *realm*] [**--realm=***realm*] [**--a** *host*] [**--admin-server=***host*] [**--s** *port*] [**--server-port=***port*]
Update one or several keys to new versions. By default, use the admin server for the realm of a keytab entry. Otherwise it will use the values specified by the options.

If no principals are given, all the ones in the keytab are updated.

copy *keytab-src keytab-dest*
Copies all the entries from *keytab-src* to *keytab-dest*.

get [**-p** *admin principal*] [**--principal=***admin principal*] [**-e** *enctype*] [**--enctypes=***enctype*] [**-r** *realm*] [**--realm=***realm*] [**-a** *admin server*] [**--admin-server=***admin server*] [**-s** *server port*] [**--server-port=***server port*] *principal ...*
For each *principal*, generate a new key for it (creating it if it doesn't already exist), and put that key in the keytab.

If no *realm* is specified, the realm to operate on is taken from the first principal.

list [**--keys**] [**--timestamp**]
List the keys stored in the keytab.

remove [**-p** *principal*] [**--principal=***principal*] [**-V** **-kvno**] [**--kvno=***kvno*] [**-e** **-enctype**] [**--enctype=***enctype*]
Removes the specified key or keys. Not specifying a *kvno* removes keys with any version number. Not specifying an *enctype* removes keys of any type.

rename *from-principal to-principal*
Renames all entries in the keytab that match the *from-principal* to *to-principal*.

purge [**--age=***age*]
Removes all old versions of a key for which there is a newer version that is at least *age* (default one week) old.

srvconvert

srv2keytab [**−s** *srvtab*] [**−−srvtab=***srvtab*]
Converts the version 4 srvtab in *srvtab* to a version 5 keytab and stores it in *keytab*. Identical to:

```
ktutil copy krb4:srvtab keytab
```

srvcreate

key2srvtab [**−s** *srvtab*] [**−−srvtab=***srvtab*]
Converts the version 5 keytab in *keytab* to a version 4 srvtab and stores it in *srvtab*. Identical to:

```
ktutil copy keytab krb4:srvtab
```

SEE ALSO

kadmin(8)

NAME

kvm_mkdb — create kernel database

SYNOPSIS

kvm_mkdb [**-o** *database*] [file]

DESCRIPTION

kvm_mkdb creates a database, /var/db/kvm.db, containing information about the specified file. If no file is specified, /dev/ksyms is used by default. The database will only be recreated if its contents do not already describe the running kernel.

The options are as follows:

-o *database*
Put the output databases in the named file.

Various library routines consult this database. The only information currently stored is the kernel namelist, which is used by the kvm_nlist(3) function. In the future, the database may contain other static information about the current system.

FILES

/netbsd
/var/db/kvm.db

SEE ALSO

kvm_nlist(3)

HISTORY

The **kvm_mkdb** utility first appeared in 4.4BSD.

BUGS

The database is not updated when kernel modules are loaded.

NAME

kxd — securely forward X conections

SYNOPSIS

kxd [**-t**] [**-i**] [**-p** *port*]

DESCRIPTION

This is the daemon for **kx**.

Options supported by **kxd**:

-t TCP. Normally **kxd** will only listen for X connections on a UNIX socket, but some machines (for example, Cray) have X libraries that are not able to use UNIX sockets and thus you need to use TCP to talk to the pseudo-xserver created by **kxd**. This option decreases the security significantly and should only be used when it is necessary and you have considered the consequences of doing so.

-i Interactive. Do not expect to be started by **inetd**, but allocate and listen to the socket yourself. Handy for testing and debugging.

-p Port. Listen on the port *port*. Only usable with **-i**.

EXAMPLES

Put the following in /etc/inetd.conf:

```
kx      stream  tcp     nowait  root    /usr/athena/libexec/kxd       kxd
```

SEE ALSO

kx(1), rxtelnet(1), rxterm(1)

NAME

lastlogin — indicate last login time of users

SYNOPSIS

lastlogin [**-nrt**] [**-f** *filename*] [**-H** *hostsize*] [**-L** *linesize*] [**-N** *namesize*] [user ...]

DESCRIPTION

lastlogin will list the last login session of specified *users*, or for all users by default. Each line of output contains the user name, the tty from which the session was conducted, any hostname, and the start time for the session.

If multiple *users* are given, the session information for each user is printed in the order given on the command line. Otherwise, information for all users is printed, sorted by uid.

lastlogin differs from last(1) in that it only prints information regarding the very last login session. The last login database is never turned over or deleted in standard usage.

The following options are available:

-f *filename*
Process input from *filename*. If the file ends with an "x", then it is assumed that it is a lastlogx(5) file, else it is assumed to be a lastlog(5) file.

-H *hostlen*
Set the field width for host output to *hostlen* characters.

-L *linelen*
Set the field width for line output to *linelen* characters.

-N *namelen*
Set the field width for name output to *namelen* characters.

-n Attempt to print numeric host addresses. This option is only supported with lastlogx(5) format files.

-r Reverse the order of the sort.

-t Sort by last login time (most recent first.)

FILES

/var/log/lastlogx	default last login database
/var/log/lastlog	compatibility last login database

EXAMPLES

lastlogin looks by default to the /var/log/lastlogx database, where some old programs that are not utmpx(5) aware might only write to /var/log/lastlog. To look at the old database one can use:

```
lastlogin -f /var/log/lastlog
```

SEE ALSO

last(1), lastlog(5), lastlogx(5), ac(8)

AUTHORS

John M. Vinopal wrote this program in January 1996 and contributed it to the NetBSD project.

NAME

ldconfig — configure the a.out shared library cache

SYNOPSIS

ldconfig [**−cmrsSv**] [*directory ...*]

DESCRIPTION

ldconfig is used to prepare a set of "hints" for use by the a.out run-time linker **ld.so** to facilitate quick lookup of shared libraries available in multiple directories. **ldconfig** is only available on systems that use the "a.out" format for executables and libraries – on ELF systems, all the work is done by **ld.elf_so**.

By default, it scans a set of built-in system directories, directories listed in /etc/ld.so.conf, and any *directories* specified on the command line (in the given order) looking for shared libraries and stores the results in the file /var/run/ld.so.hints to forestall the overhead that would otherwise result from the directory search operations **ld.so** would have to perform to load required shared libraries.

The shared libraries so found will be automatically available for loading if needed by the program being prepared for execution. This obviates the need for storing search paths within the executable.

The LD_LIBRARY_PATH environment variable can be used to override the use of directories (or the order thereof) from the cache or to specify additional directories where shared libraries might be found. LD_LIBRARY_PATH is a ':' separated list of directory paths that are searched by **ld.so** when it needs to load a shared library. It can be viewed as the run-time equivalent of the **−L** switch of **ld**.

ldconfig is typically run as part of the boot sequence.

The following options are recognized by **ldconfig**:

−c	Do not scan directories listed in /etc/ld.so.conf for shared libraries.
−m	Merge the result of the scan of the directories given as arguments into the existing hints file. The default action is to build the hints file afresh.
−r	Lists the current contents of ld.so.hints on the standard output. The hints file will not be modified.
−s	Do not scan the built-in system directory (/usr/lib), nor any directories listed in /etc/ld.so.conf for shared libraries.
−S	Do not scan the built-in system directory (/usr/lib), for shared libraries. (Directories listed in /etc/ld.so.conf are still scanned.)
−v	Switch on verbose mode.

FILES

/var/run/ld.so.hints, /etc/ld.so.conf

SEE ALSO

ld(1), ld.so(1), ld.so.conf(5), link(5)

HISTORY

A **ldconfig** utility first appeared in SunOS 4.0, it appeared in its current form in NetBSD 0.9A.

SECURITY CONSIDERATIONS

Special care must be taken when loading shared libraries into the address space of *set-user-ID* programs. Whenever such a program is run, **ld.so** will only load shared libraries from the ld.so.hints file. In particular, the LD_LIBRARY_PATH and LD_PRELOAD is not used to search for libraries. Thus, the role of ldconfig is dual. In addition to building a set of hints for quick lookup, it also serves to specify the trusted collection of directories from which shared objects can be safely loaded. It is presumed that the set of directories specified to **ldconfig** is under control of the system's administrator. **ld.so** further assists set-user-ID programs by erasing the `LD_LIBRARY_PATH` and `LD_PRELOAD` from the environment.

NAME

lfs_cleanerd — garbage collect a log-structured file system

SYNOPSIS

lfs_cleanerd [**-bcdfmqs**] [**-i** *segment-number*] [**-l** *load-threshhold*]
[**-n** *number-of-segments*] [**-r** *report-frequency*] [**-t** *timeout*]
node

DESCRIPTION

The **lfs_cleanerd** command starts a daemon process which garbage-collects the log-structured file system residing at the point named by *node* in the global file system namespace. This command is normally executed by mount_lfs(8) when the log-structured file system is mounted. The daemon will exit within a few minutes of when the file system it was cleaning is unmounted.

Garbage collection on a log-structured file system is done by scanning the file system's segments for active, i.e. referenced, data and copying it to new segments. When all of the active data in a given segment has been copied to a new segment that segment can be marked as empty, thus reclaiming the space taken by the inactive data which was in it.

The following options are available:

-b Use bytes written, rather than segments read, when determining how many segments to clean at once.

-c Coalescing mode. For each live inode, check to see if it has too many blocks that are not contiguous, and if it does, rewrite it. After a single pass through the filesystem the cleaner will exit. This option has been reported to corrupt file data; do not use it.

-d Run in debug mode. Do not become a daemon process, and print debugging information. More **-d** s give more detailed debugging information.

-f Use filesystem idle time as the criterion for aggressive cleaning, instead of system load.

-i *segment-number*
Invalidate the segment with segment number *segment-number*. This option is used by resize_lfs(8), and should not be specified on the command line.

-l *load-threshhold*
Clean more aggressively when the system load is below the given threshhold. The default threshhold is 0.2.

-m Does nothing. This option is present for historical compatibility.

-n *number-of-segments*
Clean this number of segments at a time: that is, pass this many segments' blocks through a single call to lfs_markv, or, if **-b** was also given, pass this many segments' worth of blocks through a single call to lfs_markv.

-q Quit after cleaning once.

-r *report-frequency*
Give an efficiency report after every *report-frequency* times through the main loop.

-s When cleaning the file system, send only a few blocks through lfs_markv at a time. Don't use this option.

-t *timeout*
Poll the filesystem every *timeout* seconds, looking for opportunities to clean. The default is 300, that is, five minutes. Note that **lfs_cleanerd** will be automatically awakened when the

filesystem is active, so it is not usually necessary to set *timeout* to a low value.

SEE ALSO

lfs_bmapv(2), lfs_markv(2), lfs_segwait(2), mount_lfs(8)

HISTORY

The **lfs_cleanerd** utility first appeared in 4.4BSD.

NAME

link — call the **link**() function

SYNOPSIS

link *file1* *file2*

DESCRIPTION

The **link** utility performs the function call **link**(*file1* , *file2*).

file1 must be the pathname of an existing file, and *file2* is the pathname of the new link to *file1* to be created.

EXIT STATUS

The **link** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

ln(1), link(2), unlink(8)

STANDARDS

The **link** utility conforms to X/Open Commands and Utilities Issue 5 ("XCU5").

NAME

lmcconfig — configuration program for LMC (and some SBE) wide-area network interface cards

SYNOPSIS

lmcconfig *interface* [**-abBcCdDeEfgGhiLmMpPsStTuUvVwxXyY?**]
lmcconfig *interface* **-1** [**-aABcdeEfFgiIlLpPstTuUxX**]
lmcconfig *interface* **-3** [**-aABcefFlLsSV**]

DESCRIPTION

The **lmcconfig** utility is the configuration program for the `lmc`(4) wide-area network device driver. It sets control values, such as T3 framing format, and it displays status, such as that of integrated modems, which are beyond the scope of `ifconfig`(8).

The **lmcconfig** utility displays the interface status when no parameters are specified; see the **EXAMPLES** section. For this case only, if no *interface* is specified, it defaults to "`lmc0`".

Only the super-user may modify the configuration of a network interface.

The following options are available:

interface This is the name of the interface; the default is "`lmc0`".

-1 All parameters after this apply to the T1E1 card.

-3 All parameters after this apply to the T3 card.

Parameters for all cards

The following parameters apply to more then one card type.

-a *number* Set Transmitter clock source to *number*.

1	TxClk from modem	T1E1, HSSI	(default)
2	Internal source	T1E1, HSSI	
3	RxClk from modem	T1E1, HSSIc	(loop timed)
4	External connector	T1E1, HSSIc	

An HSSI card normally takes its Tx clock from the modem connector (it is a DTE) but can use the PCI bus clock (typically 33 MHz) for loopback and null modem testing; values 3 and 4 are only applicable to a few rare CompactPCI/HSSI cards.

A T1E1 card uses an on-board synthesized oscillator if the value is 1 or 2; it *loop times* (uses the clock recovered by the receiver as the transmitter clock) if the value is 3; and it uses a clock from a header connector on the card if the value is 4.

TxClk source is not applicable to other card types.

-b Read BIOS ROM. Print the first 256 locations. The BIOS ROM is not used and not present on some cards.

-B Write BIOS ROM. Write the first 256 locations with an address pattern.

-c Use HDLC's 16-bit Cyclic Redundancy Checksum (CRC).

-C Use HDLC's 32-bit Cyclic Redundancy Checksum (CRC).

-d Clear the driver-level debug flag. Non-critical log messages are suppressed.

-D Set the driver-level debug flag. The driver generates more log messages. The driver also generates more log messages if the interface-level debug flag is set by `ifconfig`(8).

-e Set DTE (Data Terminal Equipment) mode (default). An SSI card transmitter uses the Tx clock signal from the modem connector and receives the Data Carrier Detect pin (DCD). DTE/DCE is not applicable to other card types except a few rare CompactPCI/HSSI cards.

-E Set DCE (Data Communication Equipment) mode. An SSI card transmitter uses an on-board synthesized oscillator and drives the Data Carrier Detect pin (DCD).

-f *number* Set the frequency of the built-in synthesized oscillator to *number* bits/second. The nearest frequency that the synthesizer can generate will be used. Only SSI cards and a few rare CompactPCI/HSSI cards have synthesizers.

-g Load gate array microcode from on-board ROM; see also **-U.**

-G *filename* Load gate array microcode from *filename*; see also **-U.**

-h Print help (usage message).

-i Set interface name (e.g. "lmc0").

-L *number* Set loopback mode to *number*.

1	none	default	
2	payload	outward thru framer	T1E1. T3
3	line	outward thru line if	T1E1, T3
4	other	inward thru line if	T1E1, T3
5	inward	inward thru framer	T1E1, T3
6	dual	inward and outward	T1E1, T3
16	tulip	inward thru Tulip chip	all cards
17	pins	inward thru drvrs/rcvrs	SSI
18	LA/LL	assert LA/LL modem pin	HSSI, SSI
19	LB/RL	assert LB/RL modem pin	HSSI, SSI

-m Read Tulip MII registers. Print the 32 16-bit registers in the Media Independent Interface.

-M *addr data*
Write Tulip MII register. Write *data* into register *addr*.

-p Read Tulip PCI configuration registers. Print the first 16 32-bit registers in the PCI configuration space.

-P *addr data*
Write Tulip PCI configuration register. Write *data* into register *addr*.

-s Read Tulip SROM. Print the 64 16-bit locations. The PCI subsystem vendor and device IDs are kept here.

-S *number* Write Tulip SROM. Initializes the Tulip SROM to card type *number*.

3	HSSI
4	T3
5	SSI
6	T1E1
7	HSSIc
8	SDSL
0	auto-set from uCode type

If *number* is zero, then the card type is computed from the gate array microcode version field in the MII PHYID register. *CAUTION*: if the SROM is incorrect, the card will be unusable! This command is *so* dangerous that **lmcconfig** must be edited and recompiled to enable it.

-t Read Tulip CSRs. Print the 16 32-bit Control and Status Registers.

-T *addr data*
Write Tulip CSR. Write *data* into register *addr*. Note that *addr* is a CSR number (0-15) not a byte offset into CSR space.

-u Reset event counters to zero. The driver counts events like packets in and out, errors, discards, etc. The time when the counters are reset is remembered.

-U Reset gate array microcode.

-v Set verbose mode: print more stuff.

-V Print the card configuration; see the **EXAMPLES** section.

-x *number* Set the line control protocol to *number*. Line control protocols are listed below along with the operating systems that implement them and the stacks that include them.

x	*Protocol*	*OpSys*	*Stack*
1	IPinHDLC	FNOBL	D--G-N
2	PPP	FNOBL	-SPGYN
3	CiscoHDLC	FNOBL	-SPGYN
4	FrameRelay	F--BL	-SPG-N
5	EthInHDLC	F---L	---G-N

OpSys: FreeBSD NetBSD OpenBSD BSD/OS Linux.
Stack: Driver SPPP P2P GenHDLC sYncPPP Netgraph.

-X *number* Set the line control protocol stack to *number*. Line control protocol stacks are listed below along with the operating systems that include them and the protocols that they implement.

X	*Stack*	*OpSys*	*Protocol*
1	Driver	FNOBL	I----
2	SPPP	FNO--	-PCF-
3	P2P	---B-	-PCF-
4	GenHDLC	----L	IPCFE
5	SyncPPP	----L	-PC--
6	Netgraph	F----	IPCFE

OpSys: FreeBSD NetBSD OpenBSD BSD/OS Linux.
Protocol: IPinHDLC PPP CiscoHDLC FrmRly EthInHDLC.

-y Disable SPPP/SyncPPP keep-alive packets,

-Y Enable SPPP/SyncPPP keep-alive packets.

-? Print help (usage message).

Parameters for T1E1 cards

The following parameters apply to the T1E1 card type:

-a y|a|b Stop sending alarm signal (see table below).

-A y|a|b Start sending alarm signal.

y	Yellow Alarm	varies with framing
a	Red Alarm	unframed all ones; aka AIS
b	Blue Alarm	unframed all ones

Red alarm, also known as AIS (Alarm Indication Signal), and Blue alarm are identical in T1.

-B *number* Send a Bit Oriented Protocol (BOP) message with code *number*. BOP codes are six bits.

-c *number* Set cable length to *number* meters (default: 10 meters). This is used to set receiver sensitivity and transmitter line build-out.

-d Print the status of the on-board T1 DSU/CSU; see the **EXAMPLES** section.

-e *number* Set the framing format to *number*.

9	T1-SF/AMI
27	T1-ESF/B8ZS (default)
0	E1-FAS
8	E1-FAS+CRC
16	E1-FAS+CAS
24	E1-FAS+CRC+CAS
32	E1-NO-framing

-E *number* Enable 64Kb time slots (TSs) for the T1E1 card. The *number* argument is a 32-bit hex number (default 0xFFFFFFFF). The LSB is TS0 and the MSB is TS31. TS0 and TS25-31 are ignored in T1 mode. TS0 and TS16 are determined by the framing format in E1 mode.

-f Read framer registers. Print the 512 8-bit registers in the framer chip.

-F *addr data*
Write framer register. Write *data* into register *addr*.

-g *number* Set receiver gain range to *number*.

0x24	Short	0 to 20 dB of equalized gain
0x2C	Medium	0 to 30 dB of equalized gain
0x34	Long	0 to 40 dB of equalized gain
0x3F	Extend	0 to 64 dB of equalized gain (wide open)
0xFF	Auto	auto-set based on cable length (default)

This sets the level at which *Loss-Of-Signal* is declared.

-i Send a *CSU loopback deactivate* inband command (T1 only).

-I Send a *CSU loopback activate* inband command (T1 only).

-l Send a *line loopback deactivate* BOP message (T1-ESF only).

-L Send a *line loopback activate* BOP message (T1-ESF only).

-p Send a *payload loopback deactivate* BOP message (T1-ESF only).

-P Send a *payload loopback activate* BOP message (T1-ESF only).

-s Print the status of the on-board DSU/CSU; see the **EXAMPLES** section.

-t Stop sending test pattern (see table below).

-T *number* Start sending test pattern *number*.

0	unframed X^11+X^9+1
1	unframed X^15+X^14+1
2	unframed X^20+X^17+1
3	unframed X^23+X^18+1
4	unframed X^11+X^9+1 with 7ZS
5	unframed X^15+X^14+1 with 7ZS
6	unframed X^20+X^17+1 with 14ZS (QRSS)
7	unframed X^23+X^18+1 with 14ZS
8	framed X^11+X^9+1
9	framed X^15+X^14+1
10	framed X^20+X^17+1
11	framed X^23+X^18+1
12	framed X^11+X^9+1 with 7ZS
13	framed X^15+X^14+1 with 7ZS
14	framed X^20+X^17+1 with 14ZS (QRSS)
15	framed X^23+X^18+1 with 14ZS

-u *number* Set transmit pulse shape to *number*.

0	T1 DSX 0 to 40 meters
2	T1 DSX 40 to 80 meters
4	T1 DSX 80 to 120 meters
6	T1 DSX 120 to 160 meters
8	T1 DSX 160 to 200 meters
10	E1 75-ohm coax pair
12	E1 120-ohm twisted pairs
14	T1 CSU 200 to 2000 meters; set LBO
255	auto-set based on cable length and framing format (default)

-U *number* Set transmit line build-out to *number*.

0	0 dB	FCC option A
16	7.5 dB	FCC option B
32	15 dB	FCC option C
48	22.5 dB	final span
255	auto-set based on cable length (default)	

This is only applicable if the pulse shape is T1-CSU.

-x Disable transmitter outputs.

-X Enable transmitter outputs.

Parameters for T3 cards

The following parameters apply to the T3 card type:

-a y|**a**|**b**|**i** Stop sending alarm signal (see table below).

-A y|**a**|**b**|**i** Start sending alarm signal.

y	Yellow Alarm	X-bits set to 0
a	Red Alarm	framed 1010... aka AIS
b	Blue Alarm	unframed all-ones

i	Idle signal	framed 11001100...

−B *number* Send a Far End Alarm and Control (FEAC) message with code *number*. FEAC codes are six bits.

−c *number* Set cable length to *number* meters (default: 10 meters). This is used to set receiver sensitivity and transmitter line build-out.

−d Print the status of the on-board T3 DSU; see the **EXAMPLES** section.

−e *number* Set the framing format to *number*.

100	T3-C-bit parity
101	T3-M13 format

−f Read framer registers. Print the 22 8-bit registers in the framer chip.

−F *addr data*
Write framer register. Write *data* into register *addr*.

−l Send a *line loopback deactivate* BOP message.

−L Send a *line loopback activate* BOP message.

−s Print the status of the on-board T3 DSU; see the **EXAMPLES** section.

−S *number* Set payload scrambler polynominal to *number*.

1	payload scrambler disabled
2	Xˆ43+1: DigitalLink and Kentrox
3	Xˆ20+Xˆ17+1 w/28ZS: Larscom

Payload scrambler polynomials are not standardized.

−V *number* Set transmit frequency offset to *number*. Some T3 cards can offset the transmitter frequency from 44.736 MHz. *Number* is in the range (0..4095); 2048 is zero offset; step size is about 3 Hz. A *number* is written to a Digital-Analog Converter (DAC) which connects to a Voltage Controlled Crystal Oscillator (VCXO).

Event Counters

The device driver counts many interesting events such as packets in and out, errors and discards. The table below lists the event counters and describes what they count.

Rx bytes	Bytes received in packets with good ending status.
Tx bytes	Bytes transmitted in packets with good ending status.
Rx packets	Packets received with good ending status.
Tx packets	Packets transmitted with good ending status.
Rx errors	Packets received with bad ending status.
Tx errors	Packets transmitted with bad ending status.
Rx drops	Packets received but discarded by software because the input queue was full or the link was down.
Rx missed	Packets that were missed by hardware because the receiver was enabled but had no DMA descriptors.

Tx drops	Packets presented for transmission but discarded by software because the output queue was full or the link was down.
Rx fifo overruns	Packets that started to arrive, but were aborted because the card was unable to DMA data to memory fast enough to prevent the receiver fifo from overflowing. This is reported in the ending status of DMA descriptors.
Rx overruns	Rx Fifo overruns reported by the Tulip chip in the Status CSR. The driver stops the receiver and restarts it to work around a potential hardware hangup.
Tx fifo underruns	Packets that started to transmit but were aborted because the card was unable to DMA data from memory fast enough to prevent the transmitter fifo from underflowing. This is reported in the ending status of DMA descriptors.
Tx underruns	Tx Fifo underruns reported by the Tulip chip in the Status CSR. The driver increases the transmitter threshold, requiring more bytes to be in the fifo before the transmitter is started.
Rx FDL pkts	Packets received on the T1 Facility Data Link.
Rx CRC	Cyclic Redundancy Checksum errors detected by the CRC-6 in T1 Extended SuperFrames (ESF) or the CRC-4 in E1 frames.
Rx line code	Line Coding Violation errors: Alternate Mark Inversion (AMI) errors for T1-SF, Bipolar 8-Zero Substitution (B8ZS) errors for T1-ESF, or High Density Bipolar with 3-Zero Substitution (HDB3) errors for E1 or Bipolar 3-Zero Substitution (B3ZS) errors for T3.
Rx F-bits	T1 or T3 bit errors in the frame alignment signal.
Rx FEBE	Far End Block Errors: T1 or T3 bit errors detected by the device at the far end of the link.
Rx P-parity	T3 bit errors detected by the hop-by-hop parity mechanism.
Rx C-parity	T3 bit errors detected by the end-to-end parity mechanism.
Rx M-bits	T3 bit errors in the multi-frame alignment signal.

If driver debug mode is enabled, more event counters are displayed.

Rx no bufs	Failure to allocate a replacement packet buffer for an incoming packet. The buffer allocation is retried later.
Tx no descs	Failure to allocate a DMA descriptor for an outgoing packet. The descriptor allocation is retried later.
Lock watch	The watchdog routine conflicted with an IOCTL syscall.
Lock intr	A CPU tried to enter the interrupt handler while another CPU was already inside. The second CPU simply walks away.
Spare1-4	Nameless events of interest to the device driver maintainer.

Transmit Speed

The hardware counts transmit clocks divided by 2048. The software computes “Tx speed” from this (see **EXAMPLES** below). The transmit clock is the bit rate of the circuit divided by two if the circuit is idle and divided by four if the circuit is carrying a packet. So an empty circuit reports a Tx speed equal to its bit rate, and a full circuit reports a Tx speed equal to half its bit rate.

This "bit rate" does not include circuit-level overhead bits (such as T1 or T3 frame bits) but does include HDLC stuff bits. An idle T1 circuit with a raw bit rate of 1544000 and a bit-rate-minus-overhead of 1536000 will report a "Tx speed" of ((1536000 bitand 4095) plus or minus 4096). Sometimes it will even get the correct answer of 1536000, and if the link is fully loaded it will report about 768000 bits/sec.

It is not a perfect bit rate meter (the circuit must be idle), but it is a useful circuit utilization meter if you know the circuit bit rate and do some arithmetic. Software recalculates Tx speed once a second; the measurement period has some jitter.

EXAMPLES

When "`lmc0`" is a T1E1 card, "`lmcconfig lmc0`" generates the following output:

```
Card name:              lmc0
Card type:              T1E1 (lmc1200)
Link status:            Up
Tx Speed:               1536000
Line Prot/Pkg:          PPP/P2P
CRC length:             16 bits
Tx Clk src:             Modem Rx Clk (loop timed)
Format-Frame/Code:      T1-ESF/B8ZS
TimeSlots [31-0]:       0x01FFFFFE
Cable length:           10 meters
Current time:           Wed Jan  4 05:35:10 2006
Cntrs reset:            Fri Dec 16 19:23:45 2005
Rx bytes:               176308259
Tx bytes:               35194717
Rx packets:             383162
Tx packets:             357792
```

When "`lmc0`" is a T1E1 card, "`lmcconfig lmc0 -1 -d`" generates the following output:

```
Format-Frame/Code:      T1-ESF/B8ZS
TimeSlots [31-0]:       0x01FFFFFE
Tx Clk src:             Modem Rx Clk (loop timed)
Tx Speed:               1536000
Tx pulse shape:         T1-DSX: 0 to 40 meters
Tx outputs:             Enabled
Line impedance:         100 ohms
Max line loss:          20.0 dB
Cur line loss:           0.0 dB
Invert data:            No
Line    loop:           No
Payload loop:           No
Framer  loop:           No
Analog  loop:           No
Tx AIS:                 No
Rx AIS:                 No
Tx BOP RAI:             No
Rx BOP RAI:             No
Rx LOS analog:          No
Rx LOS digital:         No
Rx LOF:                 No
Tx QRS:                 No
Rx QRS:                 No
```

```
LCV errors:             0
CRC errors:             0
Frame errors:           0
Sev Err Frms:           0
Change of Frm align:    0
Loss of Frame events:   0
SNMP Near-end performance data:
 LCV=0 LOS=0 FE=0 CRC=0 AIS=0 SEF=0 OOF=0  RAI=0
ANSI Far-end performance reports:
 SEQ=0 CRC=0 SE=0 FE=0 LV=0 SL=0 LB=0
 SEQ=1 CRC=0 SE=0 FE=0 LV=0 SL=0 LB=0
 SEQ=2 CRC=0 SE=0 FE=0 LV=0 SL=0 LB=0
 SEQ=3 CRC=0 SE=0 FE=0 LV=0 SL=0 LB=0
```

DIAGNOSTICS

Messages indicating the specified interface does not exist, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

`ioctl`(2), `lmc`(4), `ifconfig`(8), `ifnet`(9)

`http://www.sbei.com/`

HISTORY

This is a total rewrite of the program **`lmcctl`** by Michael Graff, Rob Braun and Andrew Stanley-Jones.

AUTHORS

David Boggs ⟨boggs@boggs.palo-alto.ca.us⟩

NAME

loadbsd — load and boot NetBSD/x68k kernel from Human68k

SYNOPSIS

loadbsd.x [**-hvV**] [**-abDs**] [**-r** *root_device*] *kernel_file*

DESCRIPTION

loadbsd is a program runs on Human68k. It loads and executes the specified NetBSD/x68k kernel.

The options (for **loadbsd** itself) are as follows:

-h Show help and exit.

-v Enable verbose mode.

-V Print version of **loadbsd** and exit.

The options for NetBSD kernel are as follows:

-a Auto (multi-user) boot. This disables **-s** flag.

-b Ask boot device during boot. Pass RB_ASKNAME boot flag to the kernel.

-d Use compiled-in rootdev. Pass RB_DFLTROOT boot flag to the kernel.

-D Enter kernel debugger. Pass RB_KDB boot flag to the kernel.

-r *root_device*
Specify boot device, which shall be mounted as root device. The default device is 'sd@0,0:a'. Note that the boot device name is *not* the same as that of NetBSD. See **BOOT DEVICE NAMES** below.

-s Single user boot. Pass RB_SINGLE boot flag to the kernel. This disables **-a** flag. This flag is set by default.

Although listed separately, the options may be in any order.

BOOT DEVICE NAMES

The format of boot device names is:

```
[/interface/]dev@unit[,lun][:partition]
```

interface	SCSI interface type. One of: 'spc@0', 'spc@1', 'mha@0'. If the dev is a SCSI device, and interface is omitted, the current boot interface is used.
dev	Device type. One of: 'fd' (floppy disk drive), 'sd' (SCSI disk), 'cd' (SCSI CD-ROM), 'md' (Memory disk).
unit	Device unit #. You must specify the target SCSI ID if dev is a SCSI device.
lun	SCSI LUN #. 0 is assumed if omitted.
partition	Partition letter of device. Partition 'a' is used if omitted.

FILES

/usr/mdec/loadbsd.x	You will find this program here.

SEE ALSO

reboot(2), boot(8)

HISTORY

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

BUGS

loadbsd reads the entire kernel image at once, and requires enough free area on the main memory.

NAME

local − Postfix local mail delivery

SYNOPSIS

local [generic Postfix daemon options]

DESCRIPTION

The **local**(8) daemon processes delivery requests from the Postfix queue manager to deliver mail to local recipients. Each delivery request specifies a queue file, a sender address, a domain or host to deliver to, and one or more recipients. This program expects to be run from the **master**(8) process manager.

The **local**(8) daemon updates queue files and marks recipients as finished, or it informs the queue manager that delivery should be tried again at a later time. Delivery status reports are sent to the **bounce**(8), **defer**(8) or **trace**(8) daemon as appropriate.

CASE FOLDING

All delivery decisions are made using the bare recipient name (i.e. the address localpart), folded to lower case. See also under ADDRESS EXTENSION below for a few exceptions.

SYSTEM-WIDE AND USER-LEVEL ALIASING

The system administrator can set up one or more system-wide **sendmail**-style alias databases. Users can have **sendmail**-style ˜/.**forward** files. Mail for *name* is delivered to the alias *name*, to destinations in ˜*name*/.**forward**, to the mailbox owned by the user *name*, or it is sent back as undeliverable.

The system administrator can specify a comma/space separated list of ˜/.**forward** like files through the **forward_path** configuration parameter. Upon delivery, the local delivery agent tries each pathname in the list until a file is found.

Delivery via ˜/.**forward** files is done with the privileges of the recipient. Thus, ˜/.**forward** like files must be readable by the recipient, and their parent directory needs to have "execute" permission for the recipient.

The **forward_path** parameter is subject to interpolation of **$user** (recipient username), **$home** (recipient home directory), **$shell** (recipient shell), **$recipient** (complete recipient address), **$extension** (recipient address extension), **$domain** (recipient domain), **$local** (entire recipient address localpart) and **$recipient_delimiter.** The forms *${name?value}* and *${name:value}* expand conditionally to *value* when *$name* is (is not) defined. Characters that may have special meaning to the shell or file system are replaced by underscores. The list of acceptable characters is specified with the **forward_expansion_filter** configuration parameter.

An alias or ˜/.**forward** file may list any combination of external commands, destination file names, **:include:** directives, or mail addresses. See **aliases**(5) for a precise description. Each line in a user's .**forward** file has the same syntax as the right-hand part of an alias.

When an address is found in its own alias expansion, delivery is made to the user instead. When a user is listed in the user's own ˜/.**forward** file, delivery is made to the user's mailbox instead. An empty ˜/.**forward** file means do not forward mail.

In order to prevent the mail system from using up unreasonable amounts of memory, input records read from **:include:** or from ˜/.**forward** files are broken up into chunks of length **line_length_limit**.

While expanding aliases, ˜/.**forward** files, and so on, the program attempts to avoid duplicate deliveries. The **duplicate_filter_limit** configuration parameter limits the number of remembered recipients.

MAIL FORWARDING

For the sake of reliability, forwarded mail is re-submitted as a new message, so that each recipient has a separate on-file delivery status record.

In order to stop mail forwarding loops early, the software adds an optional **Delivered-To:** header with the final envelope recipient address. If mail arrives for a recipient that is already listed in a **Delivered-To:** header, the message is bounced.

MAILBOX DELIVERY

The default per-user mailbox is a file in the UNIX mail spool directory (**/var/mail/***user* or **/var/spool/mail/***user*); the location can be specified with the **mail_spool_directory** configuration parameter. Specify a name ending in **/** for **qmail**-compatible **maildir** delivery.

Alternatively, the per-user mailbox can be a file in the user's home directory with a name specified via the **home_mailbox** configuration parameter. Specify a relative path name. Specify a name ending in **/** for **qmail**-compatible **maildir** delivery.

Mailbox delivery can be delegated to an external command specified with the **mailbox_command_maps** and **mailbox_command** configuration parameters. The command executes with the privileges of the recipient user (exceptions: secondary groups are not enabled; in case of delivery as root, the command executes with the privileges of **default_privs**).

Mailbox delivery can be delegated to alternative message transports specified in the **master.cf** file. The **mailbox_transport_maps** and **mailbox_transport** configuration parameters specify an optional message transport that is to be used for all local recipients, regardless of whether they are found in the UNIX passwd database. The **fallback_transport_maps** and **fallback_transport** parameters specify an optional message transport for recipients that are not found in the aliases(5) or UNIX passwd database.

In the case of UNIX-style mailbox delivery, the **local**(8) daemon prepends a "**From** *sender time_stamp*" envelope header to each message, prepends an **X-Original-To:** header with the recipient address as given to Postfix, prepends an optional **Delivered-To:** header with the final envelope recipient address, prepends a **Return-Path:** header with the envelope sender address, prepends a > character to lines beginning with "**From** ", and appends an empty line. The mailbox is locked for exclusive access while delivery is in progress. In case of problems, an attempt is made to truncate the mailbox to its original length.

In the case of **maildir** delivery, the local daemon prepends an optional **Delivered-To:** header with the final envelope recipient address, prepends an **X-Original-To:** header with the recipient address as given to Postfix, and prepends a **Return-Path:** header with the envelope sender address.

EXTERNAL COMMAND DELIVERY

The **allow_mail_to_commands** configuration parameter restricts delivery to external commands. The default setting (**alias, forward**) forbids command destinations in **:include:** files.

Optionally, the process working directory is changed to the path specified with **command_execution_directory** (Postfix 2.2 and later). Failure to change directory causes mail to be deferred.

The **command_execution_directory** parameter value is subject to interpolation of **$user** (recipient username), **$home** (recipient home directory), **$shell** (recipient shell), **$recipient** (complete recipient address), **$extension** (recipient address extension), **$domain** (recipient domain), **$local** (entire recipient address localpart) and **$recipient_delimiter.** The forms *${name?value}* and *${name:value}* expand conditionally to *value* when *$name* is (is not) defined. Characters that may have special meaning to the shell or file system are replaced by underscores. The list of acceptable characters is specified with the **execution_directory_expansion_filter** configuration parameter.

The command is executed directly where possible. Assistance by the shell (**/bin/sh** on UNIX systems) is used only when the command contains shell magic characters, or when the command invokes a shell built-in command.

A limited amount of command output (standard output and standard error) is captured for inclusion with non-delivery status reports. A command is forcibly terminated if it does not complete within

command_time_limit seconds. Command exit status codes are expected to follow the conventions defined in <**sysexits.h**>. Exit status 0 means normal successful completion.

Postfix version 2.3 and later support RFC 3463-style enhanced status codes. If a command terminates with a non-zero exit status, and the command output begins with an enhanced status code, this status code takes precedence over the non-zero exit status.

A limited amount of message context is exported via environment variables. Characters that may have special meaning to the shell are replaced by underscores. The list of acceptable characters is specified with the **command_expansion_filter** configuration parameter.

SHELL
The recipient user's login shell.

HOME
The recipient user's home directory.

USER The bare recipient name.

EXTENSION
The optional recipient address extension.

DOMAIN
The recipient address domain part.

LOGNAME
The bare recipient name.

LOCAL
The entire recipient address localpart (text to the left of the rightmost @ character).

RECIPIENT
The entire recipient address.

SENDER
The entire sender address.

Additional remote client information is made available via the following environment variables:

CLIENT_ADDRESS
Remote client network address. Available as of Postfix 2.2.

CLIENT_HELO
Remote client EHLO command parameter. Available as of Postfix 2.2.

CLIENT_HOSTNAME
Remote client hostname. Available as of Postfix 2.2.

CLIENT_PROTOCOL
Remote client protocol. Available as of Postfix 2.2.

SASL_METHOD
SASL authentication method specified in the remote client AUTH command. Available as of Postfix 2.2.

SASL_SENDER
SASL sender address specified in the remote client MAIL FROM command. Available as of Postfix 2.2.

SASL_USERNAME
SASL username specified in the remote client AUTH command. Available as of Postfix 2.2.

The **PATH** environment variable is always reset to a system-dependent default path, and environment variables whose names are blessed by the **export_environment** configuration parameter are exported unchanged.

The current working directory is the mail queue directory.

The **local**(8) daemon prepends a "**From** *sender time_stamp*" envelope header to each message, prepends an **X-Original-To:** header with the recipient address as given to Postfix, prepends an optional **Delivered-To:** header with the final recipient envelope address, prepends a **Return-Path:** header with the sender envelope address, and appends no empty line.

EXTERNAL FILE DELIVERY

The delivery format depends on the destination filename syntax. The default is to use UNIX-style mailbox format. Specify a name ending in **/** for **qmail**-compatible **maildir** delivery.

The **allow_mail_to_files** configuration parameter restricts delivery to external files. The default setting (**alias, forward**) forbids file destinations in **:include:** files.

In the case of UNIX-style mailbox delivery, the **local**(8) daemon prepends a "**From** *sender time_stamp*" envelope header to each message, prepends an **X-Original-To:** header with the recipient address as given to Postfix, prepends an optional **Delivered-To:** header with the final recipient envelope address, prepends a **>** character to lines beginning with "**From** ", and appends an empty line. The envelope sender address is available in the **Return-Path:** header. When the destination is a regular file, it is locked for exclusive access while delivery is in progress. In case of problems, an attempt is made to truncate a regular file to its original length.

In the case of **maildir** delivery, the local daemon prepends an optional **Delivered-To:** header with the final envelope recipient address, and prepends an **X-Original-To:** header with the recipient address as given to Postfix. The envelope sender address is available in the **Return-Path:** header.

ADDRESS EXTENSION

The optional **recipient_delimiter** configuration parameter specifies how to separate address extensions from local recipient names.

For example, with "**recipient_delimiter = +**", mail for *name*+*foo* is delivered to the alias *name*+*foo* or to the alias *name*, to the destinations listed in ˜*name*/.**forward**+*foo* or in ˜*name*/.**forward**, to the mailbox owned by the user *name*, or it is sent back as undeliverable.

In all cases the **local**(8) daemon prepends an optional ‘**Delivered-To:** header line with the final recipient address.

DELIVERY RIGHTS

Deliveries to external files and external commands are made with the rights of the receiving user on whose behalf the delivery is made. In the absence of a user context, the **local**(8) daemon uses the owner rights of the **:include:** file or alias database. When those files are owned by the superuser, delivery is made with the rights specified with the **default_privs** configuration parameter.

STANDARDS

RFC 822 (ARPA Internet Text Messages)
RFC 3463 (Enhanced status codes)

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8). Corrupted message files are marked so that the queue manager can move them to the **corrupt** queue afterwards.

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces and of other trouble.

SECURITY

The **local**(8) delivery agent needs a dual personality 1) to access the private Postfix queue and IPC mechanisms, 2) to impersonate the recipient and deliver to recipient-specified files or commands. It is therefore security sensitive.

The **local**(8) delivery agent disallows regular expression substitution of $1 etc. in **alias_maps**, because that would open a security hole.

The **local**(8) delivery agent will silently ignore requests to use the **proxymap**(8) server within **alias_maps**. Instead it will open the table directly. Before Postfix version 2.2, the **local**(8) delivery agent will terminate with a fatal error.

BUGS

For security reasons, the message delivery status of external commands or of external files is never checkpointed to file. As a result, the program may occasionally deliver more than once to a command or external file. Better safe than sorry.

Mutually-recursive aliases or ˜/.**forward** files are not detected early. The resulting mail forwarding loop is broken by the use of the **Delivered-To:** message header.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically, as **local**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

COMPATIBILITY CONTROLS

biff (yes)
Whether or not to use the local biff service.

expand_owner_alias (no)
When delivering to an alias "aliasname" that has an "owner-aliasname" companion alias, set the envelope sender address to the expansion of the "owner-aliasname" alias.

owner_request_special (yes)
Give special treatment to owner-listname and listname-request address localparts: don't split such addresses when the recipient_delimiter is set to "-".

sun_mailtool_compatibility (no)
Obsolete SUN mailtool compatibility feature.

Available in Postfix version 2.3 and later:

frozen_delivered_to (yes)
Update the **local**(8) delivery agent's idea of the Delivered-To: address (see prepend_delivered_header) only once, at the start of a delivery attempt; do not update the Delivered-To: address while expanding aliases or .forward files.

DELIVERY METHOD CONTROLS

The precedence of **local**(8) delivery methods from high to low is: aliases, .forward files, mailbox_transport_maps, mailbox_transport, mailbox_command_maps, mailbox_command, home_mailbox, mail_spool_directory, fallback_transport_maps, fallback_transport, and luser_relay.

alias_maps (see 'postconf -d' output)
The alias databases that are used for **local**(8) delivery.

forward_path (see 'postconf -d' output)
The **local**(8) delivery agent search list for finding a .forward file with user-specified delivery methods.

mailbox_transport_maps (empty)
Optional lookup tables with per-recipient message delivery transports to use for **local**(8) mailbox delivery, whether or not the recipients are found in the UNIX passwd database.

mailbox_transport (empty)
Optional message delivery transport that the **local**(8) delivery agent should use for mailbox delivery to all local recipients, whether or not they are found in the UNIX passwd database.

mailbox_command_maps (empty)
Optional lookup tables with per-recipient external commands to use for **local**(8) mailbox delivery.

mailbox_command (empty)
Optional external command that the **local**(8) delivery agent should use for mailbox delivery.

home_mailbox (empty)
Optional pathname of a mailbox file relative to a **local**(8) user's home directory.

mail_spool_directory (see 'postconf -d' output)
The directory where **local**(8) UNIX-style mailboxes are kept.

fallback_transport_maps (empty)
Optional lookup tables with per-recipient message delivery transports for recipients that the **local**(8) delivery agent could not find in the **aliases**(5) or UNIX password database.

fallback_transport (empty)
Optional message delivery transport that the **local**(8) delivery agent should use for names that are not found in the **aliases**(5) or UNIX password database.

luser_relay (empty)
Optional catch-all destination for unknown **local**(8) recipients.

Available in Postfix version 2.2 and later:

command_execution_directory (empty)
The **local**(8) delivery agent working directory for delivery to external command.

MAILBOX 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

command_time_limit (1000s)
Time limit for delivery to external commands.

duplicate_filter_limit (1000)
The maximal number of addresses remembered by the address duplicate filter for **aliases**(5) or **virtual**(5) alias expansion, or for **showq**(8) queue displays.

local_destination_concurrency_limit (2)
The maximal number of parallel deliveries via the local mail delivery transport to the same recipient (when "local_destination_recipient_limit = 1") or the maximal number of parallel deliveries to the same local domain (when "local_destination_recipient_limit > 1").

local_destination_recipient_limit (1)
The maximal number of recipients per message delivery via the local mail delivery transport.

mailbox_size_limit (51200000)
The maximal size of any **local**(8) individual mailbox or maildir file, or zero (no limit).

SECURITY CONTROLS

allow_mail_to_commands (alias, forward)
Restrict **local**(8) mail delivery to external commands.

allow_mail_to_files (alias, forward)
Restrict **local**(8) mail delivery to external files.

command_expansion_filter (see 'postconf -d' output)
Restrict the characters that the **local**(8) delivery agent allows in $name expansions of $mailbox_command.

default_privs (nobody)
The default rights used by the **local**(8) delivery agent for delivery to external file or command.

forward_expansion_filter (see 'postconf -d' output)
Restrict the characters that the **local**(8) delivery agent allows in $name expansions of $forward_path.

Available in Postfix version 2.2 and later:

execution_directory_expansion_filter (see 'postconf -d' output)
Restrict the characters that the **local**(8) delivery agent allows in $name expansions of $command_execution_directory.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

export_environment (see 'postconf -d' output)
The list of environment variables that a Postfix process will export to non-Postfix processes.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

local_command_shell (empty)
Optional shell program for **local**(8) delivery to non-Postfix command.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

prepend_delivered_header (command, file, forward)
The message delivery contexts where the Postfix **local**(8) delivery agent prepends a Delivered-To: message header with the address that the mail was delivered to.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

propagate_unmatched_extensions (canonical, virtual)
What address lookup tables copy an address extension from the lookup key to the lookup result.

queue_directory (see 'postconf -d' output)
The location of the Postfix top-level queue directory.

recipient_delimiter (empty)
The separator between user names and address extensions (user+foo).

require_home_directory (no)
Whether or not a **local**(8) recipient's home directory must exist before mail delivery is attempted.

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

The following are examples; details differ between systems.
$HOME/.forward, per-user aliasing
/etc/aliases, system-wide alias database
/var/spool/mail, system mailboxes

SEE ALSO

qmgr(8), queue manager
bounce(8), delivery status reports
newaliases(1), create/update alias database
postalias(1), create/update alias database
aliases(5), format of alias database
postconf(5), configuration parameters
master(5), generic daemon options
syslogd(8), system logging

LICENSE

HISTORY

The **Delivered-To:** message header appears in the **qmail** system by Daniel Bernstein.

The *maildir* structure appears in the **qmail** system by Daniel Bernstein.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

locate.updatedb — update locate database

SYNOPSIS

/usr/libexec/locate.updatedb

DESCRIPTION

The **locate.updatedb** program rebuilds the database used by the `locate`(1) program. It is usually run once per week, see `weekly.conf`(5).

The file systems and files (not) scanned can be configured in `locate.conf`(5).

FILES

`/var/db/locate.database`	Default database

SEE ALSO

`find`(1), `locate`(1), `fnmatch`(3), `locate.conf`(5), `weekly.conf`(5)

Woods, James A., "Finding Files Fast", *;login*, 8:1, pp. 8-10, 1983.

HISTORY

The **locate.updatedb** command appeared in 4.4BSD.

NAME

lockstat — display kernel locking statistics

SYNOPSIS

lockstat [**−ceflMmpst**] [**−b** *nbuf*] [**−E** *event*] [**−F** *func*] [**−L** *lock*] [**−N** *nlist*] [**−o** *file*] [**−T** *type*] *command ...*

DESCRIPTION

The **lockstat** command enables system wide tracing of kernel lock events, executes the specified command, and when finished reports statistics to the user.

Tracing may be ended early by sending SIGINT (Ctrl-C) to the process being executed by lockstat.

The **lockstat** pseudo-device driver must be present in the kernel, and the **lockstat** command may only be used by the root user.

The options are as follows:

−b *nbuf*	Adjust the number of trace buffers allocated by the kernel to *nbuf*.
−c	Report percentage of total events by count, and sort the output by number of events. The default is to key on event timings.
−E *event*	Limit tracing to one type of event. Use the **−e** option to list valid events.
−e	List valid event types for the **−E** option and exit.
−F *func*	Limit tracing to locking operations performed within the specified function. *func* must be the name of a valid function in the kernel.
−f	Trace only by calling functions; do not report on individual locks.
−L *lock*	Limit tracing to one lock. *lock* may either be the name of a lock object in the kernel, or a kernel virtual address.
−l	Trace only by lock; do not report on calling functions.
−M	Merge lock addresses within unique objects.
−m	Merge call sites within unique functions.
−N *nlist*	Extract symbol information from the *nlist* file.
−o *file*	Send output to the file named by *file*, instead of the standard output (the default).
−p	Show the average number of events and time spent per CPU. The default is to show the total values. May be used in conjunction with the **−s** option.
−s	Show the average number of events per second, and the average time spent per second. The default is to show the total values.
−T *type*	Limit tracing to one type of lock. Use the **−t** option to list valid lock types.
−t	List valid lock types for the **−T** option and exit.

FILES

/dev/lockstat	**lockstat** control device
/dev/ksyms	default namelist
/netbsd	namelist

EXAMPLES

```
# lockstat -T kernel_lock sleep 10
Elapsed time: 10.01 seconds.

-- Kernel lock spin

Total%  Count   Time/ms          Lock                       Caller
------ ------- --------- -------------------- ------------------------------
100.00   74941   1545.54 kernel_lock          <all>
 43.54   28467    673.00 kernel_lock          trap+71e
 42.87   34466    662.51 kernel_lock          syscall_plain+111
  7.38    7565    114.14 kernel_lock          uiomove+17a
  1.92    1221     29.61 kernel_lock          sleepq_block+20b
  1.84    1759     28.40 kernel_lock          trap+706
  0.81     124     12.54 kernel_lock          x86_softintlock+1a
  0.64     587      9.87 kernel_lock          pmap_load+2a6
  0.52     214      8.10 kernel_lock          intr_biglock_wrapper+1e
  0.20     219      3.09 kernel_lock          pmap_load+323
  0.14     175      2.12 kernel_lock          do_sys_wait+2d0
  0.09      85      1.43 kernel_lock          lwp_startup+66
  0.04      49      0.64 kernel_lock          sleepq_block+18c
  0.01      10      0.12 kernel_lock          lwp_userret+3c
```

DIAGNOSTICS

lockstat: incompatible lockstat interface version

The kernel device driver does not match the version of the **lockstat** command.

lockstat: overflowed available kernel trace buffers

Increase the number of buffers using the **-b** option.

lockstat: ioctl: Invalid argument

The number of trace buffers is outside the minimum and maximum bounds set by the kernel.

SEE ALSO

ps(1), systat(1), vmstat(1), iostat(8), pstat(8)

HISTORY

The **lockstat** command appeared in NetBSD 4.0.

NAME

lpc — line printer control program

SYNOPSIS

lpc [*command* [*argument ...*]]

DESCRIPTION

lpc is used by the system administrator to control the operation of the line printer system. For each line printer configured in `/etc/printcap`, **lpc** may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer daemons.

Without any arguments, **lpc** will prompt for commands from the standard input. If arguments are supplied, **lpc** interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing **lpc** to read commands from file. Commands may be abbreviated; the following is the list of recognized commands.

? [command ...]
help [command ...]
Print a short description of each command specified in the argument list, or, if no argument is given, a list of the recognized commands.

abort { all | printer }
Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by `lpr`(1)) for the specified printers.

clean { all | printer }
Remove any temporary files, data files, and control files that cannot be printed (i.e., do not form a complete printer job) from the specified printer queue(s) on the local machine.

disable { all | printer }
Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by `lpr`(1).

down { all | printer } message ...
Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message doesn't need to be quoted, the remaining arguments are treated like `echo`(1). This is normally used to take a printer down and let others know why `lpq`(1) will indicate the printer is down and print the status message).

enable { all | printer }
Enable spooling on the local queue for the listed printers. This will allow `lpr`(1) to put new jobs in the spool queue.

exit
quit Exit from lpc.

restart { all | printer }
Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly, leaving jobs in the queue. `lpq`(1) will report that there is no daemon present when this condition occurs. If the user is the super-user, try to abort the current daemon first (i.e., kill and restart a stuck daemon).

start { all | printer }
Enable printing and start a spooling daemon for the listed printers.

status { all | printer }
Display the status of daemons and queues on the local machine.

stop { all | printer }
Stop a spooling daemon after the current job completes and disable printing.

topq printer [jobnum ...] [user ...]
Place the jobs in the order listed at the top of the printer queue.

up { all | printer }
Enable everything and start a new printer daemon. Undoes the effects of **down**.

FILES

`/etc/printcap`	printer description file
`/var/spool/output/*`	spool directories
`/var/spool/output/*/lock`	lock file for queue control

DIAGNOSTICS

?Ambiguous command
abbreviation matches more than one command

?Invalid command
no match was found

?Privileged command
you must be a member of group "operator" or root to execute this command

SEE ALSO

`lpq`(1), `lpr`(1), `lprm`(1), `printcap`(5), `lpd`(8)

HISTORY

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

NAME

lpd — line printer spooler daemon

SYNOPSIS

lpd [**-dlsrW**] [**-b** *bind-address*] [**-n** *maxchild*] [**-w** *maxwait*] [port]

DESCRIPTION

lpd is the line printer daemon (spool area handler) and is normally invoked at boot time from the `rc`(8) file. It makes a single pass through the `printcap`(5) file to find out about the existing printers and prints any files left after a crash. It then uses the system calls `listen`(2) and `accept`(2) to receive requests to print files in the queue, transfer files to the spooling area, display the queue, or remove jobs from the queue. In each case, it forks a child to handle the request so the parent can continue to listen for more requests.

Available options:

-b Normally, if the **-s** option is not specified, **lpd** will listen on all network interfaces for incoming TCP connections. The **-b** option, followed by a *bind-address* specifies that **lpd** should listen on that address instead of INADDR_ANY. Multiple **-b** options are permitted, allowing a list of addresses to be specified. Use of this option silently overrides the **-s** option if it is also present on the command line. *bind-address* can be a numeric host name in IPv4 or IPv6 notation, or a symbolic host name which will be looked up in the normal way.

-d The **-d** option turns on the `SO_DEBUG` `socket`(2) option. See `setsockopt`(2) for more details.

-l The **-l** flag causes **lpd** to log valid requests received from the network. This can be useful for debugging purposes.

-n The **-n** flag sets *maxchild* as the maximum number of child processes that **lpd** will spawn. The default is 32.

-r The **-r** flag allows the "of" and "if" filters to be used if specified for a remote printer. Traditionally, **lpd** would not use filters for remote printers.

-s The **-s** flag selects "secure" mode, in which **lpd** does not listen on a TCP socket but only takes commands from a UNIX domain socket. This is valuable when the machine on which **lpd** runs is subject to attack over the network and it is desired that the machine be protected from attempts to remotely fill spools and similar attacks.

-w The **-w** flag sets *maxwait* as the wait time (in seconds) for dead remote server detection. If no response is returned from a connected server within this period, the connection is closed and a message logged. The default is 120 seconds.

-W The **-W** option will instruct lpd not to verify a remote tcp connection comes from a reserved port (<1024).

If the [port] parameter is passed, **lpd** listens on this port instead of the usual "printer/tcp" port from `/etc/services`.

Access control is provided by three means. First, `/etc/hosts.allow` and `/etc/hosts.deny` are consulted as described in `hosts_access`(5) with daemon name **lpd**. Second, all requests must come from one of the machines listed in the file `/etc/hosts.equiv` or `/etc/hosts.lpd`. Lastly, if the `rs` capability is specified in the `printcap`(5) entry for the printer being accessed, *lpr* requests will only be honored for those users with accounts on the machine with the printer. Requests must pass all three tests.

The file *minfree* in each spool directory contains the number of disk blocks to leave free so that the line printer queue won't completely fill the disk. The *minfree* file can be edited with your favorite text editor.

The daemon begins processing files after it has successfully set the lock for exclusive access (described a bit later), and scans the spool directory for files beginning with *cf*. Lines in each *cf* file specify files to be printed or non-printing actions to be performed. Each such line begins with a key character to specify what to do with the remainder of the line.

J	Job Name. String to be used for the job name on the burst page.
C	Classification. String to be used for the classification line on the burst page.
L	Literal. The line contains identification info from the password file and causes the banner page to be printed.
T	Title. String to be used as the title for `pr`(1).
H	Host Name. Name of the machine where `lpr`(1) was invoked.
P	Person. Login name of the person who invoked `lpr`(1). This is used to verify ownership by `lprm`(1).
M	Send mail to the specified user when the current print job completes.
f	Formatted File. Name of a file to print which is already formatted.
l	Like "f" but passes control characters and does not make page breaks.
p	Name of a file to print using `pr`(1) as a filter.
t	Troff File. The file contains `troff`(1) output (cat phototypesetter commands).
n	Ditroff File. The file contains device independent troff output.
d	DVI File. The file contains Tex l output DVI format from Stanford.
g	Graph File. The file contains data produced by **plot**.
c	Cifplot File. The file contains data produced by **cifplot**.
v	The file contains a raster image.
o	The file contains PostScript data.
r	The file contains text data with FORTRAN carriage control characters.
1	Troff Font R. Name of the font file to use instead of the default.
2	Troff Font I. Name of the font file to use instead of the default.
3	Troff Font B. Name of the font file to use instead of the default.
4	Troff Font S. Name of the font file to use instead of the default.
W	Width. Changes the page width (in characters) used by `pr`(1) and the text filters.
I	Indent. The number of characters to indent the output by (in ascii).
U	Unlink. Name of file to remove upon completion of printing.
N	File name. The name of the file which is being printed, or a blank for the standard input (when `lpr`(1) is invoked in a pipeline).

If a file cannot be opened, a message will be logged via `syslog`(3) using the *LOG_LPR* facility. **lpd** will try up to 20 times to reopen a file it expects to be there, after which it will skip the file to be printed.

lpd uses `flock`(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated to reflect the current status of **lpd** for the programs lpq(1) and lprm(1).

FILES

/etc/printcap	printer description file
/var/spool/output/*	spool directories
/var/spool/output/*/minfree	minimum free space to leave
/dev/lp*	line printer devices
/var/run/printer	socket for local requests
/etc/hosts.allow	explicit remote host access list.
/etc/hosts.deny	explicit remote host denial of service list.
/etc/hosts.equiv	lists machine names allowed printer access
/etc/hosts.lpd	lists machine names allowed printer access, but not under same administrative control.

SEE ALSO

lpq(1), lpr(1), lprm(1), setsockopt(2), syslog(3), hosts.equiv(5), hosts_access(5), hosts_options(5), printcap(5), lpc(8), pac(8)

4.3 BSD Line Printer Spooler Manual.

HISTORY

An **lpd** daemon appeared in Version 6 AT&T UNIX.

NAME

lptctl — manipulate lpt devices

SYNOPSIS

lptctl *device* [*command* [*...*]]

DESCRIPTION

lptctl is used to manipulate lpt devices so that a user can change how a printer that is attached to a parallel port works. If no command-argument pairs are specified, the status of the device is printed to standard output. The status information is also output after commands are carried out.

DEVICE COMMANDS

Multiple command-argument pairs can be specified. Available commands are:

mode *standard* | *nibble* | *ps2* | *fast* | *ecp* | *epp*

Sets port to use a mode of operation: standard centronics mode (standard), nibble mode, bidirectional mode (ps2), fast centronics mode (fast), enhanced capabilities port mode (ecp), or enhanced parallel port mode (epp).

dma *yes* | *no*

Enables or disables DMA. Note DMA is only used in some advanced modes such as ECP, and only if interrupts are enabled too.

ieee *yes* | *no*

Enables or disables the use of IEEE 1284 negotiations during mode changes and channel direction changes.

intr *yes* | *no*

Enables or disables use of interrupts for port operation. If interrupts are disabled, polling is used for data transfers. Default setting depends on device, but polling is commonly preferred.

prime *yes* | *no*

Enables (default) or disables printer initialization sequence on device open. Note the initialization sequence is never done on open of control lpt device.

autolf *yes* | *no*

Enables or disables (default) automatic LF on CR for data transfers.

EXIT STATUS

lptctl returns 0 on success, >0 on failure.

FILES

/dev/lpt?ctl - printer port control devices

/dev/lpt? - printer ports

SEE ALSO

ioctl(2), lpt(4), ppbus(4)

HISTORY

A similar utility called **lptcontrol** exists in FreeBSD to control the lpt(4) device. While similar in concept, the implementations are independent. **lptctl** was added in NetBSD 2.0.

AUTHORS

This man page and the **lptctl** utility were written by Gary Thorpe.

NAME

lwresd − lightweight resolver daemon

SYNOPSIS

lwresd [**−C** *config−file*] [**−d** *debug−level*] [**−f**] [**−g**] [**−i** *pid−file*] [**−n** *#cpus*] [**−P** *port*] [**−p** *port*] [**−s**] [**−t** *directory*] [**−u** *user*] [**−v**]

DESCRIPTION

lwresd is the daemon providing name lookup services to clients that use the BIND 9 lightweight resolver library. It is essentially a stripped−down, caching−only name server that answers queries using the BIND 9 lightweight resolver protocol rather than the DNS protocol.

lwresd listens for resolver queries on a UDP port on the IPv4 loopback interface, 127.0.0.1. This means that **lwresd** can only be used by processes running on the local machine. By default UDP port number 921 is used for lightweight resolver requests and responses.

Incoming lightweight resolver requests are decoded by the server which then resolves them using the DNS protocol. When the DNS lookup completes, **lwresd** encodes the answers in the lightweight resolver format and returns them to the client that made the request.

If */etc/resolv.conf* contains any **nameserver** entries, **lwresd** sends recursive DNS queries to those servers. This is similar to the use of forwarders in a caching name server. If no **nameserver** entries are present, or if forwarding fails, **lwresd** resolves the queries autonomously starting at the root name servers, using a built−in list of root server hints.

OPTIONS

−C *config−file*
: Use *config−file* as the configuration file instead of the default, */etc/resolv.conf*.

−d *debug−level*
: Set the daemon’s debug level to *debug−level*. Debugging traces from **lwresd** become more verbose as the debug level increases.

−f
: Run the server in the foreground (i.e. do not daemonize).

−g
: Run the server in the foreground and force all logging to *stderr*.

−n *#cpus*
: Create *#cpus* worker threads to take advantage of multiple CPUs. If not specified, **lwresd** will try to determine the number of CPUs present and create one thread per CPU. If it is unable to determine the number of CPUs, a single worker thread will be created.

−P *port*
: Listen for lightweight resolver queries on port *port*. If not specified, the default is port 921.

−p *port*
: Send DNS lookups to port *port*. If not specified, the default is port 53. This provides a way of testing the lightweight resolver daemon with a name server that listens for queries on a non−standard port number.

−s
: Write memory usage statistics to *stdout* on exit.

    Note: This option is mainly of interest to BIND 9 developers and may be removed or changed in a future release.

−t *directory*
: **chroot()** to *directory* after processing the command line arguments, but before reading the configuration file.

    Warning: This option should be used in conjunction with the **−u** option, as chrooting a process running as root doesn’t enhance security on most systems; the way **chroot()** is defined allows a process with root privileges to escape a chroot jail.

−u *user*

: **setuid()** to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

−v

: Report the version number and exit.

FILES

/etc/resolv.conf

: The default configuration file.

/var/run/lwresd.pid

: The default process−id file.

SEE ALSO

named(8), **lwres**(3), **resolver**(5).

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

mail.local — store mail in a mailbox

SYNOPSIS

mail.local [**-l**] [**-f** *from*] *user ...*

DESCRIPTION

mail.local reads the standard input up to an end-of-file and appends it to each *user's* mail file. The *user* must be a valid user name.

The options are as follows:

-f *from*
: Specify the sender's name.

-l Request that **username.lock** files be used for locking.

Individual mail messages in the mailbox are delimited by an empty line followed by a line beginning with the string ‘‘From ’’. A line containing the string ‘‘From ’’, the sender's name and a time stamp is prepended to each delivered mail message. A blank line is appended to each message. A greater-than character (‘‘>’’) is prepended to any line in the message which could be mistaken for a ‘‘From ’’ delimiter line.

If the [**-l**] flag is specified mailbox locking is done with **username.lock** files. Otherwise, the mailbox is exclusively locked with flock(2) while mail is appended.

If the ‘‘biff’’ service is returned by getservbyname(3), the biff server is notified of delivered mail.

The **mail.local** utility exits 0 on success, and >0 if an error occurs.

ENVIRONMENT

TZ Used to set the appropriate time zone on the timestamp.

FILES

/tmp/local.XXXXXX	temporary files
/var/mail/user	user's mailbox directory

SEE ALSO

mail(1), flock(2), getservbyname(3), comsat(8), sendmail(8)

HISTORY

A superset of **mail.local** (handling mailbox reading as well as mail delivery) appeared in Version 7 AT&T UNIX as the program mail(1).

NAME

mailwrapper — invoke appropriate MTA software based on configuration file

SYNOPSIS

Special. See below.

DESCRIPTION

At one time, the only Mail Transfer Agent (MTA) software easily available was `sendmail`(8). As a result of this, most Mail User Agents (MUAs) such as `mail`(1) had the path and calling conventions expected by `sendmail`(8) compiled in.

Times have changed, however. On a modern NetBSD system, the administrator may wish to use one of several available MTAs.

It would be difficult to modify all MUA software typically available on a system, so most of the authors of alternative MTAs have written their front end message submission programs so that they use the same calling conventions as `sendmail`(8) and may be put into place instead of `sendmail`(8) in `/usr/sbin/sendmail`.

`sendmail`(8) also typically has aliases named `mailq`(1) and `newaliases`(1) linked to it. The program knows to behave differently when its *argv[0]* is "mailq" or "newaliases" and behaves appropriately. Typically, replacement MTAs provide similar functionality, either through a program that also switches behavior based on calling name, or through a set of programs that provide similar functionality.

Although having replacement programs that plug replace `sendmail`(8) helps in installing alternative MTAs, it essentially makes the configuration of the system depend on hand installing new programs in `/usr`. This leads to configuration problems for many administrators, since they may wish to install a new MTA without altering the system provided `/usr`. (This may be, for example, to avoid having upgrade problems when a new version of the system is installed over the old.) They may also have a shared `/usr` among several machines, and may wish to avoid placing implicit configuration information in a read-only `/usr`.

The **mailwrapper** program is designed to replace `/usr/sbin/sendmail` and to invoke an appropriate MTA instead of `sendmail`(8) based on configuration information placed in `/etc/mailer.conf`. This permits the administrator to configure which MTA is to be invoked on the system at run time.

EXIT STATUS

mailwrapper exits 0 on success, and >0 if an error occurs.

FILES

Configuration for **mailwrapper** is kept in `/etc/mailer.conf`. `/usr/sbin/sendmail` is typically set up as a symlink to **mailwrapper** which is not usually invoked on its own.

DIAGNOSTICS

mailwrapper will print a diagnostic if its configuration file is missing or malformed, or does not contain a mapping for the name under which it was invoked.

SEE ALSO

`mail`(1), `mailq`(1), `newaliases`(1), `mailer.conf`(5), `sendmail`(8)

HISTORY

mailwrapper appeared in NetBSD 1.4.

AUTHORS

Perry E. Metzger ⟨perry@piermont.com⟩

BUGS

The entire reason this program exists is a crock. Instead, a command for how to submit mail should be standardized, and all the "behave differently if invoked with a different name" behavior of things like `mailq`(1) should go away.

NAME

makedbm — create a NIS database

SYNOPSIS

makedbm **-u** *dbfile*
makedbm [**-bls**] [**-d** *yp_domain_name*] [**-i** *yp_input_file*] [**-m** *yp_master_name*] [**-o** *yp_output_file*] *infile* *outfile*

DESCRIPTION

makedbm is the utility in NIS that creates the db(3) database file containing the NIS map.

infile is the pathname of the source file (where "-" is standard input). Each line consists of the key and the value, with a space separating the items. Blank lines are ignored, and a "#" is a comment character and indicates that the rest of the line should be ignored.

outfile is the pathname of the generated database.

The options are as follows:

-b Interdomain. Include an entry in the database informing a NIS server to use DNS to get information about unknown hosts. This option will only have effect on the maps hosts.byname and hosts.byaddr.

-l Lowercase. Convert all keys to lower case before adding them to the NIS database.

-s Secure map. Include an entry in the database informing ypxfr(8) and ypserv(8) that the NIS map is going to be handled as secure (i.e., not served to clients that don't connect from a reserved port).

-d *yp_domain_name*
Include an entry in the map with 'YP_DOMAIN_NAME' as the key and *yp_domain_name* as the value.

-i *yp_input_file*
Include an entry in the map with 'YP_INPUT_FILE' as the key and *yp_input_file* as the value.

-m *yp_master_name*
Include an entry in the map with 'YP_MASTER_NAME' as the key and *yp_master_name* as the value.

-o *yp_output_file*
Include an entry in the map with 'YP_OUTPUT_FILE' as the key and *yp_output_file* as the value.

-u *dbfile*
Dump the contents of *dbfile* to standard output, in a format suitable to be passed back into **makedbm**. *dbfile* is the pathname to the database.

SEE ALSO

db(3), nis(8), ypserv(8), ypxfr(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

makefs — create a file system image from a directory tree

SYNOPSIS

makefs [**-x**] [**-B** *byte-order*] [**-b** *free-blocks*] [**-d** *debug-mask*] [**-F** *specfile*] [**-f** *free-files*] [**-M** *minimum-size*] [**-m** *maximum-size*] [**-N** *userdb-dir*] [**-o** *fs-options*] [**-S** *sector-size*] [**-s** *image-size*] [**-t** *fs-type*] *image-file directory*

DESCRIPTION

The utility **makefs** creates a file system image into *image-file* from the directory tree *directory*. No special devices or privileges are required to perform this task.

The options are as follows:

-B *byte-order*
Set the byte order of the image to *byte-order*. Valid byte orders are `4321`, `big`, or '`be`' for big endian, and `1234`, `little`, or '`le`' for little endian. Some file systems may have a fixed byte order; in those cases this argument will be ignored.

-b *free-blocks*
Ensure that a minimum of *free-blocks* free blocks exist in the image. An optional '`%`' suffix may be provided to indicate that *free-blocks* indicates a percentage of the calculated image size.

-d *debug-mask*
Enable various levels of debugging, depending upon which bits are set in *debug-mask*. XXX: document these

-F *specfile*
Use *specfile* as an `mtree`(8) 'specfile' specification.

If a specfile entry 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 entry.

In the opposite case (where a specfile entry does not have an entry in the underlying file system) the following occurs: If the specfile entry is marked **optional**, the specfile entry is ignored. Otherwise, the entry will be created in the image, 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. If **flags** isn't provided, the current file flags will be used. Missing regular file entries will be created as zero-length files.

-f *free-files*
Ensure that a minimum of *free-files* free files (inodes) exist in the image. An optional '`%`' suffix may be provided to indicate that *free-files* indicates a percentage of the calculated image size.

-M *minimum-size*
Set the minimum size of the file system image to *minimum-size*.

-m *maximum-size*
Set the maximum size of the file system image to *maximum-size*. An error will be raised if the target file system needs to be larger than this to accommodate the provided directory tree.

-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 *fs-options*
Set file system specific options. *fs-options* is a comma separated list of options. Valid file system specific options are detailed below.

-S *sector-size*
Set the file system sector size to *sector-size*. Defaults to 512.

-s *image-size*
Set the size of the file system image to *image-size*.

-t *fs-type*
Create an *fs-type* file system image. The following file system types are supported:

ffs	BSD fast file system (default).
cd9660	ISO 9660 file system.

-x Exclude file system nodes not explicitly listed in the specfile.

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

FFS-specific options

ffs images have ffs-specific optional parameters that may be provided. Each of the options consists of a keyword, an equal sign ('='), and a value. The following keywords are supported:

avgfilesize	Expected average file size.
avgfpdir	Expected number of files per directory.
bsize	Block size.
density	Bytes per inode.
fsize	Fragment size.
maxbpg	Maximum blocks per file in a cylinder group.
minfree	Minimum % free.
optimization	Optimization preference; one of `space` or `time`.
extent	Maximum extent size.
maxbpcg	Maximum total number of blocks in a cylinder group.
version	UFS version. 1 for FFS (default), 2 for UFS2.

CD9660-specific options

cd9660 images have ISO9660-specific optional parameters that may be provided. The arguments consist of a keyword and, optionally, an equal sign ('='), and a value. The following keywords are supported:

allow-deep-trees	Allow the directory structure to exceed the maximum specified in the spec.
allow-max-name	Allow 37 instead of 33 characters for filenames by omitting the version id.

allow-multidot	Allow multiple dots in a filename.
applicationid	Application ID of the image.
boot-load-segment	Set load segment for the boot image.
bootimage	Filename of a boot image in the format "sysid;filename", where "sysid" is one of `i386`, `mac68k`, `macppc`, or `powerpc`.
generic-bootimage	Load a generic boot image into the first 32K of the cd9660 image.
hard-disk-boot	Boot image is a hard disk image.
keep-bad-images	Don't throw away images whose write was aborted due to an error. For debugging purposes.
label	Label name of the image.
no-boot	Boot image is not bootable.
no-emul-boot	Boot image is a "no emulation" ElTorito image.
no-trailing-padding	Do not pad the image (apparently Linux needs the padding).
preparer	Preparer ID of the image.
publisher	Publisher ID of the image.
rockridge	Use RockRidge extensions (for longer filenames, etc.).
volumeid	Volume set identifier of the image.

SEE ALSO

`strsuftoll`(3), `installboot`(8), `mtree`(8), `newfs`(8)

HISTORY

The **`makefs`** utility appeared in NetBSD 1.6.

AUTHORS

Luke Mewburn ⟨lukem@NetBSD.org⟩ (original program)
Daniel Watt,
Walter Deignan,
Ryan Gabrys,
Alan Perez-Rathke,
Ram Vedam (cd9660 support)

NAME

makekey — make encrypted keys or passwords

SYNOPSIS

makekey

DESCRIPTION

makekey encrypts a key and salt which it reads from the standard input and writes the result to the standard output. The key is expected to be eight bytes; the salt is expected to be two bytes. See crypt(3) for more information on what characters the key and salt can contain and how the encrypted value is calculated.

SEE ALSO

login(1), crypt(3)

HISTORY

A **makekey** command appeared in Version 7 AT&T UNIX.

NAME

makewhatis — create a whatis.db database

SYNOPSIS

/usr/libexec/makewhatis [**−fw**] [**−C** *file*] [*manpath ...*]

DESCRIPTION

makewhatis strips the NAME lines from compiled or raw man(1) pages and creates a whatis.db database for use in apropos(1), whatis(1), or with man(1)'s **−k** option. Man pages compressed with compress(1) and gzip(1) are uncompressed before processing.

When *manpath* is provided multiple times, the resulting database file is generated in the first directory specified, and contains entries for all the directories.

If *manpath* is not provided, **makewhatis** parses /etc/man.conf and regenerates the whatis database files specified there. Each database file is assumed to reside in the root of the appropriate man page hierarchy.

The options are as follows:

−C *file*	Use *file* (in man.conf(5) format) as configuration file instead of the default, /etc/man.conf.
−f	Don't spawn child processes to generate the individual database files, but do all the work synchronously in the foreground.
−w	Print warnings about input files we don't like.

FILES

whatis.db	name of the whatis database
/etc/man.conf	man(1) configuration file, used to get the location of the whatis databases when **makewhatis** is called without arguments

SEE ALSO

apropos(1), man(1), whatis(1), man.conf(5)

HISTORY

makewhatis first appeared in NetBSD 1.0, as a shell script written by J.T. Conklin ⟨jtc@NetBSD.org⟩ and Thorsten Frueauf ⟨frueauf@ira.uka.de⟩. Further work was done by Matthew Green, Luke Mewburn, and Chris Demetriou.

Matthias Scheler has reimplemented **makewhatis** in C in NetBSD 1.5.

AUTHORS

Matthias Scheler ⟨tron@NetBSD.org⟩

NAME

map-mbone − Multicast connection mapper

SYNOPSIS

/usr/sbin/map-mbone [**−d** *debug_level*] [**−f**] [**−g**] [**−n**] [**−r** *retry_count*] [**−t** *timeout_count*] [**starting_router**]

DESCRIPTION

map-mbone attempts to display all multicast routers that are reachable from the multicast *starting_router.* If not specified on the command line, the default multicast *starting_router* is the localhost.

map-mbone traverses neighboring multicast routers by sending the ASK_NEIGHBORS IGMP message to the multicast starting_router. If this multicast router responds, the version number and a list of their neighboring multicast router addresses is part of that response. If the responding router has recent multicast version number, then *map-mbone* requests additional information such as metrics, thresholds, and flags from the multicast router. For each new occurrence of neighboring multicast router in the reply and provided the flooding option has been selected, then *map-mbone* asks each of this multicast router for a list of neighbors. This search for unique routers will continue until no new neighboring multicast routers are reported.

INVOCATION

"−d" option sets the debug level. When the debug level is greater than the default value of 0, addition debugging messages are printed. Regardless of the debug level, an error condition, will always write an error message and will cause *map-mbone* to terminate. Non-zero debug levels have the following effects:

level 1	packet warnings are printed to stderr.
level 2	all level 1 messages plus notifications down networks are printed to stderr.
level 3	all level 2 messages plus notifications of all packet timeouts are printed to stderr.

"−f" option sets flooding option. Flooding allows the recursive search of neighboring multicast routers and is enable by default when starting_router is not used.

"−g" option sets graphing in GraphEd format.

"−n" option disables the DNS lookup for the multicast routers names.

"−r retry_count" sets the neighbor query retry limit. Default is 1 retry.

"−t timeout_count" sets the number of seconds to wait for a neighbor query reply before retrying. Default timeout is 2 seconds.

IMPORTANT NOTE

map-mbone must be run as root.

SEE ALSO

mrouted(8)**, mrinfo**(8)**, mtrace**(8)

AUTHOR

Pavel Curtis

NAME

master − Postfix master process

SYNOPSIS

master [**-Ddtv**] [**-c** *config_dir*] [**-e** *exit_time*]

DESCRIPTION

The **master**(8) daemon is the resident process that runs Postfix daemons on demand: daemons to send or receive messages via the network, daemons to deliver mail locally, etc. These daemons are created on demand up to a configurable maximum number per service.

Postfix daemons terminate voluntarily, either after being idle for a configurable amount of time, or after having serviced a configurable number of requests. Exceptions to this rule are the resident queue manager, address verification server, and the TLS session cache and pseudo-random number server.

The behavior of the **master**(8) daemon is controlled by the **master.cf** configuration file, as described in **master**(5).

Options:

-c *config_dir*
: Read the **main.cf** and **master.cf** configuration files in the named directory instead of the default configuration directory. This also overrides the configuration files for other Postfix daemon processes.

-D
: After initialization, run a debugger on the master process. The debugging command is specified with the **debugger_command** in the **main.cf** global configuration file.

-d
: Do not redirect stdin, stdout or stderr to /dev/null, and do not discard the controlling terminal. This must be used for debugging only.

-e *exit_time*
: Terminate the master process after *exit_time* seconds. Child processes terminate at their convenience.

-t
: Test mode. Return a zero exit status when the **master.pid** lock file does not exist or when that file is not locked. This is evidence that the **master**(8) daemon is not running.

-v
: Enable verbose logging for debugging purposes. This option is passed on to child processes. Multiple **-v** options make the software increasingly verbose.

Signals:

SIGHUP
: Upon receipt of a **HUP** signal (e.g., after "**postfix reload**"), the master process re-reads its configuration files. If a service has been removed from the **master.cf** file, its running processes are terminated immediately. Otherwise, running processes are allowed to terminate as soon as is convenient, so that changes in configuration settings affect only new service requests.

SIGTERM
: Upon receipt of a **TERM** signal (e.g., after "**postfix abort**"), the master process passes the signal on to its child processes and terminates. This is useful for an emergency shutdown. Normally one would terminate only the master ("**postfix stop**") and allow running processes to finish what they are doing.

DIAGNOSTICS

Problems are reported to **syslogd**(8).

ENVIRONMENT

MAIL_DEBUG
: After initialization, start a debugger as specified with the **debugger_command** configuration parameter in the **main.cf** configuration file.

MAIL_CONFIG
Directory with Postfix configuration files.

CONFIGURATION PARAMETERS

Unlike most Postfix daemon processes, the **master**(8) server does not automatically pick up changes to **main.cf**. Changes to **master.cf** are never picked up automatically. Use the "**postfix reload**" command after a configuration change.

RESOURCE AND RATE CONTROLS

default_process_limit (100)
The default maximal number of Postfix child processes that provide a given service.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

service_throttle_time (60s)
How long the Postfix **master**(8) waits before forking a server that appears to be malfunctioning.

MISCELLANEOUS CONTROLS

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.

debugger_command (empty)
The external command to execute when a Postfix daemon program is invoked with the -D option.

inet_interfaces (all)
The network interface addresses that this mail system receives mail on.

inet_protocols (ipv4)
The Internet protocols Postfix will attempt to use when making or accepting connections.

import_environment (see 'postconf -d' output)
The list of environment parameters that a Postfix process will import from a non-Postfix parent process.

mail_owner (postfix)
The UNIX system account that owns the Postfix queue and most Postfix daemon processes.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

FILES

/etc/postfix/main.cf, global configuration file.
/etc/postfix/master.cf, master server configuration file.

/var/spool/postfix/pid/master.pid, master lock file.

SEE ALSO

qmgr(8), queue manager
verify(8), address verification
master(5), master.cf configuration file syntax
postconf(5), main.cf configuration parameter syntax
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

mbr, **bootselect** — Master Boot Record bootcode

DESCRIPTION

An IBM PC boots from a disk by loading its first sector and executing the code in it. For a hard disk, this first sector usually contains a table of partitions present on the disk. The first sector of a disk containing such a table is called the Master Boot Record (MBR).

The code present in the MBR will typically examine the partition table, find the partition that is marked active, and boot from it. Booting from a partition simply means loading the first sector in that partition, and executing the code in it, as is done for the MBR itself.

NetBSD supplies serveral versions of the MBR bootcode:

Normal boot code `/usr/mdec/mbr`
: This version has the same functionality as that supplied by DOS/Windows and other operating systems: it picks the active partition and boots from it. Its advantage over other, older MBRs, is that it can detect and use extensions to the BIOS interface that will allow it to boot partitions that cross or start beyond the 8 Gigabyte boundary.

Bootselector `/usr/mdec/mbr_bootsel`
: The bootselecting MBR contains configurable code that will present the user with a simple menu, allowing a choice between partitions to boot from, and hard disks to boot from. The choices and default settings can be configured through `fdisk`(8).

Extended Bootselector `/usr/mdec/mbr_ext`
: The Extended Bootselecting MBR additionally allows NetBSD to be loaded from an Extended partition. It only supports systems whose BIOS supports the extensions to boot partitions beyond the 8 Gigabyte boundary.

Serial Bootselector `/usr/mdec/mbr_com0`
: This has the same features as `mbr_ext` but will read and write from the first serial port. It assumes that the BIOS has initialised the baud rate.

Serial Bootselector `/usr/mdec/mbr_com0_9600`
: This has the same features as `mbr_com0`. Additionally it initialises the serial port to 9600 baud.

The rest of this manual page will discuss the bootselecting versions of the MBR. The configurable items of the bootselector are:

timeout	The number of seconds that the bootcode will wait for the user to press a key, selecting a menu item. Must be in the range 0-3600, or −1 when it will wait forever.
default	The default partition or disk to boot from, should the timeout expire.

The bootselector will output a menu of the *bootmenu* names for each partition (as configured by `fdisk`(8)). The user can then select the partition or drive to boot from via the keyboard.

The numeric keys **1** upwards will initiate a startup from the corresponding partition.

Function keys **F1** through **F8** (keys **a** through **h** for the serial versions) will boot from harddisks 0 through 7 (BIOS numbers 0x80 through 0x87). Booting from a drive is simply done by reading the MBR of that drive and executing it, so the bootcode present in the MBR of the chosen drive determines which partition (if any) will be booted in the end.

The **Enter** key will cause the bootcode to find the active partition, and boot from it. If no key is pressed, the (configurable) default selection is picked.

DIAGNOSTICS

The following error are detected:

Code	*Text message*	*Explanation*
1	No active partition	The MBR has a partition table without an active partition.
2	Disk read error	There was an error reading the bootsector for the partition or drive selected.
3	No operating system	The bootsector was loaded successfully, but it was not valid (i.e., the magic number check failed, or it contained no code).
L	Invalid CHS read	The boot partition cannot be read using a CHS read and the system BIOS doesn't support LBA reads.
?		Unknown key.

The standard boot code will output the text message and stop. It may be necessary to reset to the system to continue.

The bootselect code will output 'Error <code>' and await further input.

SEE ALSO

`boot`(8), `disklabel`(8), `fdisk`(8), `installboot`(8), `mbrlabel`(8)

BUGS

The bootselect code has constraints because of the limited amount of space available. The only way to be absolutely sure that a bootselector will always fit on the disk when a partition table is used, is to make it small enough to fit into the first sector (512 bytes, 404 excluding the partition table and bootselect menu).

The error messages are necessarily terse.

NAME

mbrlabel — update disk label from MBR label(s)

SYNOPSIS

mbrlabel [**-fqrw**] [**-s** *sector*] *device*

DESCRIPTION

mbrlabel is used to update a NetBSD disk label from the Master Boot Record (MBR) label(s) found on disks that were previously used on DOS/Windows systems (or other MBR using systems).

mbrlabel scans the MBR contained in the very first block of the disk (or the block specified through the **-s** flag), then walks through every extended partition found and generates additional partition entries for the disk from the MBRs found in those extended partitions.

Each MBR partition which does not have an equivalent partition in the disk label (equivalent in having the same size and offset) is added to the first free partition slot in the disk label. A free partition slot is defined as one with an `fstype` of '`unused`' and a `size` of zero ('0'). If there are not enough free slots in the disk label, a warning will be issued.

The raw partition (typically partition *c*, but *d* on i386 and some other platforms) is left alone during this process.

By default, the proposed changed disk label will be displayed and no disk label update will occur.

Available options:

-f Force an update, even if there has been no change.

-q Performs operations in a quiet fashion.

-w Update the in-core label if it has been changed.

-r In conjunction with **-w**, also update the on-disk label.

-s *sector*
Specifies the logical sector number that has to be read from the disk in order to find the MBR. Useful if the disk has remapping drivers on it and the MBR is located in a non-standard place. Defaults to 0.

SEE ALSO

`disklabel`(8), `dkctl`(8), `fdisk`(8), `mbr`(8)

HISTORY

The **mbrlabel** command appeared in NetBSD 1.4.

NAME

mdconfig — configure MEMORY disks

SYNOPSIS

mdconfig *special_file 512-byte-blocks*

DESCRIPTION

The **mdconfig** command configures memory disk devices. It will associate the special file *special_file* with a range of user-virtual memory allocated by the **mdconfig** process itself. The **mdconfig** command should be run in the background. If successful, the command will not return. Otherwise, an error message will be printed.

To "unconfigure" the memory disk, just kill the background **mdconfig** process started earlier.

FILES

/dev/rmd??
/dev/md??

EXAMPLES

```
mdconfig /dev/md0c 2048 &
```

Configures the memory disk md0c with one megabyte of user-space memory.

SEE ALSO

mount(8), swapon(8), umount(8)

BUGS

The special device will become inoperative if the **mdconfig** process is killed while the special device is open.

NAME

mdsetimage — set kernel RAM disk image

SYNOPSIS

mdsetimage [**−svx**] [**−b** *bfdname*] *kernel image*

DESCRIPTION

The **mdsetimage** command copies the disk image specified by *image* into the memory disk storage area in *kernel*. The file system present in *image* will typically be used by the kernel as the root file system.

To recognize kernel executable format, the **−b** flag specifies BFD name of kernel.

If the **−s** flags is given, **mdsetimage** will write back the actual disk image size back into *kernel*.

If the **−v** flag is given, **mdsetimage** will print out status information as it is copying the image.

If the **−x** flag is given, **mdsetimage** will extract the disk image from *kernel* into the file *image*. This is the opposite of the default behavior.

SEE ALSO

md(4), mdconfig(8)

NAME

mdsetimage — set kernel RAM disk image

SYNOPSIS

mdsetimage [**-T** *address*] [**-v**] *kernel image*

DESCRIPTION

The **mdsetimage** command copies the disk image specified by *image* into the memory disk storage area in *kernel*. The file system present in *image* will typically be used by the kernel as the root file system.

For a.out kernels only, the **-T** flag specifies the starting address of kernel text. This flag is ignored for other kernel executable formats.

If the **-v** flag is given, **mdsetimage** will print out status information as it is copying the image.

SEE ALSO

md(4), mdconfig(8)

NAME

memswitch — get or set x68k memory switch

SYNOPSIS

memswitch −a
memswitch [**−h**] [**−n**] *variable ...*
memswitch −w *variable=value ...*
memswitch −r *filename*
memswitch −s *filename*

DESCRIPTION

The **memswitch** command gets or sets the x68k memory switch stored in the non-volatile static ram.

The first form shows the current values of all the variables of the memory switch.

The second form shows the current values of the specified variables. If the **−h** flag is specified, a brief descriptions of the variables are displayed. The **−n** flag suppresses printing of the variable name.

The third form sets or modifies the specified variables to the given value.

In the fourth and fifth form, the whole memory switch part of non-volatile SRAM is saved to, or restored from the specified file, respectively.

FILES

`/dev/sram`	non-volatile static memory control device

HISTORY

The **memswitch** command first appeared in NetBSD 1.5.

NAME

mk-amd-map − create database maps for Amd

SYNOPSIS

mk-amd-map [**−p**] *mapname*

DESCRIPTION

mk-amd-map creates the database maps used by the keyed map lookups in amd(8). It reads input from the named file and outputs them to a correspondingly named hashed database.

−p This option prints the map on standard output instead of generating a database. This is usually used to merge continuation lines into one physical line.

SEE ALSO

amd(8).

‘‘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

Jan-Simon Pendry <jsp@doc.ic.ac.uk>, Department of Computing, Imperial College, London, UK.

Erez Zadok <ezk@cs.sunysb.edu>, Computer Science Department, Stony Brook University, New York, USA.

Other authors and contributors to am-utils are listed in the **AUTHORS** file distributed with am-utils.

NAME

mkalias — a NIS map conversion program

SYNOPSIS

mkalias [**−deEnsuv**] *input* [*output*]

DESCRIPTION

mkalias is used to convert a `mail.aliases` map to a `mail.byaddr` map. This is a inverse map of user@host (or user!host) back to alias.

mkalias uses *input* as the input map, and if *output* is given, use that as the output map. If the output map isn't given don't create database. This can be useful when **−e** or **−E** is given.

The options are as follows:

−d	Assume domain names are OK. Only useful together with **−e** or **−E**.
−e	Check host to verify that it exists.
−E	Same as **−e**, but also check for any MX-record.
−n	Capitalize name. E.g., mats.o.jansson becomes Mats.O.Jansson.
−u	Assume UUCP names are OK. Only useful together with **−e** or **−E**.
−s	Ignored (only provided for compatibility with SunOS 4.1.x).
−v	Verbose mode.

SEE ALSO

nis(8), ypserv(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

mkbootimage — turn Alpha bootstrap programs into bootable images

SYNOPSIS

/usr/mdec/mkbootimage [**−nv**] *infile* [*outfile*]

DESCRIPTION

The **mkbootimage** utility creates bootable image files from NetBSD/alpha bootstrap programs. Bootable image files can be placed directly on disk or tape to create bootable media which can be booted by the SRM console. This is primarily useful for creating bootable tapes or disk sets with the /usr/mdec/ustarboot bootstrap program, or for creating firmware upgrade media using firmware upgrade programs.

The bootstrap program *infile* is padded to a 512-byte boundary, has a properly formed Alpha Boot Block prepended, and is written to the output file *outfile*. If no output file is specified, the result is written to standard output.

The **mkbootimage** utility does not install bootstrap programs to make disks bootable. To do that, use installboot(8). Similarly, it is not necessary to use **mkbootimage** to create images to boot over the network; network-capable bootstrap programs are usable without modification.

The options recognized by **mkbootimage** are as follows:

−n Do not actually write the result to the output file or standard output.

−v Print information about what **mkbootimage** is doing.

The **mkbootimage** utility exits 0 on success, and >0 if an error occurs.

FILES

/usr/mdec/ustarboot "ustar" file system bootstrap program

EXAMPLES

```
mkbootimage as200_v5_8.exe as200_v5_8.exe.bootimage
```

Create a bootable image from the (firmware image) file as200_v5_8.exe. That bootable image could then be written to floppy, disk, CD-ROM, or tape to create bootable firmware update media.

```
(mkbootimage /usr/mdec/ustarboot; tar cvf - netbsd) | \
    dd of=/dev/rst0
```

Make a bootable image from the bootstrap program /usr/mdec/ustarboot, concatenate it with a tar file containing a kernel, and write the output to a tape. This is an example of how to create a tape which boots a kernel.

SEE ALSO

boot(8), installboot(8)

HISTORY

The NetBSD/alpha **mkbootimage** command first appeared in NetBSD 1.4.

AUTHORS

The **mkbootimage** utility was written by Chris Demetriou.

NAME

mkbootimage — create a prep boot image

SYNOPSIS

mkbootimage [**-lsv**] [**-m** *machine_arch*] [**-b** *bootfile*] [**-k** *kernel*] [**-r** *rawdev*] *boot-image*

DESCRIPTION

mkbootimage is the utility used to create a bootable kernel image on NetBSD for prep, bebox or rs6000.

The **mkbootimage** utility takes the boot-program, and the optional kernel, and creates a boot image from them. This image contains the boot code, kernel, and optionally an i386 partition table. The image can be written directly to a floppy or hard drive with the `dd`(1) command, or it can be directly netbooted via `bootpd`(8).

The following options are available:

-b Specifies which bootloader to embed in the bootable image. Defaults to `/usr/mdec/boot`.

-k Specifies which kernel binary to embed in the bootable image. Defaults to `/netbsd`.

-l Creates a partition table for a 2.88MB floppy instead of a 1.44MB floppy. This is primarily used for El-Torrito style CD images.

-m Selects the machine architecture to build the image for. Currently supports prep, rs6000 and bebox. Defaults to the machine architecture you are currently running on. This option is required if you are building an image for another machine, such as building a prep boot image on i386.

-r Specifies the raw device to read to gather the current partition table. This is generally `/dev/rsd0c`.

-s Generates a standalone image with no partition table embedded.

-v Generates verbose output, useful for debugging.

There are three primary ways to use **mkbootimage** to build a bootable image:

The first method is to build an image suitable for a floppy or netboot. This will create an image with an embedded partition table with a single PReP boot partition of type 0x41(65). The image can be directly netbooted, or if it is small enough, written directly to a floppy with `dd`(1). **mkbootimage** will warn you if the generated image is too large to be written to a floppy.

The second method is to build a standalone image with no partition table. This should be written to the PReP boot partition on your hard drive with `dd`(1).

The third method is for use in upgrading older systems that have been built by writing the floppy image directly to the head of the hard drive. This method reads the existing partition table and embeds that in the image. This should prevent loss of your current partition layout. This image should be written directly to the head of the disk with `dd`(1).

The recommended setup for a PReP machine is to build a partition table with `fdisk`(8) that contains a PReP boot partition (type 65) as partition 0, marked active, and a second partition for NetBSD encompassing the remainder of the disk. You should then create a disklabel on that disk with a partition (such as e) pointing to the PReP boot partition. Partition c should be the whole disk, and partition d can optionally be the NetBSD portion of the disk. You may then use the other partitions for your normal disk layout. The PReP boot partition can be placed anywhere on the disk, but it is recommended that it be placed at the beginning of the disk.

EXAMPLES

Create a floppy or netboot image for prep named 'boot.fs':

```
mkbootimage -m prep -b /usr/mdec/boot -k /netbsd boot.fs
```

Create a standalone bebox image for booting from a hard disk:

```
mkbootimage -s -m bebox -b /usr/mdec/boot -k /netbsd boot.fs
```

Use the partition information on 'sd0' to create a new bootable image with com0 as the console:

```
mkbootimage -b /usr/mdec/boot_com0 -k /netbsd -r /dev/rsd0c boot.fs
```

SEE ALSO

dd(1), boot(8), bootpd(8), disklabel(8), fdisk(8)

HISTORY

mkbootimage first appeared in NetBSD 1.5.

AUTHORS

mkbootimage was written by NONAKA Kimihiro.

NAME

mknetid — a NIS filter program

SYNOPSIS

mknetid [**-q**] [**-d** *domain*] [**-p** *passwdfile*] [**-g** *groupfile*] [**-h** *hostfile*] [**-m** *netidfile*]

DESCRIPTION

mknetid is used to create a map named `netid.byname`. The map consists of information from `passwd`(5), `group`(5) and `hosts`(5) eventually concatenated with a `netid`(5) file.

The options are as follows:

-d *domain*	NIS domain to use instead of the default domain.
-g *groupfile*	Alternate `group`(5) file. Default is `/etc/group`.
-h *hostfile*	Alternate `hosts`(5) file. Default is `/etc/hosts`.
-m *netidfile*	Alternate `netid`(5) file. Default is `/etc/netid`.
-p *passwdfile*	Alternate `passwd`(5) file. Default is `/etc/passwd`.
-q	Keep quiet about multiple occurrences of a uid; ignore all but the first.

FILES

`/etc/group`
`/etc/hosts`
`/etc/netid`
`/etc/passwd`

SEE ALSO

`domainname`(1), `group`(5), `hosts`(5), `netid`(5), `passwd`(5), `nis`(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

mknod — make device special file

SYNOPSIS

mknod [**-rR**] [**-F** *fmt*] [**-g** *gid*] [**-m** *mode*] [**-u** *uid*] *name* [**c** | **b**] [*driver* | *major*] *minor*
mknod [**-rR**] [**-F** *fmt*] [**-g** *gid*] [**-m** *mode*] [**-u** *uid*] *name* [**c** | **b**] *major unit subunit*
mknod [**-rR**] [**-g** *gid*] [**-m** *mode*] [**-u** *uid*] *name* [**c** | **b**] *number*
mknod [**-rR**] [**-g** *gid*] [**-m** *mode*] [**-u** *uid*] *name* **p**
mknod -l

DESCRIPTION

The **mknod** command creates device special files, or fifos. Normally the shell script `/dev/MAKEDEV` is used to create special files for commonly known devices; it executes **mknod** with the appropriate arguments and can make all the files required for the device.

To make nodes manually, the arguments are:

-r Replace an existing file if its type is incorrect.

-R Replace an existing file if its type is incorrect. Correct the mode, user and group.

-F *fmt* Create device nodes that may be used by an operating system which uses device numbers packed in a different format than NetBSD uses. This is necessary when NetBSD is used as an NFS server for netbooted computers running other operating systems.

The following values for the *fmt* are recognized: **native**, **386bsd**, **4bsd**, **bsdos**, **freebsd**, **hpux**, **isc**, **linux**, **netbsd**, **osf1**, **sco**, **solaris**, **sunos**, **svr3**, **svr4**, and **ultrix**.

-g *gid* Specify the group for the device node. The *gid* operand may be a numeric group ID or a group name. If a group name is also a numeric group ID, the operand is used as a group name. Precede a numeric group ID with a **#** to stop it being treated as a name.

-m *mode*
Specify the mode for the device node. The mode may be absolute or symbolic, see chmod(1).

-u *uid* Specify the user for the device node. The *uid* operand may be a numeric user ID or a user name. If a user name is also a numeric user ID, the operand is used as a user name. Precede a numeric user ID with a **#** to stop it being treated as a name.

name Device name, for example "sd" for a SCSI disk on an HP300 or a "pty" for pseudo-devices.

b | **c** | **p**
Type of device. If the device is a block type device such as a tape or disk drive which needs both cooked and raw special files, the type is **b**. All other devices are character type devices, such as terminal and pseudo devices, and are type **c**. Specifying **p** creates fifo files.

driver | *major*
The major device number is an integer number which tells the kernel which device driver entry point to use. If the device driver is configured into the current kernel it may be specified by driver name or major number. To find out which major device number to use for a particular device, use **mknod -l**, check the file `/dev/MAKEDEV` to see if the device is known, or check the system dependent device configuration file:

"`/usr/src/sys/arch/<arch>/<arch>/conf.c`" (e.g. `/usr/src/sys/arch/vax/vax/conf.c`).

minor The minor device number tells the kernel which one of several similar devices the node corresponds to; for example, it may be a specific serial port or pty.

unit and *subunit*
The unit and subunit numbers select a subset of a device; for example, the unit may specify a particular SCSI disk, and the subunit a partition on that disk. (Currently this form of specification is only supported by the *bsdos* format, for compatibility with the BSD/OS **mknod**).

number A single opaque device number. Useful for netbooted computers which require device numbers packed in a format that isn't supported by **-F**.

-l List the device drivers configured into the current kernel together with their block and character major numbers.

SEE ALSO

chmod(1), mkfifo(1), mkfifo(2), mknod(2), MAKEDEV(8)

HISTORY

A **mknod** command appeared in Version 6 AT&T UNIX. The **-F** option appeared in NetBSD 1.4. The **-g**, **-l**, **-m**, **-r**, **-R**, and **-u** options, and the ability to specify a driver by name appeared in NetBSD 2.0.

NAME

mld6query — send multicast listener query

SYNOPSIS

mld6query [**-dr**] *intface* [*maddr*]

DESCRIPTION

mld6query sends an IPv6 multicast listener discovery (MLD) query packet toward the specified multicast address, *maddr*, toward interface *intface*. If you omit *maddr*, linklocal all nodes multicast address(ff02::1) is used.

After sending a query, **mld6query** waits for replies for at most 10 seconds. If a reply is returned, **mld6query** prints it with its type and then waits for another reply.

This program is provided only for debugging. It is not necessary for normal use.

With **-d**, **mld6query** will transmit MLD done packet instead of MLD query packet. With **-r**, similarly, MLD report packet will be transmitted. **-dr** options are for debugging purposes only.

EXIT STATUS

The program exits 0 on success, and >0 on failures.

HISTORY

The **mld6query** command first appeared in WIDE/KAME IPv6 protocol stack kit.

BUGS

mld6query does not take care of multicast addresses which have non link-local scope.

NAME

mlxctl — Mylex DAC960 family management utility

SYNOPSIS

mlxctl [**-f** *dev*] [**-v**] [**-a**] status [*drive*] [*. . .*]
mlxctl [**-f** *dev*] [**-a**] detach [*drive*] [*. . .*]
mlxctl [**-f** *dev*] [**-a**] check [*drive*] [*. . .*]
mlxctl [**-f** *dev*] rebuild *channel:target*
mlxctl [**-f** *dev*] cstatus
mlxctl [**-f** *dev*] rescan
mlxctl [**-f** *dev*] config

DESCRIPTION

The **mlxctl** utility performs status monitoring and management functions for Mylex DAC960 RAID controllers and attached devices.

The following options are available:

-a	Apply the action to all drives attached to the controller.
-f *dev*	Specify the control device to use. The default is `/dev/mlx0`.
-v	Increased verbosity.

The following commands are available:

cstatus	Display the controller's current status.
status	Display the status of the specified drives. This command returns 0 if all drives tested are online, 1 if one or more drives are critical and 2 if one or more are offline.
rescan	Re-scan the logical drive table, and attach or detach devices from the system as necessary.
detach	Detach the specified drives. Drives must be unmounted and unopened for this command to succeed.
check	Initiate a consistency check and repair pass on a drive that provides redundancy (e.g., RAID1 or RAID5). This command returns immediately. The *status* command can be used to monitor the progress of the check.
rebuild	Rebuild onto the specified physical drive. Note that there can be only one running rebuild operation per controller at any given time. This command returns immediately. The *cstatus* command can be used to monitor the progress of the rebuild.
config	Write the current system drive configuration to stdout.

EXAMPLES

Display the status of drive ld3 attached to the controller mlx1:

```
mlxctl -f /dev/mlx1 -v status ld3
```

SEE ALSO

`ld`(4), `mlx`(4)

HISTORY

The **mlxctl** command first appeared in NetBSD 1.5.3, and was based on the **mlxcontrol** utility found in FreeBSD.

BUGS

Modifying drive configuration is not yet supported.

Some commands do not work with older firmware revisions.

Error log extraction is not yet supported.

NAME

mmcformat — format optical media

SYNOPSIS

mmcformat [**−B**] [**−F**] [**−O**] [**−M**] [**−R**] [**−G**] [**−S**] [**−s**] [**−w**] [**−p**] [**−c** *cert-num*] [**−r**] [**−h**] [**−H**] [**−I**] [**−b −blockingnr**] [D] *special*

DESCRIPTION

The **mmcformat** utility formats optical media conforming to the MMC standard. These include CD , DVD, and Blu-Ray (BD) media.

The options are as follows:

−B Blank media when possible before formatting it.

−F Format media.

−O Old style CD-RW formatting ; recommended for CD-RW.

−M Select MRW (Mount Raineer) error correcting background format.

−R Restart previously stopped MCD-MRW or DVD+RW backfround format.

−G Grow last CD-RW/DVD-RW session.

−S Grow spare space DVD-RAM / BD-RE.

−s Format DVD+MRW / BD-RE with extra spare space.

−w Wait until completion of background format.

−p Explicitly set packet format.

−c *cert-num*
Certify media for DVD-RAM / DV-RE. The argument cert-num specifies: 0) no certification , 1) full certification , 2) quick certification.

−r Recompile defect list for DVD-RAM.

−h Show help or inquiry format possibilities for inserted media.

−H Show help or inquiry format possibilities for inserted media.

−i Show help or inquiry format possibilities for inserted media.

−b −blockingnr
Explicit select packet size in sectors for CD-RW only; not recommended to change from its default 32.

−D Verbose all SCSI/ATAPI command errors.

NOTES

Due to the enormous varieties in optical media, the tool is made as generic as possible. This can result in confusion.

EXAMPLES

```
mmcformat -B -O /dev/rcd0d
```

Blanks and then formats a CD-RW disc using the 'old-style' format command. Its recommended to use this 'old-style' command unless your drive reports its not supported; if so, resort to the default **−F** Note that a CD-RW disc can be reformatted without being blanked. Blanking swiches between sequential and fixed packet writing by erasing the disc. This can also help to revive old discs.

```
mmcformat -F -M /dev/rcd0d
```

Format a CD-RW or a DVD+RW to use MRW (Mount Raineer). This format tries to hide media flaws as much as possible by relocation.

SEE ALSO

`scsictl`(8)

BUGS

It could be merged with `scsictl`(8) but that tool is very harddisc oriented.

HISTORY

The **`mmcformat`** command first appeared in NetBSD 5.0.

AUTHORS

Reinoud Zandijk ⟨reinoud@NetBSD.org⟩.

NAME

modload — load a kernel module

SYNOPSIS

modload [**-dfnsSv**] [**-A** *kernel*] [**-e** *entry*] [**-p** *postinstall*] [**-o** *output_file*] [**-T** *linker_script*] *input_file*

DESCRIPTION

The **modload** utility loads a loadable kernel module into a running system. The input file is an object file (.o file).

The options to **modload** are as follows:

-d Debug. Used to debug **modload** itself.

-f This forces load of the module, even if it doesn't match the currently running kernel. When LKM is loaded, the kernel normally checks if the LKM is compatible with the running kernel. This option disables this check. *Note* an incompatible LKM can cause system instability, including data loss or corruption. Don't use this option unless you are sure what you are doing.

-n Do everything, except calling the module entry point (and any post-install program).

-v Print comments about the loading process.

-s Load the symbol table.

-S Do not remove the temporary object file. By default, the ld(1) output is removed after being loaded into the kernel.

-A *kernel*
Specify the file that is passed to the linker to resolve module references to external symbols. The symbol file must be for the currently running kernel or the module is likely to crash the system.

-e *entry*
Specify the module entry point. This is passed by **modload** to ld(1) when the module is linked. The default module entry point name is 'xxxinit'. If 'xxxinit' cannot be found, an attempt to use '<module_name>_lkmentry' will be made, where <module_name> is the filename being loaded without the '.o'.

-p *postinstall*
Specify the name of a shell script or program that will be executed if the module is successfully loaded. It is always passed the module id (in decimal) and module type (in hexadecimal) as the first two arguments. For loadable drivers, the third argument is the character major device number and the fourth argument is the block major device number. For a loadable system call, the third argument is the system call number.

-o *output_file*
Specify the name of the output file that is produced by the linker.

-T *linker_script*
Specify the name of the linker script use to link against the kernel.

FILES

/netbsd	default file passed to the linker to resolve external references in the module
/usr/include/sys/lkm.h	file containing definitions of module types

DIAGNOSTICS

The **modload** utility exits with a status of 0 on success and with a nonzero status if an error occurs.

Mismatched LKM and kernel versions will be reported to the console and to the system message buffer.

SEE ALSO

`ld`(1), `lkm`(4), `modstat`(8), `modunload`(8)

HISTORY

The **modload** command was designed to be similar in functionality to the corresponding command in SunOS 4.1.3.

AUTHORS

Terrence R. Lambert ⟨terry@cs.weber.edu⟩.

BUGS

Loading the symbol table is expensive in terms of space: it presently duplicates all the kernel symbols for each lkm loaded with **-s**.

NAME

modstat — display status of loaded kernel modules

SYNOPSIS

modstat [**-n** *name*]

DESCRIPTION

The **modstat** utility displays the status of any kernel modules present in the kernel.

The options are as follows:

-n *name*
Display the status of only the module with this name.

In addition to listing the currently loaded modules' name, the information reported by **modstat** includes:

CLASS	Module class, such as "vfs", "driver", "exec" or "misc".
SOURCE	Where the module was loaded from. "builtin" indicates that the module was built into the running kernel. "boot" indicates that the module was loaded during system bootstrap. "filesys" indicates that the module was loaded from the file system.
SIZE	Size of the module in bytes.
REFS	Number of references held on the module.
REQUIRES	Additional modules that must be present.

EXIT STATUS

The **modstat** utility exits with a status of 0 on success and with a nonzero status if an error occurs.

SEE ALSO

modload(8), modunload(8)

HISTORY

The **modstat** command was designed to be similar in functionality to the corresponding command in SunOS 4.1.3.

NAME

modstat — display status of loaded kernel modules

SYNOPSIS

modstat [**-i** *id*] [**-n** *name*]

DESCRIPTION

The **modstat** utility displays the status of any loadable kernel modules present in the kernel.

The options are as follows:

-i *id* Display the status of only the module with this ID.

-n *name*
Display the status of only the module with this name.

In addition to listing the currently loaded modules' name, the information reported by **modstat** includes:

`Type`	Type of module, such as "SYSCALL", "VFS", "DEV", "EXEC", "COMPAT", "MISC", or "(UNKNOWN)". See lkm(4) for details.
`Id`	Module identification number.
`Offset`	Target table offset.
`Loadaddr`	Address the module loaded at.
`Size`	Size in kilobytes (in hex).
`Info`	Kernel address of private area.
`Rev`	Version of the module interface. (Not the version of the module itself.)

EXIT STATUS

The **modstat** utility exits with a status of 0 on success and with a nonzero status if an error occurs.

SEE ALSO

lkm(4), modload(8), modunload(8)

HISTORY

The **modstat** command was designed to be similar in functionality to the corresponding command in SunOS 4.1.3.

AUTHORS

Terrence R. Lambert ⟨terry@cs.weber.edu⟩

NAME

modunload — unload a kernel module

SYNOPSIS

modunload [**-i** *id*] [**-n** *name*]
modunload [*id*|*name*]

DESCRIPTION

The **modunload** utility unloads a loadable kernel module from a running system. The *id* or *name* is the ID or name of the module as shown by `modstat`(8).

One of the following options may be specified:

-i *id* Unload the module with the ID *id*.

-n *name*
Unload the module with the name *name*.

If the option is not specified, the provided argument is used as module ID if it's numeric. If the argument is not numeric, it's used as module name.

DIAGNOSTICS

The **modunload** utility exits with a status of 0 on success and with a nonzero status if an error occurs.

SEE ALSO

`lkm`(4), `modload`(8), `modstat`(8)

HISTORY

The **modunload** command was designed to be similar in functionality to the corresponding command in SunOS 4.1.3.

AUTHORS

Terrence R. Lambert ⟨terry@cs.weber.edu⟩.

NAME

mopd — Maintenance Operations Protocol (MOP) Loader Daemon

SYNOPSIS

mopd [**-adf**] [**-s** *mopdir*] [*interface*] [*...*]

DESCRIPTION

mopd services DEC Maintenance Operations Protocol (MOP) Load requests on the Ethernet connected to *interface* or all interfaces if **-a** option is given.

In a load request received by **mopd** a filename can be given by the client. This is the normal case for terminal servers. If a filename isn't in the client load request **mopd** must know what image to load.

Upon receiving a request, **mopd** checks if the requested file exists in `/tftpboot/mop` (unless the **-s** option is given, see below) the filename is normally uppercase and with an extension of `.SYS`. If the filename isn't given, the ethernet address of the target is used as filename, e.g. `08002b09f4de.SYS` and it might be a soft link to another file.

mopd supports two kinds of files. The first type that is check is if the file is in `a.out`(5) format. If not, a couple of Digital's formats are checked.

In normal operation, **mopd** forks a copy of itself and runs in the background. Anomalies and errors are reported via `syslog`(3).

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 stdout. The process will run in the foreground.

-f Run in the foreground.

-s Change the directory to look for files in from `/tftpboot/mop` to *mopdir*.

FILES

`/tftpboot/mop`

NOTES

mopd automatically appends an upper case `.SYS` to the filename provided by the client. The typical client sends the requested file name in upper case.

SEE ALSO

`mopchk`(1), `mopcopy`(1), `mopprobe`(1), `moptrace`(1), `bpf`(4)

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

mount — mount file systems

SYNOPSIS

mount [**−Aadfruvw**] [**−t** *type*]
mount [**−dfruvw**] *special* | *node*
mount [**−dfruvw**] [**−o** *options*] [**−t** *type*] *special node*

DESCRIPTION

The **mount** command invokes a file system-specific program to prepare and graft the *special* device or remote node (rhost:path) on to the file system tree at the point *node*.

If either *special* or *node* are not provided, the appropriate information is taken from the `fstab`(5) file. The provided argument is looked up first in the "fs_file", then in the "fs_spec" column. If the matching entry in `fstab`(5) has the string "`from_mount`" as its "fs_spec" field, the device or remote file system already mounted at the location specified by "fs_spec" will be used.

If both *special* and *node* are given, the disklabel is checked for the file system type.

In NetBSD, a file system can only be mounted by an ordinary user who owns the point *node* and has access to the *special* device (at least read permissions). Also, the *vfs.generic.usermount* `sysctl`(3) must be set to 1 to permit file system mounting by ordinary users, see `sysctl`(8). Finally, the flags **nosuid** and **nodev** must be given for non-superuser mounts.

The system maintains a list of currently mounted file systems. If no arguments are given to **mount**, this list is printed.

The options are as follows:

−A Causes **mount** to try to mount all of the file systems listed in the `fstab`(5) file except those for which the "noauto" option is specified.

−a Similar to the **−A** flag, except that if a file system (other than the root file system) appears to be already mounted, **mount** will not try to mount it again. **mount** assumes that a file system is already mounted if a file system with the same type is mounted on the given mount point. More stringent checks are not possible because some file system types report strange values for the mounted-from device for mounted file systems.

−d Causes everything to be done except for the invocation of the file system-specific program. This option is useful in conjunction with the **−v** flag to determine what the **mount** command is trying to do.

−f Forces the revocation of write access when trying to downgrade a file system mount status from read-write to read-only.

−o Options are specified with a **−o** flag followed by a comma separated string of options. The following options are available:

async All I/O to the file system should be done asynchronously. In the event of a crash, *it is impossible for the system to verify the integrity of data on a file system mounted with this option*. You should only use this option if you have an application-specific data recovery mechanism, or are willing to recreate the file system from scratch.

noasync Clear **async** mode.

force The same as **-f**; forces the revocation of write access when trying to downgrade a file system mount status from read-write to read-only.

getargs Retrieves the file system specific mount arguments for the given mounted file system and prints them.

hidden By setting the `MNT_IGNORE` flag, causes the mount point to be excluded from the list of file systems shown by default with `df`(1).

noatime Never update the access time field for files. This option is useful for optimizing read performance on file systems that are used as news spools.

noauto This file system should be skipped when mount is run with the **-a** flag.

nocoredump Do not allow programs to create crash dumps (core files) on the file system. This option can be used to help protect sensitive data by keeping core files (which may contain sensitive data) from being created on insecure file systems. Only core files that would be created by program crashes are prevented by use of this flag; the behavior of `savecore`(8) is not affected.

nodev Do not interpret character or block special devices on the file system. This option is useful for a server that has file systems containing special devices for architectures other than its own.

nodevmtime Do not update modification times on device special files. This option is useful on laptops or other systems that perform power management.

noexec Do not allow execution of any binaries on the mounted file system. This option is useful for a server that has file systems containing binaries for architectures other than its own.

nosuid Do not allow set-user-identifier or set-group-identifier bits to take effect.

port (NFS only) Use the specified NFS port.

rdonly The same as **-r**; mount the file system read-only (even the super-user may not write it).

reload Reload all incore data for a file system. This is used mainly after running `fsck`(8) on the root file system and finding things to fix. The file system must be mounted read-only. All cached meta-data are invalidated, superblock and summary information is re-read from disk, all cached inactive vnodes and file data are invalidated and all inode data are re-read for all active vnodes.

softdep (FFS only) Mount the file system using soft dependencies. This means that metadata will not be written immediately, but is written in an ordered fashion to keep the on-disk state of the file system consistent. This results in significant speedups for file create/delete operations. This option will be ignored when using the **-u** flag and a file system is already mounted read/write. This option has gone through moderate to heavy testing, but should still be used with care. A file system mounted with **softdep** can not be mounted with **async**. It requires the `SOFTDEP` option to be enabled in the running kernel.

symperm Recognize permission of symbolic link when reading or traversing link.

sync All I/O to the file system should be done synchronously. This is not equivalent to the normal mode in which only metadata is written synchronously.

nosync Clear **sync** mode.

union Causes the namespace at the mount point to appear as the union of the mounted file system root and the existing directory. Lookups will be done in the mounted file system first. If those operations fail due to a non-existent file the underlying directory is then accessed. All creates are done in the mounted file system, except for the fdesc file system.

update The same as **−u**; indicate that the status of an already mounted file system should be changed.

Any additional options specific to a given file system type (see the **−t** option) may be passed as a comma separated list; these options are distinguished by a leading "-" (dash). Options that take a value are specified using the syntax -option=value. For example, the mount command:

```
mount -t mfs -o nosuid,-N,-s=32m swap /tmp
```

causes **mount** to execute the equivalent of:

```
/sbin/mount_mfs -o nosuid -N -s 32m swap /tmp
```

−r The file system is to be mounted read-only. Mount the file system read-only (even the super-user may not write it). The same as the "rdonly" argument to the **−o** option.

−t *type*
The argument following the **−t** is used to indicate the file system type. The type *ffs* is the default. The **−t** option can be used to indicate that the actions should only be taken on file systems of the specified type. More than one type may be specified in a comma separated list. The list of file system types can be prefixed with "no" to specify the file system types for which action should *not* be taken. For example, the **mount** command:

```
mount -a -t nonfs,mfs
```

mounts all file systems except those of type NFS and MFS.

mount will attempt to execute a program in /sbin/mount_*XXX* where *XXX* is replaced by the type name. For example, nfs file systems are mounted by the program /sbin/mount_nfs.

−u The **−u** flag indicates that the status of an already mounted file system should be changed. Any of the options discussed above (the **−o** option) may be changed; also a file system can be changed from read-only to read-write or vice versa. An attempt to change from read-write to read-only will fail if any files on the file system are currently open for writing unless the **−f** flag is also specified. The set of options is determined by first extracting the options for the file system from the fstab(5) file, then applying any options specified by the **−o** argument, and finally applying the **−r** or **−w** option.

−v Verbose mode. If this flag is specified more than once, then the file system-specific mount arguments are printed for the given mounted file system.

−w The file system object is to be read and write.

The options specific to the various file system types are described in the manual pages for those file systems' **mount_XXX** commands. For instance the options specific to Berkeley Fast File System (FFS) are described in the mount_ffs(8) manual page.

The particular type of file system in each partition of a disk can be found by examining the disk label with the disklabel(8) command.

FILES

`/etc/fstab` file system table

EXAMPLES

Some useful examples:

CD-ROM

```
mount -t cd9660 -r /dev/cd0a /cdrom
```

MS-DOS

```
mount -t msdos /dev/fd0a /floppy
```

NFS

```
mount nfs-server-host:/directory/path /mount-point
```

MFS (32 megabyte)

```
mount -t mfs -o nosuid,-s=32m swap /tmp
```

The "noauto" directive in `/etc/fstab` can be used to make it easy to manually mount and unmount removable media using just the mountpoint filename, with an entry like this:

```
/dev/cd0a /cdrom cd9660 ro,noauto 0 0
```

That would allow a simple command like "mount /cdrom" or "umount /cdrom" for media using the ISO-9660 file system format in the first CD-ROM drive.

DIAGNOSTICS

The error "Operation not supported by device" indicates that the mount for the specified file-system type cannot be completed because the kernel lacks support for the said file-system. See `options`(4).

The error "Operation not permitted" may indicate that the mount options include privileged options and/or don't include options that exclude privileged options. One should try using at least "nodev" and "nosuid" in such cases:

```
mount -t cd9660 -o nodev,nosuid /dev/cd0a /mnt
```

SEE ALSO

`df`(1), `mount`(2), `options`(4), `fstab`(5), `disklabel`(8), `fsck`(8), `mount_ados`(8), `mount_cd9660`(8), `mount_ext2fs`(8), `mount_fdesc`(8), `mount_ffs`(8), `mount_filecore`(8), `mount_kernfs`(8), `mount_lfs`(8), `mount_mfs`(8), `mount_msdos`(8), `mount_nfs`(8), `mount_ntfs`(8), `mount_null`(8), `mount_overlay`(8), `mount_portal`(8), `mount_procfs`(8), `mount_tmpfs`(8), `mount_udf`(8), `mount_umap`(8), `mount_union`(8), `umount`(8)

HISTORY

A **`mount`** command appeared in Version 6 AT&T UNIX.

NAME

mount_9p — mount a file server using the 9P resource sharing protocol

SYNOPSIS

mount_9p [**-s**] [**-o** *mntopts*] [**-p** *port*] *[user@]host[:path]* *mount_point*

DESCRIPTION

The **mount_9p** program is used to mount a file hierarchy served with the Plan 9 file sharing protocol: 9P. After the file system is mounted, the files on the remote *host* will be accessed using the credentials of the user named *user* and whatever UID the user happens to have on the remote server. If *path* is supplied, it is used as the mount rootpath on the remote host. *path* must be an absolute path.

SEE ALSO

puffs(3), puffs(4), mount(8)

HISTORY

The **mount_9p** utility first appeared in NetBSD 5.0.

CAVEATS

Permissions are not handled well.

Authentication support is missing.

Error code handling is missing.

Under construction.

NAME

mount_ados — mount an AmigaDOS file system

SYNOPSIS

mount_ados [**-o** *options*] [**-u** *uid*] [**-g** *gid*] [**-m** *mask*] special node

DESCRIPTION

The **mount_ados** command attaches the AmigaDOS filesystem residing on the device special to the global filesystem namespace at the location indicated by node. Both *special* and *node* are converted to absolute paths before use. This command is normally executed by mount(8) at boot time, but can be used by any user to mount an AmigaDOS file system on any directory that they own (provided, of course, that they have appropriate access to the device that contains the file system).

The options are as follows:

-o *options*
Use the specified mount *options*, as described in mount(8).

-u *uid*
Set the owner of the files in the file system to *uid*. The default owner is the owner of the directory on which the file system is being mounted.

-g *gid*
Set the group of the files in the file system to *gid*. The default group is the group of the directory on which the file system is being mounted.

-m *mask*
Specify the maximum file permissions for files in the file system. (For example, a mask of 755 specifies that, by default, the owner should have read, write, and execute permissions for files, but others should only have read and execute permissions. See chmod(1) for more information about octal file modes.) Only the nine low-order bits of *mask* are used. The default mask is taken from the directory on which the file system is being mounted.

SEE ALSO

mount(2), unmount(2), fstab(5), mount(8)

HISTORY

The **mount_ados** utility first appeared in NetBSD 1.0.

BUGS

The 'ados' filesystem currently supports the Amiga fast file system.

The 'ados' filesystem implementation currently is read-only. The **mount_ados** utility silently retries the mount read-only, as if the *ro* option were specified, when it encounters the [EROFS] error.

NAME

mount_cd9660 — mount an ISO-9660 filesystem

SYNOPSIS

mount_cd9660 [**-o** *options*] *special node*

DESCRIPTION

The **mount_cd9660** command attaches the ISO-9660 filesystem residing on the device `special` to the global filesystem namespace at the location indicated by `node`. Both *special* and *node* are converted to absolute paths before use.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. Besides options mentioned in `mount`(8) man page, following cd9660-specific options are supported:

extatt Enable the use of extended attributes.

gens Do not strip version numbers on files and leave the case of the filename alone. (By default, uppercase characters are translated to lowercase, and if there are files with different version numbers on the disk, only the last one will be listed.)

In either case, files may be opened without giving a version number, in which case you get the last one, or by explicitly stating a version number (albeit it's quite difficult to know it, if you are not using the **gens** option), in which case you get the specified version.

nojoliet
Do not make use of Joliet extensions for long filenames which may be present in the filesystem.

Interpretation of Joliet extensions is enabled by default, Unicode file names are encoded into UTF-8.

nomaplcase
File names on cd9660 cdrom without Rock Ridge extension present should be uppercase only. By default, cd9660 recodes file names read from a non-Rock Ridge disk to all lowercase characters. **nomaplcase** turns off this mapping.

norrip Do not use any Rockridge extensions included in the filesystem.

nrr Same as **norrip**. For compatibility with Solaris only.

rrcaseins
Makes all lookups case-insensitive even for CD-ROMs with Rock-Ridge extensions (for Rock-Ridge, default is case-sensitive lookup).

For compatibility with previous releases, following obsolete flags are still recognized:

-e Same as **-o extatt**.

-j Same as **-o nojoliet**.

-g Same as **-o gens**.

-r Same as **-o norrip**.

SEE ALSO

`mount`(2), `unmount`(2), `fstab`(5), `mount`(8), `mscdlabel`(8)

HISTORY

The **mount_cd9660** utility first appeared 4.4BSD. Support for Joliet filesystem appeared in NetBSD 1.4. Options **nomaplcase** and **rrcaseins** were added in NetBSD 1.5. UTF-8 encoding of Unicode file names for Joliet filesystems was added in NetBSD 3.0.

NOTES

For Joliet filesystems, the Unicode file names used to be filtered to ISO-8859-1 character set. This changed in NetBSD 3.0, file names are encoded into UTF-8 now by default. The behaviour is controllable by the *vfs.cd9660.utf8_joliet* sysctl; the former behaviour is available by setting it to 0.

BUGS

For some cdroms the information in the Rock Ridge extension is wrong and the cdrom needs to be mounted with "norrip". A sign that something is wrong is that the `stat`(2) system call returns `EBADF` causing, e.g., "ls -l" to fail with "Bad file descriptor".

The cd9660 filesystem does not support the original "High Sierra" ("CDROM001") format.

POSIX device node mapping is currently not supported.

Version numbers are not stripped if Rockridge extensions are in use. In this case, you have to use the original name of the file as recorded on disk, i.e. use uppercase and append the version number to the file.

There is no ECMA support.

NAME

mount_efs — Mount an EFS file system

SYNOPSIS

mount_efs [**-o** *options*] *special node*

DESCRIPTION

The **mount_efs** command attaches an EFS file system *special* device on to the file system tree at the point *node*. Both *special* and *node* are converted to absolute paths before use.

This command is normally executed by mount(8) at boot time.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma-separated string of options. See the mount(8) man page for possible options and their meanings.

SEE ALSO

mount(2), unmount(2), options(4), fstab(5), mount(8), svhlabel(8)

HISTORY

The **mount_efs** utility first appeared in NetBSD 5.0.

BUGS

Write support is not presently implemented.

EFS file systems are limited to 8 gigabytes in size.

Because of EFS limitations, an EFS file system can't be used with any UID or GID greater than 65535.

NAME

mount_ext2fs — Mount an EXT2FS file system

SYNOPSIS

mount_ext2fs [**-o** *options*] *special node*

DESCRIPTION

The **mount_ext2fs** command attaches an EXT2FS file system *special* device on to the file system tree at the point *node*. Both *special* and *node* are converted to absolute paths before use.

This command is normally executed by mount(8) at boot time.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma-separated string of options. See the mount(8) man page for possible options and their meanings.

SEE ALSO

mount(2), unmount(2), options(4), fstab(5), mount(8)

HISTORY

The **mount_ext2fs** utility first appeared in FreeBSD 2.2.

BUGS

Some EXT2FS-specific options, features or file flags are not supported.

Because of EXT2FS limitations, an EXT2FS file system can't be used with any UID or GID greater than 65535.

NAME

mount_fdesc — mount the file-descriptor file system

SYNOPSIS

mount_fdesc [**-o** *options*] *fdesc mount_point*

DESCRIPTION

The **mount_fdesc** command attaches an instance of the per-process file descriptor namespace to the global filesystem namespace. The conventional mount point is /dev and the filesystem should be union mounted in order to augment, rather than replace, the existing entries in /dev. The directory specified by *mount_point* is converted to an absolute path before use.

This command is normally executed by mount(8) at boot time.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

The contents of the mount point are fd, stderr, stdin, stdout and tty.

fd is a directory whose contents appear as a list of numbered files which correspond to the open files of the process reading the directory. The files /dev/fd/0 through /dev/fd/# refer to file descriptors which can be accessed through the file system. If the file descriptor is open and the mode the file is being opened with is a subset of the mode of the existing descriptor, the call:

```
fd = open("/dev/fd/0", mode);
```

and the call:

```
fd = fcntl(0, F_DUPFD, 0);
```

are equivalent.

The files /dev/stdin, /dev/stdout and /dev/stderr appear as symlinks to the relevant entry in the /dev/fd sub-directory. Opening them is equivalent to the following calls:

```
fd = fcntl(STDIN_FILENO,  F_DUPFD, 0);
fd = fcntl(STDOUT_FILENO, F_DUPFD, 0);
fd = fcntl(STDERR_FILENO, F_DUPFD, 0);
```

Flags to the open(2) call other than O_RDONLY, O_WRONLY and O_RDWR are ignored.

The /dev/tty entry is an indirect reference to the current process's controlling terminal. It appears as a named pipe (FIFO) but behaves in exactly the same way as the real controlling terminal device.

FILES

/dev/fd/#
/dev/stdin
/dev/stdout
/dev/stderr
/dev/tty

SEE ALSO

mount(2), unmount(2), tty(4), fstab(5), mount(8)

HISTORY

The **mount_fdesc** utility first appeared in 4.4BSD.

BUGS

This filesystem may not be NFS-exported.

NAME

mount_ffs, **mount_ufs** — mount a Berkeley Fast File System

SYNOPSIS

mount_ffs [**-o** *options*] *special node*

DESCRIPTION

The **mount_ffs** command attaches the Berkeley Fast File System on the *special* device on to the file system tree at point *node*. Both *special* and *node* are converted to absolute paths before use.

The **mount_ufs** form of the command is meant for backward compatibility only. Fast File Systems should no longer be listed as type "ufs" in fstab(5) and instead should be listed as type "ffs".

This command is normally executed by mount(8) at boot time. The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

SEE ALSO

mount(2), unmount(2), fstab(5), mount(8)

M. McKusick and G. Ganger, "Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast File System", *Proceedings of the FREENIX track: 1999 USENIX Annual Technical Conference*, pp. 1-17, June 1999.

HISTORY

A **mount_ffs** command appeared in NetBSD 1.1.

BUGS

It is possible for a corrupted file system to cause a crash.

NAME

mount_filecore — mount a FILECORE file system

SYNOPSIS

mount_filecore [**-afnR**] [**-g** *gid*] [**-o** *options*] [**-u** *uid*] special node

ORIGIN

The NetBSD FILECORE filesystem is a read only implementation of the filecore file system found in Acorn Computers RISC OS operating system. This operating system is the ROM based operating system found on their ARM 6, ARM7 and StrongARM 110 based RiscPC machines that are supported by the arm32 port. Under RISC OS, filecore will have multiple instantiations for file systems on different block devices such as floppies, IDE discs, SCSI discs etc. and these frequently are considered to be different filesystems e.g. ADFS, IDEFS, SCSIFS etc.

DESCRIPTION

The **mount_filecore** command attaches the FILECORE filesystem residing on the device special to the global filesystem namespace at the location indicated by node. Both *special* and *node* are converted to absolute paths before use. This command is normally executed by mount(8) at boot time, but can be used by any user to mount a FILECORE file system on any directory that they own (provided, of course, that they have appropriate access to the device that contains the file system).

The options are as follows:

-a Give all files world access.

-f Append the filetype to each filename. This option currently has no effect.

-g *gid*
Set the group of the files in the file system to *gid*. The default group is the group of the directory on which the file system is being mounted.

-n Give all files owner access.

-o *options*
Use the specified mount *options*, as described in mount(8).

-R Give all files owner read access.

-u *uid*
Set the owner of the files in the file system to *uid*. The default owner is the owner of the directory on which the file system is being mounted.

SEE ALSO

mount(2), unmount(2), fstab(5), mount(8)

HISTORY

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

CAVEATS

The 'filecore' filesystem currently supports the Acorn filecore file system found on Acorn Computers RiscPC desktop machines with versions of RISC OS up to 3.70.

NAME

mount_hfs — mount an Apple HFS+ File System

SYNOPSIS

mount_hfs [**-o** *options*] *special node*

DESCRIPTION

The **mount_hfs** command attaches the Apple HFS+ File System on the *special* device on to the file system tree at point *node*. Both *special* and *node* are converted to absolute paths before use.

This command is normally executed by mount(8) at boot time. The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

NOTES

Apple disk images (.dmg files) and hybrid CD-ROMs contain multiple partitions with an HFS+ file system on one of them. Use apmlabel(8) to enter it into the disklabel and mount the resulting partition.

SEE ALSO

mount(2), unmount(2), fstab(5), apmlabel(8), mount(8)

HISTORY

The **mount_hfs** utility first appeared in NetBSD 5.0.

AUTHORS

mount_hfs was developed by Yevgeny Binder ⟨yevbee@comcast.net⟩ during the 2005 Google Summer of Code. It was improved and imported into NetBSD by Dieter Baron ⟨dillo@NetBSD.org⟩.

BUGS

HFS+ support is still experimental. Currently, no write support is present. Also, Unicode decomposition is not performed on file names prior to lookup.

NAME

mount_kernfs — mount the /kern file system

SYNOPSIS

mount_kernfs [**-o** *options*] */kern mount_point*

DESCRIPTION

The **mount_kernfs** command attaches an instance of the kernel parameter namespace to the global filesystem namespace. The conventional mount point is `/kern`. The directory specified by *mount_point* is converted to an absolute path before use. This command is normally executed by `mount`(8) at boot time.

The filesystem includes several regular files which can be read, some of which can also be written. The contents of the files is in a machine-independent format, either a string, or an integer in decimal ASCII. Where numbers are returned, a trailing newline character is also added.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the `mount`(8) man page for possible options and their meanings.

FILES

`boottime`	the time at which the system was last booted (decimal ASCII).
`copyright`	kernel copyright message.
`hostname`	the hostname, with a trailing newline. The hostname can be changed by writing to this file. A trailing newline will be stripped from the hostname being written.
`hz`	the frequency of the system clock (decimal ASCII).
`ipsecsa`	the directory that contains IPsec security associations (SA) in `PF_KEY` format. Filenames are SPI in decimal number. The content of files can be inspected by using `setkey`(8).
`ipsecsp`	the directory that contains IPsec security policies in `PF_KEY` format. Filenames are security policy ID in decimal number. The content of files can be inspected by using `setkey`(8).
`loadavg`	the 1, 5 and 15 minute load average in kernel fixed-point format. The final integer is the fix-point scaling factor. All numbers are in decimal ASCII.
`msgbuf`	the kernel message buffer, also read by `syslogd`(8), through the `log` device, and by `dmesg`(8).
`pagesize`	the machine pagesize (decimal ASCII).
`physmem`	the number of pages of physical memory in the machine (decimal ASCII).
`rootdev`	the root device.
`rrootdev`	the raw root device.
`time`	the second and microsecond value of the system clock. Both numbers are in decimal ASCII.
`version`	the kernel version string. The head line for `/etc/motd` can be generated by running: "**sed 1q /kern/version**"

SEE ALSO

`mount`(2), `unmount`(2), `ipsec`(4), `fstab`(5), `dmesg`(8), `mount`(8), `setkey`(8), `syslogd`(8)

HISTORY

The **mount_kernfs** utility first appeared in 4.4BSD.

BUGS

This filesystem may not be NFS-exported.

`lkm`(4) version does not support IPsec-related files/directories.

NAME

mount_lfs — mount a log-structured file system

SYNOPSIS

mount_lfs [**-bdins**] [**-N** *nsegs*] [**-o** *options*] *special node*

DESCRIPTION

The **mount_lfs** command attaches a log-structured file system *special* device on to the file system tree at the point *node*. Both *special* and *node* are converted to absolute paths before use. In addition, the lfs_cleanerd(8) utility is invoked to clean the file system periodically.

This command is normally executed by mount(8) at boot time.

The options are as follows:

-b Instruct the cleaner to count bytes written, rather than segments read, to determine how many segments to clean at once.

-d Run lfs_cleanerd(8) in debug mode.

-i Instruct the cleaner to use filesystem idle time as the criterion for aggressive cleaning, instead of system load.

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

-N *nsegs*
Clean *nsegs* segments (or bytes' worth of segments if **-b** is also specified) at a time.

-n Don't start lfs_cleanerd(8) on the file system.

-s Cause lfs_cleanerd(8) to read data in small chunks when cleaning the file system.

SEE ALSO

mount(2), unmount(2), fstab(5), dump_lfs(8), lfs_cleanerd(8), mount(8), newfs_lfs(8)

Ousterhout and Douglis, "Beating the I/O Bottleneck: A Case for Log-structured File Systems", *Operating Systems Review*, No. 1, Vol. 23, pp. 11-27, 1989, also available as Technical Report UCB/CSD 88/467.

Rosenblum and Ousterhout, "The Design and Implementation of a Log-Structured File System", *ACM SIGOPS Operating Systems Review*, No. 5, Vol. 25, 1991.

Seltzer, "File System Performance and Transaction Support", *PhD Thesis, University of California, Berkeley*, 1992, also available as Technical Report UCB/ERL M92.

Seltzer, Bostic, McKusick and Staelin, "An Implementation of a Log-Structured File System for UNIX", *Proc. of the Winter 1993 USENIX Conf.*, pp. 315-331, 1993.

HISTORY

The **mount_lfs** function first appeared in 4.4BSD.

NAME

mount_mfs — mount a memory based file system

SYNOPSIS

mount_mfs [**-N**] [**-a** *maxcontig*] [**-b** *block-size*] [**-d** *rotdelay*] [**-e** *maxbpg*]
[**-f** *frag-size*] [**-g** *groupname*] [**-i** *bytes-per-inode*]
[**-m** *free-space*] [**-n** *inodes*] [**-o** *options*] [**-p** *permissions*]
[**-s** *size*] [**-u** *username*] [**-V** *verbose*] *special node*

DESCRIPTION

mount_mfs is used to build a file system in virtual memory and then mount it on a specified node. **mount_mfs** exits and the contents of the file system are lost when the file system is unmounted. If **mount_mfs** is sent a signal while running, for example during system shutdown, it will attempt to unmount its corresponding file system. *special* is ignored.

Options with numeric arguments may contain an optional (case-insensitive) suffix:

b	Bytes; causes no modification. (Default)
k	Kilo; multiply the argument by 1024
m	Mega; multiply the argument by 1048576
g	Giga; multiply the argument by 1073741824

The following options define the general layout policies:

-N Causes the memory file system parameters to be printed out without really mounting the memory file system.

-a *maxcontig*
This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see the **-d** option). The default value is 8. See tunefs(8) for more details on how to set this option.

-b *block-size*
The block size of the file system, in bytes. It must be a power of two. The smallest allowable size is 4096 bytes. The default size depends upon the size of the file system:

file system size	*block-size*
< 20 MB	4 KB
< 1024 MB	8 KB
>= 1024 MB	16 KB

-d *rotdelay*
This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. The default is 0 milliseconds. See tunefs(8) for more details on how to set this option.

-e *maxbpg*
This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. The default is about one quarter of the total blocks in a cylinder group. See tunefs(8) for more details on how to set this option.

-f *frag-size*
The fragment size of the file system in bytes. It must be a power of two ranging in value between *block-size*/8 and *block-size*. The optimal *block-size*:*frag-size* ratio is 8:1. Other ratios are possible, but are not recommended, and may produce unpredictable results. The default size depends upon the size of the file system:

file system size	*frag-size*
< 20 MB	0.5 KB
< 1024 MB	1 KB
>= 1024 MB	2 KB

-g *groupname*
This specifies the group name or group id of the root inode of the file system.

-i *bytes-per-inode*
This specifies the density of inodes in the file system. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given. The default is to create an inode for every (4 ∗ *frag-size*) bytes of data space:

file system size	*bytes-per-inode*
< 20 MB	2 KB
< 1024 MB	4 KB
>= 1024 MB	8 KB

-m *free-space*
The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 5%. See `tunefs`(8) for more details on how to set this option.

-n *inodes*
This specifies the number of inodes for the filesystem. If both **-i** and **-n** are specified then **-n** takes precedence.

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the `mount`(8) man page for possible options and their meanings.

-p *permissions*
This specifies the permissions of the root inode of the file system.

-s *size* The size of the file system in sectors. An 's' suffix will be interpreted as the number of sectors (the default). All other suffixes are interpreted as per other numeric arguments, except that the number is converted into sectors by dividing by the default sector size (which is 512 bytes) after suffix interpretation.

-u *username*
This specifies the user name or user id of the root inode of the file system.

-V *verbose*
This controls the amount of information written to stdout:

0	No output
1	Overall size and cylinder group details.
2	A progress bar (dots ending at right hand margin).
3	The first few super-block backup sector numbers are displayed before the progress bar.
4	All the super-block backup sector numbers are displayed (no progress bar).

The default is 0. If **-N** is specifed **mount_mfs** stops before outputting the progress bar.

NOTES

The owner and group ids of the root node of the new file system are set to the effective uid and gid of the user mounting the file system.

EXAMPLES

Mount a 32 MB mfs on /tmp:

```
mount_mfs -s 32m swap /tmp
```

SEE ALSO

`disktab`(5), `fs`(5), `disklabel`(8), `diskpart`(8), `dumpfs`(8), `fsck_ffs`(8), `fsirand`(8), `mount`(8), `newfs`(8), `tunefs`(8)

M. McKusick, W. Joy, S. Leffler, and R. Fabry, "A Fast File System for UNIX,", *ACM Transactions on Computer Systems 2*, 3, pp 181-197, August 1984, (reprinted in the BSD System Manager's Manual).

HISTORY

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

BUGS

The **async** `mount`(8) option is currently disabled in this file system because it causes hangs when writing lots of data. The problem is that MFS needs to allocate pages to clean pages, so if it waits until the last minute to clean pages then there may not be any of them available to do the cleaning.

NAME

mount_msdos — mount an MS-DOS file system

SYNOPSIS

mount_msdos [**-9Gls**] [**-g** *gid*] [**-M** *mask*] [**-m** *mask*] [**-o** *options*] [**-t** *gmtoff*]
[**-u** *uid*] special node

DESCRIPTION

The **mount_msdos** command attaches the MS-DOS filesystem residing on the device special to the global filesystem namespace at the location indicated by node. Both *special* and *node* are converted to absolute paths before use. This command is normally executed by mount(8) at boot time, but can be used by any user to mount an MS-DOS file system on any directory that they own (provided, of course, that they have appropriate access to the device that contains the file system).

Support for FAT16 and VFAT32 as well as long file names is available.

The options are as follows:

-9 Ignore the special Win'95 directory entries even if deleting or renaming a file. This forces **-s**.

-G This option causes the filesystem to be interpreted as an Atari-Gemdos filesystem. The differences to the MSDOS filesystem are minimal and limited to the boot block. This option enforces **-s**.

-g *gid* Set the group of the files in the file system to *gid*. The default group is the group of the directory on which the file system is being mounted.

-l Force listing and generation of Win'95 long filenames and separate creation/modification/access dates.

If neither **-s** nor **-l** are given, **mount_msdos** searches the root directory of the filesystem to be mounted for any existing Win'95 long filenames. If no such entries are found, **-s** is the default. Otherwise **-l** is assumed.

-M *mask* Specify the maximum file permissions for directories in the file system. The value of **-m** is used if it is supplied and **-M** is omitted.

-m *mask* Specify the maximum file permissions for files in the file system. (For example, a mask of 755 specifies that, by default, the owner should have read, write, and execute permissions for files, but others should only have read and execute permissions. See chmod(1) for more information about octal file modes.) Only the nine low-order bits of *mask* are used. The value of **-M** is used if it is supplied and **-m** is omitted. The default mask is taken from the directory on which the file system is being mounted.

-o *options* Use the specified mount *options*, as described in mount(8).

-s Force behaviour to ignore and not generate Win'95 long filenames. See also **-l**.

-t *gmtoff* Set the time zone offset (in seconds) from UTC to *gmtoff*, with positive values indicating east of the Prime Meridian. If not set, the user's current time zone will be used.

-u *uid* Set the owner of the files in the file system to *uid*. The default owner is the owner of the directory on which the file system is being mounted.

EXAMPLES

To remove the 'execute' permission bit for all files, but still keep directories searchable, use:

```
mount_msdos -m 0644 -M 0755 /dev/wd0e /msdos
```

SEE ALSO

`mount`(2), `unmount`(2), `fstab`(5), `mount`(8)

HISTORY

The **`mount_msdos`** utility first appeared in NetBSD 0.9. Its predecessor, the **`mount_pcfs`** utility appeared in NetBSD 0.8, and was abandoned in favor of the more aptly-named **`mount_msdos`**.

BUGS

Compressed partitions are not supported.

The use of the **`-9`** flag could result in damaged filesystems, albeit the damage is in part taken care of by procedures similar to the ones used in Win'95.

The default handling for **`-s`** and **`-l`** will result in empty filesystems to be populated with short filenames only. To generate long filenames on empty DOS filesystems use **`-l`**.

NAME

mount_nfs — mount NFS file systems

SYNOPSIS

mount_nfs [**-23bCcdilPpqsTUX**] [**-a** *maxreadahead*] [**-D** *deadthresh*]
[**-g** *maxgroups*] [**-I** *readdirsize*] [**-L** *leaseterm*] [**-o** *options*]
[**-R** *retrycnt*] [**-r** *readsize*] [**-t** *timeout*] [**-w** *writesize*]
[**-x** *retrans*] *rhost:path node*

DESCRIPTION

The **mount_nfs** command calls the mount(2) system call to prepare and graft a remote NFS file system (rhost:path) on to the file system tree at the mount point *node*. The directory specified by *node* is converted to an absolute path before use. This command is normally executed by mount(8). It implements the mount protocol as described in RFC 1094, Appendix A and *NFS: Network File System Version 3 Protocol Specification*, Appendix I.

The options are:

-2 Use the NFS Version 2 protocol.

-3 Use the NFS Version 3 protocol. The default is to try version 3 first, and fall back to version 2 if the mount fails.

-a *maxreadahead*
Set the read-ahead count to the specified value. This may be in the range of 0 - 4, and determines how many blocks will be read ahead when a large file is being read sequentially. Trying a value greater than 1 for this is suggested for mounts with a large bandwidth * delay product.

-b If an initial attempt to contact the server fails, fork off a child to keep trying the mount in the background. Useful for fstab(5), where the filesystem mount is not critical to multiuser operation.

-C For UDP mount points, do a connect(2). Although this flag increases the efficiency of UDP mounts it cannot be used for servers that do not reply to requests from the standard NFS port number 2049, or for servers with multiple network interfaces. In these cases if the socket is connected and the server replies from a different port number or a different network interface the client will get ICMP port unreachable and the mount will hang.

-c For UDP mount points, do not do a connect(2). This flag is deprecated and connectionless UDP mounts are the default.

-D *deadthresh*
Set the "dead server threshold" to the specified number of round trip timeout intervals. After a "dead server threshold" of retransmit timeouts, "not responding" message is printed to a tty.

-d Turn off the dynamic retransmit timeout estimator. This may be useful for UDP mounts that exhibit high retry rates, since it is possible that the dynamically estimated timeout interval is too short.

-g *maxgroups*
Set the maximum size of the group list for the credentials to the specified value. This should be used for mounts on old servers that cannot handle a group list size of 16, as specified in RFC 1057. Try 8, if users in a lot of groups cannot get response from the mount point.

-I *readdirsize*
Set the readdir read size to the specified value. The value should normally be a multiple of DIRBLKSIZ that is ≤ the read size for the mount.

-i Make the mount interruptible, which implies that file system calls that are delayed due to an unresponsive server will fail with EINTR when a termination signal is posted for the process.

-L *leaseterm*
Ignored. It used to be NQNFS lease term.

-l Used with NFS Version 3 to specify that the **ReaddirPlus**() RPC should be used. This option reduces RPC traffic for cases such as **ls -l**, but tends to flood the attribute and name caches with prefetched entries. Try this option and see whether performance improves or degrades. Probably most useful for client to server network interconnects with a large bandwidth times delay product.

-o *options*
Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

The following NFS specific options are also available:

bg Same as **-b**.

conn Same as **-C**.

deadthresh=⟨*deadthresh*⟩
Same as **-D** *deadthresh*.

dumbtimer
Same as **-d**.

intr Same as **-i**.

leaseterm=⟨*leaseterm*⟩
Same as **-L** *leaseterm*.

maxgrps=⟨*maxgroups*⟩
Same as **-g** *maxgroups*.

mntudp Same as **-U**.

nfsv2 Same as **-2**.

nfsv3 Same as **-3**.

noresvport
Same as **-p**.

nqnfs Same as **-q**.

port=⟨*portnumber*⟩
Use the specified port number for NFS requests. The default is to query the portmapper for the NFS port.

rdirplus
Same as **-l**.

readahead=⟨*maxreadahead*⟩
Same as **-a** *maxreadahead*.

rsize=⟨*readsize*⟩
Same as **--r** *readsize*.

soft Same as **-s**.

tcp Same as **−T**.

timeo=⟨*timeout*⟩
Same as **−t** *timeout*.

wsize=⟨*writesize*⟩
Same as **−w** *writesize*.

−P Use a reserved socket port number. This is the default, and available for backwards compatibility purposes only.

−p Do not use a reserved port number for RPCs. This option is provided only to be able to mimic the old default behavior of not using a reserved port, and should rarely be useful.

−q A synonym of **−3**. It used to specify NQNFS.

−R *retrycnt*
Set the retry count for doing the mount to the specified value. The default is 10000.

−r *readsize*
Set the read data size to the specified value in bytes. It should normally be a power of 2 greater than or equal to 1024.

This should be used for UDP mounts when the "fragments dropped after timeout" value is getting large while actively using a mount point. Use netstat(1) with the **−s** option to see what the "fragments dropped after timeout" value is. See the **mount_nfs −w** option also.

−s A soft mount, which implies that file system calls will fail after *retrans* round trip timeout intervals.

−T Use TCP transport instead of UDP. This is recommended for servers that are not on the same physical network as the client. Not all NFS servers, especially not old ones, support this.

−t *timeout*
Set the initial retransmit timeout to the specified value in 0.1 seconds. May be useful for fine tuning UDP mounts over internetworks with high packet loss rates or an overloaded server. Try increasing the interval if nfsstat(1) shows high retransmit rates while the file system is active or reducing the value if there is a low retransmit rate but long response delay observed. Normally, the -d option should be specified when using this option to manually tune the timeout interval. The default is 3 seconds.

−U Force the mount protocol to use UDP transport, even for TCP NFS mounts. This is necessary for some old BSD servers.

−w *writesize*
Set the write data size to the specified value in bytes.

The same logic applies for use of this option as with the **mount_nfs −r** option, but using the "fragments dropped after timeout" value on the NFS server instead of the client. Note that both the **−r** and **−w** options should only be used as a last ditch effort at improving performance when mounting servers that do not support TCP mounts.

−X Perform 32 <-> 64 bit directory cookie translation for version 3 mounts. This may be need in the case of a server using the upper 32 bits of version 3 directory cookies, and when you are running emulated binaries that access such a filesystem. Native NetBSD binaries will never need this option. This option introduces some overhead.

−x *retrans*
Set the retransmit timeout count for soft mounts to the specified value. The default is 10.

EXAMPLES

The simplest way to invoke **mount_nfs** is with a command like:

```
mount remotehost:/filesystem /localmountpoint
```

or:

```
mount -t nfs remotehost:/filesystem /localmountpoint
```

It is also possible to automatically mount filesystems at boot from your `/etc/fstab` by using a line like:

```
remotehost:/home /home nfs rw 0 0
```

PERFORMANCE

As can be derived from the comments accompanying the options, performance tuning of NFS can be a non-trivial task. Here are some common points to watch:

- Increasing the read and write size with the **−r** and **−w** options respectively will increase throughput if the network interface can handle the larger packet sizes.

  The default size for NFS version 2 is 8K when using UDP, 64K when using TCP.

  The default size for NFS version 3 is platform dependent: on NetBSD/i386, the default is 32K, for other platforms it is 8K. Values over 32K are only supported for TCP, where 64K is the maximum.

  Any value over 32K is unlikely to get you more performance, unless you have a very fast network.

- If the network interface cannot handle larger packet sizes or a long train of back to back packets, you may see low performance figures or even temporary hangups during NFS activity.

  This can especially happen with older Ethernet network interfaces. What happens is that either the receive buffer on the network interface on the client side is overflowing, or that similar events occur on the server, leading to a lot of dropped packets.

  In this case, decreasing the read and write size, using TCP, or a combination of both will usually lead to better throughput. Should you need to decrease the read and write size for all your NFS mounts because of a slow Ethernet network interface (e.g. a USB 1.1 to 10/100 Ethernet network interface), you can use

  ```
  options NFS_RSIZE=value
  options NFS_WSIZE=value
  ```

  in your kernel `config`(1) file to avoid having do specify the sizes for all mounts.

- For connections that are not on the same LAN, and/or may experience packet loss, using TCP is strongly recommended.

ERRORS

Some common problems with **mount_nfs** can be difficult for first time users to understand.

```
mount_nfs: can't access /foo: Permission denied
```

This message means that the remote host, is either not exporting the filesystem you requested, or is not exporting it to your host. If you believe the remote host is indeed exporting a filesystem to you, make sure the `exports`(5) file is exporting the proper directories.

A common mistake is that `mountd`(8) will not export a filesystem with the **−alldirs** option, unless it is a mount point on the exporting host. It is not possible to remotely mount a subdirectory of an exported mount, unless it is exported with the **−alldirs** option.

The following error:

```
NFS Portmap: RPC: Program not registered
```

means that the remote host is not running `mountd`(8). The program `rpcinfo`(8) can be used to determine if the remote host is running nfsd, and mountd by issuing the command:

```
rpcinfo -p remotehostname
```

If the remote host is running nfsd, and mountd, it would display:

```
100005    3   udp    719  mountd
100005    1   tcp    720  mountd
100005    3   tcp    720  mountd
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
```

The error:

```
mount_nfs: can't get net id for host
```

indicates that **`mount_nfs`** cannot resolve the name of the remote host.

SEE ALSO

`nfsstat`(1), `mount`(2), `unmount`(2), `options`(4), `exports`(5), `fstab`(5), `mount`(8), `mountd`(8), `rpcinfo`(8)

NFS: Network File System Protocol specification, RFC 1094, March 1989.

NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GCC and Kerberos V5, RFC 2623, June 1999.

NFS Version 4 Design Considerations, RFC 2624, June 1999.

Authentication Mechanisms for ONC RPC, RFC 2695, September 1999.

CAVEATS

An NFS server shouldn't loopback-mount its own exported file systems because it's fundamentally prone to deadlock.

NAME

mount_ntfs — mount an NTFS file system

SYNOPSIS

mount_ntfs [**-a**] [**-i**] [**-u** *uid*] [**-g** *gid*] [**-m** *mask*] special node

DESCRIPTION

The **mount_ntfs** command attaches the NTFS filesystem residing on the device special to the global filesystem namespace at the location indicated by node. Both *special* and *node* are converted to absolute paths before use. This command is normally executed by mount(8) at boot time, but can be used by any user to mount an NTFS file system on any directory that they own (provided, of course, that they have appropriate access to the device that contains the file system).

The supported NTFS versions include both NTFS4, as used by Microsoft Windows NT 4.0, and NTFS5, as used by Microsoft Windows 2000 and XP.

The options are as follows:

-a Force behaviour to return MS-DOS 8.3 names also on **readdir**().

-i Make name lookup case insensitive for all names except POSIX names.

-u *uid*
Set the owner of the files in the file system to *uid*. The default owner is the owner of the directory on which the file system is being mounted.

-g *gid*
Set the group of the files in the file system to *gid*. The default group is the group of the directory on which the file system is being mounted.

-m *mask*
Specify the maximum file permissions for files in the file system.

FEATURES

NTFS file attributes

NTFS file attributes can be accessed in the following way:

```
foo[[:ATTRTYPE]:ATTRNAME]
```

'ATTRTYPE' is one of identifier listed in $AttrDef file of volume. Default is $DATA. 'ATTRNAME' is an attribute name. Default is none.

Examples:

To get volume name (in Unicode):

```
# cat /mnt/\$Volume:\$VOLUME_NAME
```

To read directory raw data:

```
# cat /mnt/foodir:\$INDEX_ROOT:\$I30
```

Limited support for writing

There is limited writing ability for files. Limitations:

- file must be non-resident
- file must *not* contain any holes (uninitialized areas)

- file can't be compressed

Note that it's not currently possible to create or remove files on NTFS filesystems.

Warning: do not mount NTFS filesystems read-write. The write support is not very useful and is not tested well. It's not safe to write to any file on NTFS; you might damage the filesystem. Unless you want to debug NTFS filesystem code, mount the NTFS filesystem read-only.

SEE ALSO

`mount`(2), `unmount`(2), `fstab`(5), `disklabel`(8), `mbrlabel`(8), `mount`(8)

HISTORY

Support for NTFS first appeared in FreeBSD 3.0. It was ported to NetBSD and first appeared in NetBSD 1.5.

AUTHORS

NTFS kernel implementation, **`mount_ntfs`** and this manual were originally written by Semen Ustimenko ⟨semenu@FreeBSD.org⟩.

The NetBSD port was done by
Christos Zoulas ⟨christos@NetBSD.org⟩ and
Jaromir Dolecek ⟨jdolecek@NetBSD.org⟩.

BUGS

The write support should be enhanced to actually be able to change file size, and to create and remove files and directories. It's not very useful right now.

If the attempt to mount NTFS gives you an error like this:

```
# mount -t ntfs /dev/wd0k /mnt
mount_ntfs: /dev/wd0k on /mnt: Invalid argument
```

make sure that appropriate partition has correct entry in the disk label, particularly that the partition offset is correct. If the NTFS partition is the first partition on the disk, the offset should be '63' on i386 (see `disklabel`(8)). `mbrlabel`(8) could help you to set up the disk label correctly.

If the NTFS partition is marked as `dynamic` under Microsoft Windows XP, it won't be possible to access it under NetBSD anymore.

NAME

mount_null — mount a loopback filesystem sub-tree; demonstrate the use of a null file system layer

SYNOPSIS

mount_null [**-o** *options*] *target mount-point*

DESCRIPTION

The **mount_null** command creates a null layer, duplicating a sub-tree of the file system name space under another part of the global file system namespace. This allows existing files and directories to be accessed using a different pathname.

The primary differences between a virtual copy of the filesystem and a symbolic link are that getcwd(3) functions correctly in the virtual copy, and that other filesystems may be mounted on the virtual copy without affecting the original. A different device number for the virtual copy is returned by stat(2), but in other respects it is indistinguishable from the original.

The **mount_null** filesystem differs from a traditional loopback file system in two respects: it is implemented using a stackable layers technique, and its "null-nodes" stack above all lower-layer vnodes (not just above directory vnodes).

Both *target* and *mount-point* are converted to absolute paths before use.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

The null layer has two purposes. First, it serves as a demonstration of layering by providing a layer which does nothing. Second, the null layer can serve as a prototype layer. Since it provides all necessary layer framework, new file system layers can be created very easily by starting with a null layer.

The remainder of this man page examines the null layer as a basis for constructing new layers.

INSTANTIATING NEW NULL LAYERS

New null layers are created with **mount_null**. **mount_null** takes two arguments, the pathname of the lower vfs (target-pn) and the pathname where the null layer will appear in the namespace (mount-point-pn). After the null layer is put into place, the contents of target-pn subtree will be aliased under mount-point-pn.

OPERATION OF A NULL LAYER

The null layer is the minimum file system layer, simply passing all possible operations to the lower layer for processing there. The majority of its activity centers on the bypass routine, through which nearly all vnode operations pass.

The bypass routine accepts arbitrary vnode operations for handling by the lower layer. It begins by examining vnode operation arguments and replacing any null-nodes by their lower-layer equivalents. It then invokes the operation on the lower layer. Finally, it replaces the null-nodes in the arguments and, if a vnode is returned by the operation, stacks a null-node on top of the returned vnode.

Although bypass handles most operations, *vop_getattr*, *vop_inactive*, *vop_reclaim*, and *vop_print* are not bypassed. *vop_getattr* must change the fsid being returned. *vop_inactive* and vop_reclaim are not bypassed so that they can handle freeing null-layer specific data. *vop_print* is not bypassed to avoid excessive debugging information.

INSTANTIATING VNODE STACKS

Mounting associates the null layer with a lower layer, in effect stacking two VFSes. Vnode stacks are instead created on demand as files are accessed.

The initial mount creates a single vnode stack for the root of the new null layer. All other vnode stacks are created as a result of vnode operations on this or other null vnode stacks.

New vnode stacks come into existence as a result of an operation which returns a vnode. The bypass routine stacks a null-node above the new vnode before returning it to the caller.

For example, imagine mounting a null layer with

```
mount_null /usr/include /dev/layer/null
```

Changing directory to `/dev/layer/null` will assign the root null-node (which was created when the null layer was mounted). Now consider opening `sys`. A vop_lookup would be done on the root null-node. This operation would bypass through to the lower layer which would return a vnode representing the UFS `sys`. null_bypass then builds a null-node aliasing the UFS `sys` and returns this to the caller. Later operations on the null-node `sys` will repeat this process when constructing other vnode stacks.

CREATING OTHER FILE SYSTEM LAYERS

One of the easiest ways to construct new file system layers is to make a copy of the null layer, rename all files and variables, and then begin modifying the copy. `sed`(1) can be used to easily rename all variables.

The umap layer is an example of a layer descended from the null layer.

INVOKING OPERATIONS ON LOWER LAYERS

There are two techniques to invoke operations on a lower layer when the operation cannot be completely bypassed. Each method is appropriate in different situations. In both cases, it is the responsibility of the aliasing layer to make the operation arguments "correct" for the lower layer by mapping any vnode arguments to the lower layer.

The first approach is to call the aliasing layer's bypass routine. This method is most suitable when you wish to invoke the operation currently being handled on the lower layer. It has the advantage that the bypass routine already must do argument mapping. An example of this is *null_getattrs* in the null layer.

A second approach is to directly invoke vnode operations on the lower layer with the *VOP_OPERATIONNAME* interface. The advantage of this method is that it is easy to invoke arbitrary operations on the lower layer. The disadvantage is that vnode arguments must be manually mapped.

SEE ALSO

`mount`(8)

UCLA Technical Report CSD-910056, *Stackable Layers: an Architecture for File System Development*.

HISTORY

The **`mount_null`** utility first appeared in 4.4BSD.

NAME

mount_overlay — mount an overlay filesystem; demonstrate the use of an overlay file system layer

SYNOPSIS

mount_overlay [**-o** *options*] */overlay mount-point*

DESCRIPTION

The **mount_overlay** command creates an overlay layer, interposing the overlay filesystem between the over-mounted file store and future pathname lookups.

A different device number for the virtual copy is returned by stat(2), but in other respects it is indistinguishable from the original.

The **mount_overlay** filesystem differs from the null filesystem in that the **mount_overlay** filesystem does not replicate the sub-tree, it places itself between the sub-tree and all future access.

The overlay layer has two purposes. First, it serves as a demonstration of layering by providing a layer which does nothing other than insert itself over the over-mounted file system. Second, the overlay layer can serve as a prototype layer. Since it provides all necessary layer framework, new file system layers which need to block access to the overlayed file system can be created very easily by starting with an overlay layer.

The internal operation of the overlay layer is identical to that of the null layer. See its documentation for details.

SEE ALSO

mount(8), mount_null(8)

UCLA Technical Report CSD-910056, *Stackable Layers: an Architecture for File System Development*.

HISTORY

The **mount_overlay** utility first appeared in NetBSD 1.5.

NAME

mount_portal — mount the portal daemon

SYNOPSIS

mount_portal [**-o** *options*] */etc/portal.conf mount_point*

DESCRIPTION

The **mount_portal** command attaches an instance of the portal daemon to the global filesystem namespace. The conventional mount point is `/p`. The directory specified by *mount_point* is converted to an absolute path before use. This command is normally executed by `mount`(8) at boot time.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the `mount`(8) man page for possible options and their meanings.

The portal daemon provides an *open* service. Objects opened under the portal mount point are dynamically created by the portal daemon according to rules specified in the named configuration file. Using this mechanism allows descriptors such as sockets to be made available in the filesystem namespace.

The portal daemon works by being passed the full pathname of the object being opened. The daemon creates an appropriate descriptor according to the rules in the configuration file, and then passes the descriptor back to the calling process as the result of the open system call.

NAMESPACE

By convention, the portal daemon divides the namespace into sub-namespaces, each of which handles objects of a particular type.

Currently, four sub-namespaces are implemented: `tcp`, `fs`, `rfilter` and `wfilter`. The `tcp` namespace takes a hostname and a port (slash separated) and creates an open TCP/IP connection. The `fs` namespace opens the named file, starting back at the root directory. This can be used to provide a controlled escape path from a chrooted environment.

The `rfilter` and `wfilter` namespaces open a pipe to a process, typically a data-filter such as compression or decompression programs. The `rfilter` namespace opens a read-only pipe, while the `wfilter` namespace opens a write-only pipe. See the **EXAMPLES** section below for more examples.

CONFIGURATION FILE

The configuration file contains a list of rules. Each rule takes one line and consists of two or more whitespace separated fields. A hash (“#”) character causes the remainder of a line to be ignored. Blank lines are ignored.

The first field is a pathname prefix to match against the requested pathname. If a match is found, the second field tells the daemon what type of object to create. Subsequent fields are passed to the creation function.

The `rfilter` and `wfilter` namespaces have additional meanings for the remaining fields. The third field specifies a prefix that is to be stripped off of the passed name before passing it on to the pipe program. If the prefix does not match, no stripping is performed. The fourth argument specifies the program to use for the pipe. Any remaining fields are passed to the pipe program. If the string “`%s`” exists within these remaining fields, it will be replaced by the path after stripping is performed. If there is no field after the program name, “`%s`” will be assumed, to maintain similarity with the `tcp` and `fs` namespaces.

FILES

```
/p/*
```

EXAMPLES

A tutorial and several examples are provided in `/usr/share/examples/mount_portal`. The following is an example configuration file.

```
# @(#)portal.conf      5.1 (Berkeley) 7/13/92
tcp/           tcp tcp/
fs/            file fs/
echo/          rfilter echo/   echo %s
echo_nostrip/  rfilter nostrip echo %s
echo_noslash     rfilter echo_noslash    echo %s
gzcat/         rfilter gzcat/ gzcat %s
gzip/          wfilter gzip/   gzip > %s
gzip9/         wfilter gzip9/  gzip -9 > %s
ftp/           rfilter ftp/    ftp -Vo - %s
ftp://         rfilter nostrip ftp -Vo - %s
http://        rfilter nostrip ftp -Vo - %s
bzcat/         rfilter bzcat/  bzcat %s
nroff/         rfilter nroff/  nroff -man %s
```

As is true with many other filesystems, a weird sense of humor is handy.

Notice that after the keynames, like nroff/ and bzcat/, we typically use another slash. In reality, the **mount_portal** process changes directory to `/`, which makes the subsequent slash unnecessary. However, the extra slash provides a visual hint that we are not operating on an ordinary file. An alternative would be to change the configuration file to something like:

```
nroff%  rfilter nroff%  nroff -man
```

One might then use

```
less /p/nroff%/usr/share/man/man8/mount_portal.8
```

SEE ALSO

`mount`(2), `unmount`(2), `fstab`(5), `mount`(8)

HISTORY

The **mount_portal** utility first appeared in 4.4BSD. The `rfilter` and `wfilter` capabilities first appeared in NetBSD 1.5.

BUGS

This filesystem may not be NFS-exported.

NAME

mount_procfs — mount the process file system

SYNOPSIS

mount_procfs [**-o** *options*] /proc mount_point

DESCRIPTION

The **mount_procfs** command attaches an instance of the process namespace to the global filesystem namespace. The conventional mount point is `/proc`. The directory specified by *mount_point* is converted to an absolute path before use. This command is normally executed by `mount`(8) at boot time.

The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the `mount`(8) man page for possible generic options and their meanings. Currently, one procfs-specific option is defined, the **linux** option. This option enables a few extra features that are compatible with the proc filesystem as implemented in Linux. This option can be used if you run Linux binaries that need Linux-specific features in the proc filesystem (see also `compat_linux`(8)).

The root of the process filesystem contains an entry for each active process. These processes are visible as a directory whose name is the process' pid. In addition, the special entries `curproc` and `self` reference the current process. The `self` symlink appears for compatibility with the Linux procfs implementation.

Each directory contains several files.

`cmdline`
This file is readonly and returns null-terminated strings corresponding to the process' command line arguments. For a system or zombie process, this file contains only a string with the name of the process.

`ctl` a writeonly file which supports a variety of control operations. Control commands are written as strings to the `ctl` file. The control commands are:

attach	stops the target process and arranges for the sending process to become the debug control process.
detach	continue execution of the target process and remove it from control by the debug process.
run	continue running the target process until a signal is delivered, a breakpoint is hit, or the target process exits.
step	single step the target process, with no signal delivery.
wait	wait for the target process to stop. The target process must be stopped before any of the run, step, or signal commands are allowed.

The string can also be the name of a signal, lower case and without the `SIG` prefix, in which case that signal is delivered to the process (see `sigaction`(2)).

`cwd` A symbolic link that points to the current working directory of the process. If the target process's current working directory is not available or is not at or below the current process's root directory, this link will point to "/".

`fd/#` File descriptors which can be accessed through the file system. See `fd`(4) for more information.

`file` A reference to the vnode from which the process text was read. This can be used to gain access to the process' symbol table, or to start another copy of the process.

`map` A map of the process' virtual memory.

`maps` A map of the process' virtual memory in a form like the proc filesystem as implemented in Linux. Note that the paths corresponding to file backed mappings will not be present unless the kernel was built with the NAMECACHE_ENTER_REVERSE option.

`mem` The complete virtual memory image of the process. Only those addresses which exist in the process can be accessed. Writes to this file modify the process. Writes to the text segment normally remain private to the process, since the text segment is mapped with MAP_PRIVATE; however, this is not guaranteed.

`note` Not implemented.

`notepg` Not implemented.

`regs` Allows read and write access to the process' register set. This file contains a binary data structure `struct regs` defined in `<machine/reg.h>`. `regs` can only be written when the process is stopped.

`fpregs` The floating point registers as defined by `struct fpregs` in `<machine/reg.h>`. `fpregs` is only implemented on machines which have distinct general purpose and floating point register sets.

`root` A symbolic link that points to the root directory of the process. If the target process's root directory is not available or is not at or below the current process's root directory, this link will point to "/".

`status` The process status. This file is readonly and returns a single line containing multiple space-separated fields as follows:

- command name
- process id
- parent process id
- process group id
- session id
- *major,minor* of the controlling terminal, or `-1,-1` if there is no controlling terminal.
- a list of process flags: `ctty` if there is a controlling terminal, `sldr` if the process is a session leader, `noflags` if neither of the other two flags are set.
- the process start time in seconds and microseconds, comma separated.
- the user time in seconds and microseconds, comma separated.
- the system time in seconds and microseconds, comma separated.
- the wait channel message
- the process credentials consisting of the effective user id and the list of groups (whose first member is the effective group id) all comma separated.

In a normal debugging environment, where the target is fork/exec'd by the debugger, the debugger should fork and the child should stop itself (with a self-inflicted `SIGSTOP` for example). The parent should issue a `wait` and then an `attach` command via the appropriate `ctl` file. The child process will receive a `SIGTRAP` immediately after the call to exec (see `execve`(2)).

FILES

```
/proc/#
/proc/#/cmdline
/proc/#/ctl
```

```
/proc/#/cwd
/proc/#/exe
/proc/#/file
/proc/#/fpregs
/proc/#/map
/proc/#/maps
/proc/#/mem
/proc/#/note
/proc/#/notepg
/proc/#/regs
/proc/#/root
/proc/#/status
/proc/curproc
/proc/self
```

If the **linux** mount option is used, the following files are also available:

```
/proc/#/stat
/proc/cpuinfo
/proc/devices
/proc/meminfo
/proc/mounts
/proc/uptime
```

SEE ALSO

mount(2), sigaction(2), unmount(2)

HISTORY

The **mount_procfs** utility first appeared in 4.4BSD.

BUGS

This filesystem may not be NFS-exported since most of the functionality of procfs requires that state be maintained.

NAME

mount_psshfs — sshfs implementation for puffs

SYNOPSIS

mount_psshfs [*options*] *user@host[:path]* *mount_point*

DESCRIPTION

The **mount_psshfs** utility can be used to mount a file system using the ssh sftp subprotocol, making a remote directory hierarchy appear in the local directory tree. This functionality is commonly known as *sshfs*.

The mandatory parameters are the target host name and local mount point. The target host parameter can optionally contain a username whose credentials will be used by the remote sshd, and a relative or absolute path for the remote mount point's root. If no user is given, the credentials of the user issuing the mount command are used. If no path is given, the user's home directory on the remote machine will be used.

The following command line options are available:

−e Makes the mounted file system NFS exportable. If this option is used, it is very important to understand that **mount_psshfs** can not provide complete support for NFS due to the limitations in the backend. Files are valid only for the time that **mount_psshfs** is running and in the event of e.g. a server crash, all client retries to access files will fail.

−F *configfile*
Pass a configuration file to `ssh`(1). This will make it ignore the system-wide `/etc/ssh/ssh_config` configuration file and use `configfile` instead of `~/.ssh/config`.

−o *[no]option*
This flag can be used to give standard mount options and options to puffs.

−O *sshopt=value*
Pass an option to `ssh`(1), for example **−O** *Port=22*. For a list of valid options, see `ssh_config`(5).

−r *max_reads*
Limits maximum outstanding read requests for each node to *max_reads*. This can be used to improve interactive performance on low-bandwidth links when also performing bulk data reads.

−s This flag can be used to make the program stay on top. The default is to detach from the terminal and run in the background.

−t *timeout*
By default **mount_psshfs** caches directory contents and node attributes for 30 seconds before re-fetching from the server to check if anything has changed on the server. This option is used to adjust the timeout period to *timeout* seconds. A value 0 means the cache is never valid and −1 means it is valid indefinitely. It is possible to force a re-read regardless of timeout status by sending `SIGHUP` to the **mount_psshfs** process.

Note: the file system will still free nodes when requested by the kernel and will lose all cached information in doing so. How frequently this happens depends on system activity and the total number of available vnodes in the system (kern.maxvnodes).

EXAMPLES

The following example illustrates how to mount the directory */usr* on server *bigiron* as user *abc* on local directory */mnt* with ssh transport compression enabled:

```
mount_psshfs -O Compression=yes abc@bigiron:/usr /mnt
```

It is possible to use `fstab`(5) for psshfs mounts, with SSH public key authentication:

```
abc@bigiron:/usr                          /mnt                          psshfs
rw,noauto,-O=BatchMode=yes,-O=IdentityFile=/root/.ssh/id_rsa,-t=-1
```

SEE ALSO

`sftp`(1), `puffs`(3), `puffs`(4), `fstab`(5), `ssh_config`(5), `mount`(8), `sshd`(8)

HISTORY

The **`mount_psshfs`** utility first appeared in NetBSD 5.0.

CAVEATS

Permissions are not handled. Do not expect the file system to behave except for a single user.

NAME

mount_ptyfs — mount the /dev/pts file system

SYNOPSIS

mount_ptyfs [**-g** *group*|*gid*] [**-m** *mode*] [**-o** *options*] *ptyfs* *mount_point*

DESCRIPTION

The **mount_ptyfs** command attaches an instance of the pseudo-terminal device filesystem to the global filesystem namespace. The conventional mount point is /dev/pts. The directory specified by *mount_point* is converted to an absolute path before use. This command is normally executed by mount(8) at boot time.

The filesystem contains pseudo-terminal slave device nodes which are allocated dynamically via ptm(4), or they are already open via traditional BSD style ptys.

The options are as follows:

-g *group*|*gid*
Specify the group ownership of the slave pseudo-tty.

-m *mode*
Specify the default *mode* of the slave pseudo-tty.

-o Options are specified with a **-o** flag followed by a comma separated string of options.

mount_ptyfs specific options are group which corresponds to **-g**, and mode which corresponds to **-m**. See the mount(8) man page for possible options and their meanings.

FILES

n The nth pseudo-terminal device in use.

SEE ALSO

mount(2), unmount(2), ptm(4), fstab(5), mount(8)

HISTORY

The **mount_ptyfs** utility first appeared in NetBSD 3.0.

BUGS

This filesystem may not be NFS-exported. This filesystem is experimental.

NAME

mount_smbfs — mount a shared resource from an SMB/CIFS file server

SYNOPSIS

mount_smbfs [**-E** *cs1*:*cs2*] [**-I** *host*] [**-L** *locale*] [**-M** *crights*:*srights*] [**-N**]
[**-O** *cowner*:*cgroup*/*sowner*:*sgroup*] [**-R** *retrycount*] [**-T** *timeout*]
[**-W** *workgroup*] [**-c** *case*] [**-d** *mode*] [**-f** *mode*] [**-g** *gid*] [**-n** *opt*]
[**-u** *uid*] //*user*@*server*/*share* *node*

DESCRIPTION

The **mount_smbfs** command mounts a share from a remote server using SMB/CIFS protocol.

The options are as follows:

-E *cs1*:*cs2*
: Specifies local (*cs1*) and server's (*cs2*) character sets.

-I *host*
: Do not use NetBIOS name resolver and connect directly to *host*, which can be either a valid DNS name or an IP address.

-L *locale*
: Use *locale* for lower/upper case conversion routines. Set the locale for case conversion. By default, **mount_smbfs** tries to use an environment variable LC_* to determine it.

-M *crights*:*srights*
: Assign access rights to the newly created connection.

-N
: Do not ask for a password. At run time, **mount_smbfs** reads the ~/.nsmbrc file for additional configuration parameters and a password. If no password is found, **mount_smbfs** prompts for it.

-O *cowner*:*cgroup*/*sowner*:*sgroup*
: Assign owner/group attributes to the newly created connection.

-R *retrycount*
: How many retries should be done before the SMB requester decides to drop the connection.

-T *timeout*
: Timeout in seconds for each request.

-W *workgroup*
: This option specifies the workgroup to be used in the authentication request.

-c *case*
: Set a *case* option which affects name representation. *case* can be one of the following:

Value	*Meaning*
l	All existing file names are converted to lower case. Newly created file gets a lower case.
u	All existing file names are converted to upper case. Newly created file gets an upper case.

-f *mode*, **-d** *mode*
: Specify permissions that should be assigned to files and directories. The values must be specified as octal numbers. Default value for the file mode is taken from mount point, default value for the directory mode adds execute permission where the file mode gives read permission.

  Note that these permissions can differ from the rights granted by SMB server.

−u *uid*, **−g** *gid*
: User ID and group ID assigned to files. The default are owner and group IDs from the directory where the volume is mounted.

//*user*@*server*/*share*
: The **mount_smbfs** command will use *server* as the NetBIOS name of remote computer, *user* as the remote user name and *share* as the resource name on a remote server. If your connections are refused, try using the **−I** option and use a server name of '*SMBSERVER'.

node
: Path to mount point.

FILES

`/etc/nsmb.conf`	System wide parameters for smbfs mounts.
`~/.nsmbrc`	Keeps static parameters for connections and other information. See `/usr/share/examples/smbfs/dot.nsmbrc` for details.

EXAMPLES

The following example illustrates how to connect to SMB server *SAMBA* as user *GUEST*, and mount shares *PUBLIC* and *TMP*:

```
mount_smbfs -I samba.mydomain.com //guest@samba/public /smb/public
mount_smbfs -I 192.168.20.3 -E koi8-r:cp866 //guest@samba/tmp /smb/tmp
```

If you keep on getting "Connection reset by peer" errors, try:

```
mount_smbfs -N -I 10.0.0.4 //'*SMBSERVER'/tmp /smb/tmp
```

It is possible to use `fstab`(5) for smbfs mounts:

```
//guest@samba/public    /smb/public     smbfs  rw,noauto 0   0
```

SEE ALSO

`mount`(8)

HISTORY

Support for SMBFS first appeared in FreeBSD 4.4. It has been ported to NetBSD and first appeared in NetBSD 2.0.

AUTHORS

Boris Popov ⟨bp@butya.kz⟩, ⟨bp@FreeBSD.org⟩. NetBSD port done by
Matt Debergalis ⟨deberg@NetBSD.org⟩, and
Jaromir Dolecek ⟨jdolecek@NetBSD.org⟩.

NAME

mount_sysctlfs — mount sysctl namespace as a directory hierarchy

SYNOPSIS

mount_sysctlfs [**-o** *mntopts*] *sysctlfs* *mount_point*

DESCRIPTION

The **mount_sysctlfs** program provides the sysctl(8) hierarchy through the file system namespace. It is possible to browse the tree, query node values and modify them.

SEE ALSO

puffs(3), puffs(4), mount(8), sysctl(8)

HISTORY

The **mount_sysctlfs** utility first appeared in NetBSD 5.0.

CAVEATS

Tree nodes of type "struct" are not accessible through this interface.

NAME

mount_sysvbfs — mount a System V Boot File System

SYNOPSIS

mount_sysvbfs [**-o** *options*] *special node*

DESCRIPTION

The **mount_sysvbfs** command attaches the System V Boot File System on the *special* device on to the file system tree at point *node*. Both *special* and *node* are converted to absolute paths before use.

This command is normally executed by mount(8) at boot time. The options are as follows:

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

SEE ALSO

mount(2), unmount(2), fstab(5), mount(8)

HISTORY

A **mount_sysvbfs** command first appeared in NetBSD 4.0.

BUGS

The sysvbfs support is still experimental and there are few sanity checks, so it is possible for a corrupted file system to cause a crash.

NAME

mount_tmpfs — mount an efficient memory file system

SYNOPSIS

mount_tmpfs [**-g** *group*] [**-m** *mode*] [**-n** *nodes*] [**-o** *options*] [**-s** *size*] [**-u** *user*] *tmpfs* *mount_point*

DESCRIPTION

The **mount_tmpfs** command attaches an instance of the efficient memory file system to the global file system namespace. The *tmpfs* parameter only exists for compatibility with the other mount commands and is ignored. The directory specified by *mount_point* is converted to an absolute path before use and its attributes (owner, group and mode) are inherited unless explicitly overriden by the options described below.

The following options are supported:

-g *group* Specifies the group name or GID of the root inode of the file system. Defaults to the mount point's GID.

-m *mode* Specifies the mode (in octal notation) of the root inode of the file system. Defaults to the mount point's mode.

-n *nodes* Specifies the maximum number of nodes available to the file system. If not specified, the file system chooses a reasonable maximum given its size at mount time, which can be limited with **-s**.

-o *options*
Options are specified with a **-o** flag followed by a comma-separated string of options. See the mount(8) man page for possible options and their meanings.

-s *size* Specifies the total file system size in bytes. If zero is given (the default), the available amount of memory (including main memory and swap space) will be used. Note that four megabytes are always reserved for the system and cannot be assigned to the file system.

-u *user* Specifies the user name or UID of the root inode of the file system. Defaults to the mount point's UID.

Every option that accepts a numerical value as its argument can take a trailing 'b' to indicate bytes (the default), a trailing 'k' to indicate kilobytes, a trailing 'M' to indicate megabytes or a trailing 'G' to indicate gigabytes. Note that both lowercase and uppercase forms of these letters are allowed.

EXAMPLES

The following command mounts a tmpfs instance over the /tmp directory, inheriting its owner, group and mode settings:

```
mount -t tmpfs tmpfs /tmp
```

The following command mounts a tmpfs instance over the /mnt directory, setting a 20 megabytes limit in space, owned by the 'joe' user and belonging to the 'users' group, with a restricted 0700 mode:

```
mount -t tmpfs -o -s20M -o -ujoe -o -gusers -o -m0700 tmpfs /mnt
```

See /usr/share/examples/fstab/fstab.ramdisk for some examples on how to add tmpfs entries to /etc/fstab.

SEE ALSO

fstab(5), mount(8)

HISTORY

The **mount_tmpfs** utility first appeared in NetBSD 4.0.

BUGS

File system meta-data is not pageable. If there is not enough main memory to hold this information, the system may become unstable or very unresponsive because it will not be able to allocate required memory. A malicious user could trigger this condition if he could create lots of files inside a size-unbounded tmpfs file system. Limiting the number of nodes per file system (**-n**) will prevent this; the default value for this setting is also often adjusted to an adequate value to resolve this.

NAME

mount_udf — mount an UDF file system

SYNOPSIS

mount_udf [**-g** *gid*] [**-o** *options*] [**-t** *gmtoff*] [**-s** *session*] [**-u** *uid*] *special node*

DESCRIPTION

The **mount_udf** command attaches the UDF file system residing on the specified *special* device node on the location indicated with *node*.

Anonymous files stored on the UDF disc will be represented and saved in the specified uid:gid pair. If unspecified, it will default to nobody:nobody. Both uid and gid can be either specified with their names as with their numerical equivalents.

-g *gid* Set the group of anonymous files on the file system. The default group is the nobody group.

-o *options*
Use the specified mount *options* as specified in `mount`(8).

-s *session*
Select the session *session* to be mounted instead of the default last one. Positive *session* values indicate an absolute session number. Negative *session* values are relative to the last session found on the disc. Note that this option only makes sense when mounting sequential recordable media like CD-R and DVD∗R.

-t *gmtoff* Set the time zone offset (in seconds) from UTC to *gmtoff*, with positive values indicating east of the Prime Meridian. If not set, the user's current time zone will be used.

-u *uid* Set the owner of anonymous files on the file system. The default owner is the user nobody.

SEE ALSO

`mount`(2), `vnd`(4), `fstab`(5), `mount`(8), `umount`(8), `vnconfig`(8)

NOTES

UDF is a file system defined by the OSTA standardization group and is tailored for data interchange on optical discs (like CDs and DVDs) between different operating systems. Its also more and more common on other media like Compact Flash (CF) cards.

Currently only read access is supported for all media types that CD/DVD type drives can recognise including DVD-RAM and BluRay drives. Write access is planned and in preparation.

Implemented and tested media types are CD-ROM, CD-R, CD-RW, CD-MRW, DVD-ROM, DVD∗R, DVD∗RW, DVD+MRW but the same code can also read DVD-RAM, HD-DVD and BluRay discs. Discs created and written by UDFclient, Nero's InCD, and Roxio's DirectCD/Drag2Disc can be read without problems. Both open and closed media are supported so there is no need to close discs or sessions.

All current UDF versions up to version 2.60 are supported.

Hard disk partitions and `vnd`(4) devices may also be mounted. Note when mounting a `vnd`(4) device it might be necessary to specify the file image sector size in the `geomspec` when creating the `vnd`(4) device or the disc sector size will be used.

BUGS

Due to lack of test media, versions 2.50 and 2.60 are prepared but not implemented and tested yet.

NAME

mount_umap — user and group ID remapping file system layer

SYNOPSIS

mount_umap [**-o** *options*] **-g** *gid-mapfile* **-u** *uid-mapfile target mount-point*

DESCRIPTION

The **mount_umap** command is used to mount a sub-tree of an existing file system that uses a different set of uids and gids than the local system. Such a file system could be mounted from a remote site via NFS, a local file system on removable media brought from some foreign location that uses a different user/group database, or could be a local file system for another operating system which does not support Unix-style user/group IDs, or which uses a different numbering scheme.

Both *target* and *mount-point* are converted to absolute paths before use.

The options are as follows:

-g *gid-mapfile*
: Use the group ID mapping specified in *gid-mapfile*. This flag is required.

-o
: Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

-u *uid-mapfile*
: Use the user ID mapping specified in *uid-mapfile*. This flag is required.

The **mount_umap** command uses a set of files provided by the user to make correspondences between uids and gids in the sub-tree's original environment and some other set of ids in the local environment. For instance, user smith might have uid 1000 in the original environment, while having uid 2000 in the local environment. The **mount_umap** command allows the subtree from smith's original environment to be mapped in such a way that all files with owning uid 1000 look like they are actually owned by uid 2000.

target should be the current location of the sub-tree in the local system's name space. *mount-point* should be a directory where the mapped subtree is to be placed. *uid-mapfile* and *gid-mapfile* describe the mappings to be made between identifiers.

The format of the user and group ID mapping files is very simple. The first line of the file is the total number of mappings present in the file. The remaining lines each consist of two numbers: the ID in the mapped subtree and the ID in the original subtree.

For example, to map uid 1000 in the original subtree to uid 2000 in the mapped subtree:

```
1
2000 1000
```

For user IDs in the original subtree for which no mapping exists, the user ID will be mapped to the user "nobody". For group IDs in the original subtree for which no mapping exists, the group ID will be mapped to the group "nobody".

There is a limit of 64 user ID mappings and 16 group ID mappings.

The mapfiles can be located anywhere in the file hierarchy, but they must be owned by root, and they must be writable only by root. **mount_umap** will refuse to map the sub-tree if the ownership or permissions on these files are improper. It will also report an error if the count of mappings in the first line of the map files is not correct.

SEE ALSO

`mount`(8), `mount_null`(8)

HISTORY

The **`mount_umap`** utility first appeared in 4.4BSD.

BUGS

The implementation is not very sophisticated.

NAME

mount_union — mount union filesystems

SYNOPSIS

mount_union [**-b**] [**-o** *options*] *directory uniondir*

DESCRIPTION

The **mount_union** command attaches *directory* above *uniondir* in such a way that the contents of both directory trees remain visible. By default, *directory* becomes the *upper* layer and *uniondir* becomes the *lower* layer.

Both *directory* and *uniondir* are converted to absolute paths before use.

The options are as follows:

-b Invert the default position, so that *directory* becomes the lower layer and *uniondir* becomes the upper layer. However, *uniondir* remains the mount point.

-o Options are specified with a **-o** flag followed by a comma separated string of options. See the mount(8) man page for possible options and their meanings.

Filenames are looked up in the upper layer and then in the lower layer. If a directory is found in the lower layer, and there is no entry in the upper layer, then a *shadow* directory will be created in the upper layer. It will be owned by the user who originally did the union mount, with mode "rwxrwxrwx" (0777) modified by the umask in effect at that time.

If a file exists in the upper layer then there is no way to access a file with the same name in the lower layer. If necessary, a combination of loopback and union mounts can be made which will still allow the lower files to be accessed by a different pathname.

Except in the case of a directory, access to an object is granted via the normal filesystem access checks. For directories, the current user must have access to both the upper and lower directories (should they both exist).

Requests to create or modify objects in *uniondir* are passed to the upper layer with the exception of a few special cases. An attempt to open for writing a file which exists in the lower layer causes a copy of the *entire* file to be made to the upper layer, and then for the upper layer copy to be opened. Similarly, an attempt to truncate a lower layer file to zero length causes an empty file to be created in the upper layer. Any other operation which would ultimately require modification to the lower layer fails with EROFS.

The union filesystem manipulates the namespace, rather than individual filesystems. The union operation applies recursively down the directory tree now rooted at *uniondir*. Thus any filesystems which are mounted under *uniondir* will take part in the union operation. This differs from the *union* option to mount(8) which only applies the union operation to the mount point itself, and then only for lookups.

EXAMPLES

The commands

```
mount -t cd9660 -o ro /dev/cd0a /usr/src
mount -t union /var/obj /usr/src
```

mount the CD-ROM drive /dev/cd0a on /usr/src and then attaches /var/obj on top. For most purposes the effect of this is to make the source tree appear writable even though it is stored on a CD-ROM.

The command

```
mount -t union -o -b /sys $HOME/sys
```

attaches the system source tree below the `sys` directory in the user's home directory. This allows individual users to make private changes to the source, and build new kernels, without those changes becoming visible to other users. Note that the files in the lower layer remain accessible via `/sys`.

SEE ALSO

`intro`(2), `mount`(2), `unmount`(2), `fstab`(5), `fsck_ffs`(8), `mount`(8), `mount_null`(8), `sysctl`(8)

HISTORY

The **`mount_union`** command first appeared in 4.4BSD.

BUGS

Without whiteout support from the filesystem backing the upper layer, there is no way that delete and rename operations on lower layer objects can be done. An attempt to mount a union directory under one which does not have whiteout support will return `EOPNOTSUPP ("Operation not supported").` Whiteout support can be added to an existing FFS filesystem by using the **`-c`** option of `fsck_ffs`(8).

Running `find`(1) over a union tree has the side-effect of creating a tree of shadow directories in the upper layer.

NAME

mountd — service remote NFS mount requests

SYNOPSIS

mountd [**-dNn**] [**-P** *policy*] [**-p** *port*] [*exportsfile*]

DESCRIPTION

mountd is the server for NFS mount requests from other client machines. **mountd** listens for service requests at the port indicated in the NFS server specification; see *Network File System Protocol Specification*, RFC 1094, Appendix A and *NFS: Network File System Version 3 Protocol Specification*, Appendix I.

Options and operands available for **mountd**:

-d Enable debugging mode. **mountd** will not detach from the controlling terminal and will print debugging messages to stderr.

-N Do not require privileged ports for mount or NFS RPC calls. This option is equivalent to specifying "-noresvport -noresvmnt" on every export. See exports(5) for more information.

-n This flag used to indicate that clients were required to make requests from reserved ports, but it is now no longer functional. It is only provided for backwards compatibility. Requests are checked for reserved ports on a per-export basis, see exports(5).

-P *policy*
IPsec *policy* string, as described in ipsec_set_policy(3). Multiple IPsec policy strings may be specified by using a semicolon as a separator. If conflicting policy strings are found in a single line, the last string will take effect. If an invalid IPsec policy string is used **mountd** logs an error message and terminates itself.

-p *port*
Force **mountd** to bind to the given port. If this option is not given, **mountd** may bind to every anonymous port (in the range 600-1023) which causes trouble when trying to use NFS through a firewall.

exportsfile
The *exportsfile* argument specifies an alternative location for the exports file.

When **mountd** is started, it loads the export host addresses and options into the kernel using the nfssvc(2) system call. After changing the exports file, a hangup signal should be sent to the **mountd** daemon to get it to reload the export information. After sending the SIGHUP (kill −s HUP ‘cat /var/run/mountd.pid‘), check the syslog output to see if **mountd** logged any parsing errors in the exports file.

After receiving SIGTERM, **mountd** sends a broadcast request to remove the mount list from all the clients. This can take a long time, since the broadcast request waits for each client to respond.

FILES

/etc/exports	the list of exported filesystems
/var/run/mountd.pid	the pid of the currently running **mountd**
/var/db/mountdtab	the list of remotely mounted filesystems

SEE ALSO

nfsstat(1), nfssvc(2), exports(5), nfsd(8), rpcbind(8), showmount(8)

HISTORY

The **mountd** utility first appeared in 4.4BSD.

NAME

moused — pass mouse data to mouse mux

SYNOPSIS

moused [**-DPRacdfs**] [**-I** *file*] [**-F** *rate*] [**-r** *resolution*] [**-S** *baudrate*]
[**-W** *devicename*] [**-a** *X*[,Y]] [**-m** *N=M*] [**-w** *N*] [**-z** *target*] [**-t** *mousetype*]
[**-3** [**-E** *timeout*]] **-p** *port*

moused [**-Pd**] **-p** *port* **-i** *info*

DESCRIPTION

The mouse daemon **moused** and the console driver work together to support access to serial mice from user programs. They virtualize the mouse and provide user programs with mouse data in the standard format (see `wsmouse`(4)).

moused listens to the specified port for mouse data, interprets and then passes it via ioctls to the console driver. It reports translation movement, button press/release events and movement of the roller or the wheel if available. The roller/wheel movement is reported as “Z” axis movement.

If **moused** receives the signal `SIGHUP`, it will reopen the mouse port and reinitializes itself. Useful if the mouse is attached/detached while the system is suspended.

The following options are available:

-3 Emulate the third (middle) button for 2-button mice. It is emulated by pressing the left and right physical buttons simultaneously.

-D Lower DTR on the serial port. This option is valid only if *mousesystems* is selected as the protocol type. The DTR line may need to be dropped for a 3-button mouse to operate in the *mousesystems* mode.

-E *timeout*
When the third button emulation is enabled (see above), **moused** waits *timeout* milliseconds at most before deciding whether two buttons are being pressed simultaneously. The default timeout is 100 milliseconds.

-F *rate*
Set the report rate (reports per second) of the device if supported.

-I *file*
Write the process id of **moused** in the specified file. Without this option, the process id will be stored in `/var/run/moused.pid`.

-P Do not start the Plug and Play COM device enumeration procedure when identifying the serial mouse. If this option is given together with the **-i** option, **moused** will not be able to print useful information for the serial mouse.

-R Lower RTS on the serial port. This option is valid only if *mousesystems* is selected as the protocol type by the **-t** option below. It is often used with the **-D** option above. Both RTS and DTR lines may need to be dropped for a 3-button mouse to operate in the *mousesystems* mode.

-S *baudrate*
Select the baudrate for the serial port (1200 to 9600). Not all serial mice support this option.

-W *devicename*
Select the `wsmux`(4) control device. The default is `/dev/wsmuxctl0`.

−a *X*[,Y]
Accelerate or decelerate the mouse input. This is a linear acceleration only. Values less than 1.0 slow down movement, values greater than 1.0 speed it up. Specifying only one value sets the acceleration for both axes.

−c Some mice report middle button down events as if the left and right buttons are being pressed. This option handles this.

−d Enable debugging messages.

−f Do not become a daemon and instead run as a foreground process. Useful for testing and debugging.

−i *info*
Print specified information and quit. Available pieces of information are:

port	Port (device file) name, e.g. `/dev/tty00`.
if	Interface type: serial, bus, inport or ps/2.
type	Protocol type. It is one of the types listed under the **−t** option below.
model	Mouse model. **moused** may not always be able to identify the model.
all	All of the above items. Print port, interface, type and model in this order in one line.

If **moused** cannot determine the requested information, it prints ‘‘unknown’’ or ‘‘generic’’.

−m *N=M*
Assign the physical button *M* to the logical button *N*. You may specify as many instances of this option as you like. More than one physical button may be assigned to a logical button at the same time. In this case the logical button will be down, if either of the assigned physical buttons is held down. Do not put space around ‘=’.

−p *port*
Use *port* to communicate with the mouse.

−r *resolution*
Set the resolution of the device; in Dots Per Inch, or *low*, *medium-low*, *medium-high* or *high*. This option may not be supported by all the device.

−s Select a baudrate of 9600 for the serial line. Not all serial mice support this option.

−t *type*
Specify the protocol type of the mouse attached to the port. You may explicitly specify a type listed below, or use *auto* to let **moused** automatically select an appropriate protocol for the given mouse. If you entirely omit this option on the command line, **−t** *auto* is assumed. Under normal circumstances, you need to use this option only if **moused** is not able to detect the protocol automatically.

Note that if a protocol type is specified with this option, the **−P** option above is implied and Plug and Play COM device enumeration procedure will be disabled.

Valid types for this option are listed below.

For the serial mouse:

microsoft	Microsoft serial mouse protocol. Most 2-button serial mice use this protocol.
intellimouse	Microsoft IntelliMouse protocol. Genius NetMouse, ASCII Mie Mouse, Logitech MouseMan+ and FirstMouse+ use this protocol too. Other mice with a roller/wheel may be compatible with this protocol.

mousesystems	MouseSystems 5-byte protocol. 3-button mice may use this protocol.
mmseries	MM Series mouse protocol.
logitech	Logitech mouse protocol. Note that this is for old Logitech models. *mouseman* or *intellimouse* should be specified for newer models.
mouseman	Logitech MouseMan and TrackMan protocol. Some 3-button mice may be compatible with this protocol. Note that MouseMan+ and FirstMouse+ use *intellimouse* protocol rather than this one.
glidepoint	ALPS GlidePoint protocol.
thinkingmouse	Kensington ThinkingMouse protocol.
mmhitab	Hitachi tablet protocol.
x10mouseremote	X10 MouseRemote.
kidspad	Genius Kidspad and Easypad protocol.
versapad	Interlink VersaPad protocol.

-w *N* Make the physical button *N* act as the wheel mode button. While this button is pressed, X and Y axis movement is reported to be zero and the Y axis movement is mapped to Z axis. You may further map the Z axis movement to virtual buttons by the **-z** option below.

-z *target*
Map Z axis (roller/wheel) movement to another axis or to virtual buttons. Valid *target* maybe:

x
y X or Y axis movement will be reported when the Z axis movement is detected.

N Report down events for the virtual buttons *N* and *N+1* respectively when negative and positive Z axis movement is detected. There do not need to be physical buttons *N* and *N+1*. Note that mapping to logical buttons is carried out after mapping from the Z axis movement to the virtual buttons is done.

N1 N2
Report down events for the virtual buttons *N1* and *N2* respectively when negative and positive Z axis movement is detected.

N1 N2 N3 N4
This is useful for the mouse with two wheels of which the second wheel is used to generate horizontal scroll action, and for the mouse which has a knob or a stick which can detect the horizontal force applied by the user.

The motion of the second wheel will be mapped to the buttons *N3*, for the negative direction, and *N4*, for the positive direction. If the buttons *N3* and *N4* actually exist in this mouse, their actions will not be detected.

Note that horizontal movement or second roller/wheel movement may not always be detected, because there appears to be no accepted standard as to how it is encoded.

Note also that some mice think left is the negative horizontal direction, others may think otherwise. Moreover, there are some mice whose two wheels are both mounted vertically, and the direction of the second vertical wheel does not match the first one's.

Multiple Mice

As many instances of **moused** as the number of mice attached to the system may be run simultaneously; one instance for each serial mouse.

FILES

`/dev/wsmuxctl0`	default device to control mouse mux
`/var/run/moused.pid`	process id of the currently running **moused**

EXAMPLES

```
moused -p /dev/tty00 -i type
```

Let **moused** determine the protocol type of the mouse at the serial port `/dev/tty00`. If successful, **moused** will print the type, otherwise it will say ''unknown''.

```
moused -p /dev/tty00
```

If **moused** is able to identify the protocol type of the mouse at the specified port automatically, you can start the daemon without the **−t** option and enable the mouse pointer in the text console as above.

```
moused -p /dev/tty01 -t microsoft
```

Start **moused** on the serial port `/dev/tty01`. The protocol type *microsoft* is explicitly specified by the **−t** option.

```
moused -p /dev/tty01 -m 1=3 -m 3=1
```

Assign the physical button 3 (right button) to the logical button 1 (logical left) and the physical button 1 (left) to the logical button 3 (logical right). This will effectively swap the left and right buttons.

```
moused -p /dev/tty01 -t intellimouse -z 4
```

Report negative Z axis (roller) movement as the button 4 pressed and positive Z axis movement as the button 5 pressed.

The mouse daemon is normally enabled by setting `moused=YES` in `/etc/rc.conf`.

SEE ALSO

`wsmouse`(4), `wsmux`(4), `rc.conf`(5), `wsmoused`(8)

STANDARDS

moused partially supports "Plug and Play External COM Device Specification" in order to support PnP serial mice. However, due to various degrees of conformance to the specification by existing serial mice, it does not strictly follow version 1.0 of the standard. Even with this less strict approach, it may not always determine an appropriate protocol type for the given serial mouse.

HISTORY

The mouse daemon **moused** first appeared in FreeBSD 2.2 and NetBSD 1.6.

AUTHORS

moused was written by Michael Smith ⟨msmith@FreeBSD.org⟩. This manual page was written by
Mike Pritchard ⟨mpp@FreeBSD.org⟩. The daemon and manual page have since been updated by
Kazutaka Yokota ⟨yokota@FreeBSD.org⟩. The NetBSD port was done by
Lennart Augustsson ⟨augustss@NetBSD.org⟩.

BUGS

Many pad devices behave as if the first (left) button were pressed if the user 'taps' the surface of the pad. In contrast, some ALPS GlidePoint and Interlink VersaPad models treat the tapping action as fourth button events. Use the option ''-m 1=4'' for these models to obtain the same effect as the other pad devices.

NAME

mrinfo − Displays configuration info from a multicast router

SYNOPSIS

/usr/sbin/mrinfo [**−d** *debug_level*] [**−r** *retry_count*] [**−t** *timeout_count*] **multicast_router**

DESCRIPTION

mrinfo attempts to display the configuration information from the multicast router *multicast_router.*

mrinfo uses the ASK_NEIGHBORS IGMP message to the specified multicast router. If this multicast router responds, the version number and a list of their neighboring multicast router addresses is part of that response. If the responding router has a recent multicast version number, then *mrinfo* requests additional information such as metrics, thresholds, and flags from the multicast router. Once the specified multicast router responds, the configuration is displayed to the standard output.

INVOCATION

"−d" option sets the debug level. When the debug level is greater than the default value of 0, addition debugging messages are printed. Regardless of the debug level, an error condition, will always write an error message and will cause *mrinfo* to terminate. Non-zero debug levels have the following effects:

level 1	packet warnings are printed to stderr.
level 2	all level 1 messages plus notifications down networks are printed to stderr.
level 3	all level 2 messages plus notifications of all packet timeouts are printed to stderr.

"−r retry_count" sets the neighbor query retry limit. Default is 3 retry.

"−t timeout_count" sets the number of seconds to wait for a neighbor query reply. Default timeout is 4 seconds.

SAMPLE OUTPUT

mrinfo mbone.phony.dom.net

```
127.148.176.10 (mbone.phony.dom.net) [version 3.3]:
 127.148.176.10 -> 0.0.0.0 (?) [1/1/querier]
 127.148.176.10 -> 127.0.8.4 (mbone2.phony.dom.net) [1/45/tunnel]
 127.148.176.10 -> 105.1.41.9 (momoney.com) [1/32/tunnel/down]
 127.148.176.10 -> 143.192.152.119 (mbone.dipu.edu) [1/32/tunnel]
```

For each neighbor of the queried multicast router, the IP of the queried router is displayed, followed by the IP and name of the neighbor. In square brackets the metric (cost of connection), the threshold (multicast ttl) is displayed. If the queried multicast router has a newer version number, the type (tunnel, srcrt) and status (disabled, down) of the connection is displayed.

IMPORTANT NOTE

mrinfo must be run as root.

SEE ALSO

mrouted(8)**, map-mbone**(8)**, mtrace**(8)

AUTHOR

Van Jacobson

NAME

mrouted — IP multicast routing daemon

SYNOPSIS

mrouted [**-c** *config_file*] [**-d** *debug_level*] [**-p**]

DESCRIPTION

mrouted is an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP), an earlier version of which is specified in RFC 1075. It maintains topological knowledge via a distance-vector routing protocol (like RIP, described in RFC 1058), upon which it implements a multicast datagram forwarding algorithm called Reverse Path Multicasting.

mrouted forwards a multicast datagram along a shortest (reverse) path tree rooted at the subnet on which the datagram originates. The multicast delivery tree may be thought of as a broadcast delivery tree that has been pruned back so that it does not extend beyond those subnetworks that have members of the destination group. Hence, datagrams are not forwarded along those branches which have no listeners of the multicast group. The IP time-to-live of a multicast datagram can be used to limit the range of multicast datagrams.

In order to support multicasting among subnets that are separated by (unicast) routers that do not support IP multicasting, **mrouted** includes support for "tunnels", which are virtual point-to-point links between pairs of **mrouted** daemons located anywhere in an internet. IP multicast packets are encapsulated for transmission through tunnels, so that they look like normal unicast datagrams to intervening routers and subnets. The encapsulation is added on entry to a tunnel, and stripped off on exit from a tunnel. By default, the packets are encapsulated using the IP-in-IP protocol (IP protocol number 4). Older versions of **mrouted** tunnel using IP source routing, which puts a heavy load on some types of routers. This version does not support IP source route tunneling.

The tunneling mechanism allows **mrouted** to establish a virtual internet, for the purpose of multicasting only, which is independent of the physical internet, and which may span multiple Autonomous Systems. This capability is intended for experimental support of internet multicasting only, pending widespread support for multicast routing by the regular (unicast) routers. **mrouted** suffers from the well-known scaling problems of any distance-vector routing protocol, and does not (yet) support hierarchical multicast routing.

mrouted handles multicast routing only; there may or may not be unicast routing software running on the same machine as **mrouted**. With the use of tunnels, it is not necessary for **mrouted** to have access to more than one physical subnet in order to perform multicast forwarding.

INVOCATION

If no **-d** option is given, or if the debug level is specified as 0, **mrouted** detaches from the invoking terminal. Otherwise, it remains attached to the invoking terminal and responsive to signals from that terminal. If **-d** is given with no argument, the debug level defaults to 2. Regardless of the debug level, **mrouted** always writes warning and error messages to the system log daemon. Non-zero debug levels have the following effects:

1	all syslog'ed messages are also printed to stderr.
2	all level 1 messages plus notifications of "significant" events are printed to stderr.
3	all level 2 messages plus notifications of all packet arrivals and departures are printed to stderr.

Upon startup, mrouted writes its pid to the file `/var/run/mrouted.pid`.

CONFIGURATION

mrouted automatically configures itself to forward on all multicast-capable interfaces, i.e., interfaces that have the IFF_MULTICAST flag set (excluding the loopback "interface"), and it finds other **mrouted** directly reachable via those interfaces. To override the default configuration, or to add tunnel links to other

mrouted configuration commands may be placed in `/etc/mrouted.conf` (or an alternative file, specified by the **-c** option). There are four types of configuration commands:

```
phyint <local-addr> [disable] [metric <m>]
        [threshold <t>] [rate_limit <b>]
        [boundary (<boundary-name>|<scoped-addr>/<mask-len>)]
        [altnet <network>/<mask-len>]

        tunnel <local-addr> <remote-addr> [metric <m>]
        [threshold <t>] [rate_limit <b>]
        [boundary (<boundary-name>|<scoped-addr>/<mask-len>)]

cache_lifetime <ct>

pruning <off/on>

name <boundary-name> <scoped-addr>/<mask-len>
```

The file format is free-form; whitespace (including newlines) is not significant. The *boundary* and *altnet* options may be specified as many times as necessary.

The phyint command can be used to disable multicast routing on the physical interface identified by local IP address *<local-addr>*, or to associate a non-default metric or threshold with the specified physical interface. The local IP address *<local-addr>* may be replaced by the interface name (e.g., le0). If a phyint is attached to multiple IP subnets, describe each additional subnet with the altnet keyword. Phyint commands must precede tunnel commands.

The tunnel command can be used to establish a tunnel link between local IP address *<local-addr>* and remote IP address *<remote-addr>*, and to associate a non-default metric or threshold with that tunnel. The local IP address *<local-addr>* may be replaced by the interface name (e.g., le0). The remote IP address *<remote-addr>* may be replaced by a host name, if and only if the host name has a single IP address associated with it. The tunnel must be set up in the mrouted.conf files of both routers before it can be used.

The cache_lifetime is a value that determines the amount of time that a cached multicast route stays in kernel before timing out. The value of this entry should lie between 300 (5 min) and 86400 (1 day). It defaults to 300.

The *pruning* option is provided for **mrouted** to act as a non-pruning router. It is also possible to start **mrouted** in a non-pruning mode using the **-p** option on the command line. It is expected that a router would be configured in this manner for test purposes only. The default mode is pruning enabled.

You may assign names to boundaries to make configuration easier with the name keyword. The boundary option on phyint or tunnel commands can accept either a name or a boundary.

The metric is the "cost" associated with sending a datagram on the given interface or tunnel; it may be used to influence the choice of routes. The metric defaults to 1. Metrics should be kept as small as possible, because **mrouted** cannot route along paths with a sum of metrics greater than 31.

The threshold is the minimum IP time-to-live required for a multicast datagram to be forwarded to the given interface or tunnel. It is used to control the scope of multicast datagrams. (The TTL of forwarded packets is only compared to the threshold, it is not decremented by the threshold. Every multicast router decrements the TTL by 1.) The default threshold is 1.

In general, all **mrouted** connected to a particular subnet or tunnel should use the same metric and threshold for that subnet or tunnel.

The rate_limit option allows the network administrator to specify a certain bandwidth in Kbits/second which would be allocated to multicast traffic. It defaults to 500Kbps on tunnels, and 0 (unlimited) on physical interfaces.

The boundary option allows an interface to be configured as an administrative boundary for the specified scoped address. Packets belonging to this address will not be forwarded on a scoped interface. The boundary option accepts either a name or a boundary spec.

mrouted will not initiate execution if it has fewer than two enabled vifs, where a vif (virtual interface) is either a physical multicast-capable interface or a tunnel. It will log a warning if all of its vifs are tunnels; such an **mrouted** configuration would be better replaced by more direct tunnels (i.e., eliminate the middle man).

EXAMPLE CONFIGURATION

This is an example configuration for a mythical multicast router at a big school.

```
#
# mrouted.conf example
#
# Name our boundaries to make it easier.
name LOCAL 239.255.0.0/16
name EE 239.254.0.0/16
#
# le1 is our gateway to compsci, don't forward our
# local groups to them.
phyint le1 boundary EE
#
# le2 is our interface on the classroom net, it has four
# different length subnets on it.
# Note that you can use either an ip address or an
# interface name
phyint 172.16.12.38 boundary EE altnet 172.16.15.0/26
        altnet 172.16.15.128/26 altnet 172.16.48.0/24
#
# atm0 is our ATM interface, which doesn't properly
# support multicasting.
phyint atm0 disable
#
# This is an internal tunnel to another EE subnet.
# Remove the default tunnel rate limit, since this
# tunnel is over ethernets.
tunnel 192.168.5.4 192.168.55.101 metric 1 threshold 1
        rate_limit 0
#
# This is our tunnel to the outside world.
# Careful with those boundaries, Eugene.
tunnel 192.168.5.4 10.11.12.13 metric 1 threshold 32
        boundary LOCAL boundary EE
```

SIGNALS

mrouted responds to the following signals:

HUP	restarts **mrouted**. The configuration file is reread every time this signal is evoked.
INT	terminates execution gracefully (i.e., by sending good-bye messages to all neighboring routers).

TERM
: same as INT

USR1
: dumps the internal routing tables to `/var/tmp/mrouted.dump`.

USR2
: dumps the internal cache tables to `/var/tmp/mrouted.cache`.

QUIT
: dumps the internal routing tables to stderr (only if **mrouted** was invoked with a non-zero debug level).

For convenience in sending signals, **mrouted** writes its pid to `/var/run/mrouted.pid` upon startup.

FILES

```
/etc/mrouted.conf
/var/run/mrouted.pid
/var/tmp/mrouted.dump
/var/tmp/mrouted.cache
```

EXAMPLES

The routing tables look like this:

```
Virtual Interface Table
 Vif  Local-Address                    Metric  Thresh  Flags
  0   36.2.0.8      subnet: 36.2          1       1    querier
                    groups: 224.0.2.1
                            224.0.0.4
                   pkts in: 3456
                  pkts out: 2322323

  1   36.11.0.1     subnet: 36.11         1       1    querier
                    groups: 224.0.2.1
                            224.0.1.0
                            224.0.0.4
                   pkts in: 345
                  pkts out: 3456

  2   36.2.0.8      tunnel: 36.8.0.77     3       1
                     peers: 36.8.0.77 (2.2)
                boundaries: 239.0.1
                          : 239.1.2
                   pkts in: 34545433
                  pkts out: 234342

  3   36.2.0.8     tunnel: 36.6.8.23    3       16

Multicast Routing Table (1136 entries)
 Origin-Subnet   From-Gateway    Metric Tmr In-Vif  Out-Vifs
 36.2                                 1   45    0    1* 2  3*
 36.8            36.8.0.77            4   15    2    0* 1* 3*
 36.11                                1   20    1    0* 2  3*
 .
 .
 .
```

In this example, there are four vifs connecting to two subnets and two tunnels. The vif 3 tunnel is not in use (no peer address). The vif 0 and vif 1 subnets have some groups present; tunnels never have any groups. This instance of **mrouted** is the one responsible for sending periodic group membership queries on the vif 0 and vif 1 subnets, as indicated by the "querier" flags. The list of boundaries indicate the scoped addresses on that interface. A count of the number of incoming and outgoing packets is also shown at each interface.

Associated with each subnet from which a multicast datagram can originate is the address of the previous hop router (unless the subnet is directly- connected), the metric of the path back to the origin, the amount of time since we last received an update for this subnet, the incoming vif for multicasts from that origin, and a list of outgoing vifs. "∗" means that the outgoing vif is connected to a leaf of the broadcast tree rooted at the origin, and a multicast datagram from that origin will be forwarded on that outgoing vif only if there are members of the destination group on that leaf.

mrouted also maintains a copy of the kernel forwarding cache table. Entries are created and deleted by **mrouted**.

The cache tables look like this:

```
Multicast Routing Cache Table (147 entries)
 Origin           Mcast-group     CTmr  Age Ptmr IVif Forwvifs
 13.2.116/22       224.2.127.255     3m   2m    -  0    1
>13.2.116.19
>13.2.116.196
 138.96.48/21      224.2.127.255     5m   2m    -  0    1
>138.96.48.108
 128.9.160/20      224.2.127.255     3m   2m    -  0    1
>128.9.160.45
 198.106.194/24    224.2.135.190     9m  28s   9m  0P
>198.106.194.22
```

Each entry is characterized by the origin subnet number and mask and the destination multicast group. The 'CTmr' field indicates the lifetime of the entry. The entry is deleted from the cache table when the timer decrements to zero. The 'Age' field is the time since this cache entry was originally created. Since cache entries get refreshed if traffic is flowing, routing entries can grow very old. The 'Ptmr' field is simply a dash if no prune was sent upstream, or the amount of time until the upstream prune will time out. The 'Ivif' field indicates the incoming vif for multicast packets from that origin. Each router also maintains a record of the number of prunes received from neighboring routers for a particular source and group. If there are no members of a multicast group on any downward link of the multicast tree for a subnet, a prune message is sent to the upstream router. They are indicated by a "P" after the vif number. The Forwvifs field shows the interfaces along which datagrams belonging to the source-group are forwarded. A "p" indicates that no datagrams are being forwarded along that interface. An unlisted interface is a leaf subnet with are no members of the particular group on that subnet. A "b" on an interface indicates that it is a boundary interface, i.e., traffic will not be forwarded on the scoped address on that interface. An additional line with a ">" as the first character is printed for each source on the subnet. Note that there can be many sources in one subnet.

SEE ALSO

`map-mbone`(8), `mrinfo`(8), `mtrace`(8)

DVMRP is described, along with other multicast routing algorithms, in the paper "Multicast Routing in Internetworks and Extended LANs" by S. Deering, in the Proceedings of the ACM SIGCOMM '88 Conference.

AUTHORS

Steve Deering, Ajit Thyagarajan, Bill Fenner

NAME

mscdlabel — generate disk label from CD track information

SYNOPSIS

mscdlabel [*device* | *file*]

DESCRIPTION

mscdlabel is used to generate a NetBSD disk label from track information read from the CD. This way, data of previous sessions of a multi-session CD can be accessed.

mscdlabel scans the CD's TOC, beginning with the last track. For each data track where an ISO9660 filesystem is identified, basic information (volume label, creation date) is printed and a partition entry added to the in-core disklabel.

The raw partition (typically partition *c*, but *d* on i386 and some other platforms) is left alone during this process.

The **mscdlabel** utility can also be used on files or non-CD devices. In this case a single track is assumed. If the device supports disk labels, a label will be written as described above. Otherwise, just the ISO volume label information will be printed.

SEE ALSO

cdplay(1), disklabel(8), mbrlabel(8)

NAME

mtrace − print multicast path from a source to a receiver

SYNOPSIS

mtrace [**−g** *gateway*] [**−i** *if_addr*] [**−l**] [**−M**] [**−m** *max_hops*] [**−n**] [**−p**] [**−q** *nqueries*] [**−r** *resp_dest*] [**−s**] [**−S** *stat_int*] [**−t** *ttl*] [**−v**] [**−w** *waittime*] *source* [*receiver*] [*group*]

DESCRIPTION

Assessing problems in the distribution of IP multicast traffic can be difficult. **mtrace** uses a tracing feature implemented in multicast routers (**mrouted** version 3.3 and later) that is accessed via an extension to the IGMP protocol. A trace query is passed hop-by-hop along the reverse path from the *receiver* to the *source*, collecting hop addresses, packet counts, and routing error conditions along the path, and then the response is returned to the requestor.

The only required parameter is the *source* host name or address. The default *receiver* is the host running mtrace, and the default *group* is "MBone Audio" (224.2.0.1), which is sufficient if packet loss statistics for a particular multicast group are not needed. These two optional parameters may be specified to test the path to some other receiver in a particular group, subject to some constraints as detailed below. The two parameters can be distinguished because the *receiver* is a unicast address and the *group* is a multicast address.

NOTE: For Solaris 2.4/2.5, if the multicast interface is not the default interface, the -i option must be used to set the local address.

OPTIONS

−g *gwy* Send the trace query via unicast directly to the multicast router *gwy* rather than multicasting the query. This must be the last-hop router on the path from the intended *source* to the *receiver*.

CAUTION!! Versions 3.3 and 3.5 of **mrouted** will crash if a trace query is received via a unicast packet and **mrouted** has no route for the *source* address. Therefore, do not use the **−g** option unless the target **mrouted** has been verified to be 3.4 or newer than 3.5.

−i *addr* Use *addr* as the local interface address (on a multi-homed host) for sending the trace query and as the default for the *receiver* and the response destination.

−l Loop indefinitely printing packet rate and loss statistics for the multicast path every 10 seconds (see **−S** *stat_int*).

−M Always send the response using multicast rather than attempting unicast first.

−m *n* Set to *n* the maximum number of hops that will be traced from the *receiver* back toward the *source*. The default is 32 hops (infinity for the DVMRP routing protocol).

−n Print hop addresses numerically rather than symbolically and numerically (saves a nameserver address-to-name lookup for each router found on the path).

−q *n* Set the maximum number of query attempts for any hop to *n*. The default is 3.

−p Listen passively for multicast responses from traces initiated by others. This works best when run on a multicast router.

−r *host* Send the trace response to *host* rather than to the host on which **mtrace** is being run, or to a multicast address other than the one registered for this purpose (224.0.1.32).

−s Print a short form output including only the multicast path and not the packet rate and loss statistics.

−S *n* Change the interval between statistics gathering traces to *n* seconds (default 10 seconds).

−t *ttl* Set the *ttl* (time-to-live, or number of hops) for multicast trace queries and responses. The default is 64, except for local queries to the "all routers" multicast group which use ttl 1.

−v Verbose mode; show hop times on the initial trace and statistics display.

−w *n* Set the time to wait for a trace response to *n* seconds (default 3 seconds).

USAGE

How It Works

The technique used by the **traceroute** tool to trace unicast network paths will not work for IP multicast because ICMP responses are specifically forbidden for multicast traffic. Instead, a tracing feature has been built into the multicast routers. This technique has the advantage that additional information about packet rates and losses can be accumulated while the number of packets sent is minimized.

Since multicast uses reverse path forwarding, the trace is run backwards from the *receiver* to the *source*. A trace query packet is sent to the last hop multicast router (the leaf router for the desired *receiver* address). The last hop router builds a trace response packet, fills in a report for its hop, and forwards the trace packet using unicast to the router it believes is the previous hop for packets originating from the specified *source*. Each router along the path adds its report and forwards the packet. When the trace response packet reaches the first hop router (the router that is directly connected to the source's net), that router sends the completed response to the response destination address specified in the trace query.

If some multicast router along the path does not implement the multicast traceroute feature or if there is some outage, then no response will be returned. To solve this problem, the trace query includes a maximum hop count field to limit the number of hops traced before the response is returned. That allows a partial path to be traced.

The reports inserted by each router contain not only the address of the hop, but also the ttl required to forward and some flags to indicate routing errors, plus counts of the total number of packets on the incoming and outgoing interfaces and those forwarded for the specified *group*. Taking differences in these counts for two traces separated in time and comparing the output packet counts from one hop with the input packet counts of the next hop allows the calculation of packet rate and packet loss statistics for each hop to isolate congestion problems.

Finding the Last-Hop Router

The trace query must be sent to the multicast router which is the last hop on the path from the *source* to the *receiver*. If the receiver is on the local subnet (as determined using the subnet mask), then the default method is to multicast the trace query to all-routers.mcast.net (224.0.0.2) with a ttl of 1. Otherwise, the trace query is multicast to the *group* address since the last hop router will be a member of that group if the receiver is. Therefore it is necessary to specify a group that the intended receiver has joined. This multicast is sent with a default ttl of 64, which may not be sufficient for all cases (changed with the **−t** option). If the last hop router is known, it may also be addressed directly using the **−g** option). Alternatively, if it is desired to trace a group that the receiver has not joined, but it is known that the last-hop router is a member of another group, the **−g** option may also be used to specify a different multicast address for the trace query.

When tracing from a multihomed host or router, the default receiver address may not be the desired interface for the path from the source. In that case, the desired interface should be specified explicitly as the *receiver*.

Directing the Response

By default, **mtrace** first attempts to trace the full reverse path, unless the number of hops to trace is explicitly set with the **−m** option. If there is no response within a 3 second timeout interval (changed with the **−w** option), a "*" is printed and the probing switches to hop-by-hop mode. Trace queries are issued starting with a maximum hop count of one and increasing by one until the full path is traced or no response is received. At each hop, multiple probes are sent (default is three, changed with **−q** option). The first half of the attempts (default is one) are made with the unicast address of the host running **mtrace** as the destination for the response. Since the unicast route may be blocked, the remainder of attempts request that the response be multicast to mtrace.mcast.net (224.0.1.32) with the ttl set to 32 more than what's needed to pass the thresholds seen so far along the path to the receiver. For the last quarter of the attempts (default is one), the ttl is increased by another 32 each time up to a maximum of 192. Alternatively, the ttl may be set explicitly with the **−t** option and/or the initial unicast attempts can be forced to use multicast instead with the **−M** option. For each attempt, if no response is received within the timeout, a "*" is printed. After the specified number of attempts have failed, **mtrace** will try to query the next hop router with a DVMRP_ASK_NEIGHBORS2 request (as used by the **mrinfo** program) to see what kind of router it is.

EXAMPLES

The output of **mtrace** is in two sections. The first section is a short listing of the hops in the order they are queried, that is, in the reverse of the order from the *source* to the *receiver*. For each hop, a line is printed showing the hop number (counted negatively to indicate that this is the reverse path); the multicast routing protocol (DVMRP, MOSPF, PIM, etc.); the threshold required to forward data (to the previous hop in the listing as indicated by the up-arrow character); and the cumulative delay for the query to reach that hop (valid only if the clocks are synchronized). This first section ends with a line showing the round-trip time which measures the interval from when the query is issued until the response is received, both derived from the local system clock. A sample use and output might be:

```
oak.isi.edu 80# mtrace -l caraway.lcs.mit.edu 224.2.0.3
Mtrace from 18.26.0.170 to 128.9.160.100 via group 224.2.0.3
Querying full reverse path...
  0  oak.isi.edu (128.9.160.100)
 -1  cub.isi.edu (128.9.160.153)  DVMRP  thresh^ 1  3 ms
 -2  la.dart.net (140.173.128.1)  DVMRP  thresh^ 1  14 ms
 -3  dc.dart.net (140.173.64.1)  DVMRP  thresh^ 1  50 ms
 -4  bbn.dart.net (140.173.32.1)  DVMRP  thresh^ 1  63 ms
 -5  mit.dart.net (140.173.48.2)  DVMRP  thresh^ 1  71 ms
 -6  caraway.lcs.mit.edu (18.26.0.170)
Round trip time 124 ms
```

The second section provides a pictorial view of the path in the forward direction with data flow indicated by arrows pointing downward and the query path indicated by arrows pointing upward. For each hop, both the entry and exit addresses of the router are shown if different, along with the initial ttl required on the packet in order to be forwarded at this hop and the propagation delay across the hop assuming that the routers at both ends have synchronized clocks. The right half of this section is composed of several columns of statistics in two groups. Within each group, the columns are the number of packets lost, the number of packets sent, the percentage lost, and the average packet rate at each hop. These statistics are calculated from differences between traces and from hop to hop as explained above. The first group shows the statistics for all traffic flowing out the interface at one hop and in the interface at the next hop. The second group shows the statistics only for traffic forwarded from the specified *source* to the specified *group*.

These statistics are shown on one or two lines for each hop. Without any options, this second section of the output is printed only once, approximately 10 seconds after the initial trace. One line is shown for each hop showing the statistics over that 10-second period. If the **−l** option is given, the second section is repeated every 10 seconds and two lines are shown for each hop. The first line shows the statistics for the last 10 seconds, and the second line shows the cumulative statistics over the period since the initial trace, which is 101 seconds in the example below. The second section of the output is omitted if the **−s** option is set.

```
Waiting to accumulate statistics... Results after 101 seconds:

  Source       Response Dest  Packet Statistics For  Only For Traffic
18.26.0.170    128.9.160.100  All Multicast Traffic  From 18.26.0.170
     |       __/ rtt  125 ms  Lost/Sent = Pct  Rate    To 224.2.0.3
     v      /    hop   65 ms  ---------------------  ------------------
18.26.0.144
140.173.48.2   mit.dart.net
     |     ^     ttl    1      0/6    = --%   0 pps   0/2  = --%  0 pps
     v     |     hop    8 ms   1/52   =  2%   0 pps   0/18 =  0%  0 pps
140.173.48.1
140.173.32.1   bbn.dart.net
     |     ^     ttl    2      0/6    = --%   0 pps   0/2  = --%  0 pps
     v     |     hop   12 ms   1/52   =  2%   0 pps   0/18 =  0%  0 pps
140.173.32.2
140.173.64.1   dc.dart.net
     |     ^     ttl    3      0/271  =  0%  27 pps   0/2  = --%  0 pps
     v     |      hop   34 ms  -1/2652 =  0%  26 pps   0/18 =  0%  0 pps
140.173.64.2
140.173.128.1  la.dart.net
     |     ^      ttl    4     -2/831  =  0%  83 pps   0/2  = --%  0 pps
     v     |      hop   11 ms  -3/8072 =  0%  79 pps   0/18 =  0%  0 pps
140.173.128.2
128.9.160.153  cub.isi.edu
     |      \__   ttl    5        833         83 pps     2         0 pps
     v         \ hop   -8 ms     8075         79 pps     18        0 pps
128.9.160.100  128.9.160.100
  Receiver     Query Source
```

Because the packet counts may be changing as the trace query is propagating, there may be small errors (off by 1 or 2) in these statistics. However, those errors should not accumulate, so the cumulative statistics line should increase in accuracy as a new trace is run every 10 seconds. There are two sources of larger errors, both of which show up as negative losses:

- If the input to a node is from a multi-access network with more than one other node attached, then the input count will be (close to) the sum of the output counts from all the attached nodes, but the output count from the previous hop on the traced path will be only part of that. Hence the output count minus the input count will be negative.
- In release 3.3 of the DVMRP multicast forwarding software for SunOS and other systems, a multicast packet generated on a router will be counted as having come in an interface even though it did not. This creates the negative loss that can be seen in the example above.

Note that these negative losses may mask positive losses.

In the example, there is also one negative hop time. This simply indicates a lack of synchronization between the system clocks across that hop. This example also illustrates how the percentage loss is shown as two dashes when the number of packets sent is less than 10 because the percentage would not be statistically valid.

A second example shows a trace to a receiver that is not local; the query is sent to the last-hop router with the **−g** option. In this example, the trace of the full reverse path resulted in no response because there was a node running an old version of **mrouted** that did not implement the multicast traceroute function, so **mtrace** switched to hop-by-hop mode. The “Route pruned” error code indicates that traffic for group 224.2.143.24 would not be forwarded.

```
oak.isi.edu 108# mtrace -g 140.173.48.2 204.62.246.73 \
                           butter.lcs.mit.edu 224.2.143.24
Mtrace from 204.62.246.73 to 18.26.0.151 via group 224.2.143.24
Querying full reverse path... * switching to hop-by-hop:
  0  butter.lcs.mit.edu (18.26.0.151)
 -1  jam.lcs.mit.edu (18.26.0.144)  DVMRP  thresh^ 1  33 ms  Route pruned
 -2  bbn.dart.net (140.173.48.1)  DVMRP  thresh^ 1  36 ms
 -3  dc.dart.net (140.173.32.2)  DVMRP  thresh^ 1  44 ms
 -4  darpa.dart.net (140.173.240.2)  DVMRP  thresh^ 16  47 ms
 -5  * * * noc.hpc.org (192.187.8.2) [mrouted 2.2] didn’t respond
Round trip time 95 ms
```

AUTHOR

Implemented by Steve Casner based on an initial prototype written by Ajit Thyagarajan. The multicast traceroute mechanism was designed by Van Jacobson with help from Steve Casner, Steve Deering, Dino Farinacci, and Deb Agrawal; it was implemented in **mrouted** by Ajit Thyagarajan and Bill Fenner. The option syntax and the output format of **mtrace** are modeled after the unicast **traceroute** program written by Van Jacobson.

SEE ALSO

mrouted(8)**, mrinfo**(8)**, map-mbone**(8)**, traceroute**(8)

NAME

mtree — map a directory hierarchy

SYNOPSIS

mtree [**-cCdDelLMPruUWx**] [**-i** | **-m**] [**-f** *spec*] [**-p** *path*] [**-k** *keywords*] [**-K** *keywords*] [**-R** *keywords*] [**-E** *tags*] [**-I** *tags*] [**-N** *dbdir*] [**-s** *seed*] [**-X** *exclude-file*]

DESCRIPTION

The **mtree** utility compares the file hierarchy rooted in the current directory against a specification read from the standard input. Messages are written to the standard output for any files whose characteristics do not match the specification, or which are missing from either the file hierarchy or the specification.

The options are as follows:

-c Print a specification for the file hierarchy to the standard output.

-d Ignore everything except directory type files.

-C Print ('dump') the specification as provided by **-f** *spec* in a format that's easier to parse with various tools. The full path name is always printed as the first field, and **-k**, **-K**, and **-R** can be used to control which other keywords are printed, and **-E** and **-I** can be used to control which files are printed.

-D As per **-C**, except that the path name is always printed as the last field instead of the first.

-E *tags*
Add the comma separated tags to the "exclusion" list. Non-directories with tags which are in the exclusion list are not printed with **-D**.

-e Don't complain about files that are in the file hierarchy, but not in the specification.

-f *spec*
Read the specification from *file*, instead of from the standard input.

-I *tags*
Add the comma separated tags to the "inclusion" list. Non-directories with tags which are in the inclusion list are printed with **-D**. If no inclusion list is provided, the default is to display all files.

-i If specified, set the schg and/or sappnd flags.

-K *keywords*
Add the specified (whitespace or comma separated) keywords to the current set of keywords. If `all` is specified, add all of the other keywords.

-k *keywords*
Use the **type** keyword plus the specified (whitespace or comma separated) keywords instead of the current set of keywords. If `all` is specified, use all of the other keywords. If the **type** keyword is not desired, suppress it with **-R** *type*.

-l Do "loose" permissions checks, in which more stringent permissions will match less stringent ones. For example, a file marked mode 0444 will pass a check for mode 0644. "Loose" checks apply only to read, write and execute permissions -- in particular, if other bits like the sticky bit or suid/sgid bits are set either in the specification or the file, exact checking will be performed. This flag may not be set at the same time as the **-u** or **-U** flags.

-L Follow all symbolic links in the file hierarchy.

-m If the schg and/or sappnd flags are specified, reset these flags. Note that this is only possible with securelevel less than 1 (i.e. in single user mode or while the system is running in insecure mode). See `init`(8) for information on security levels.

-M Permit merging of specification entries with different types, with the last entry take precedence.

-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.

-p *path*
Use the file hierarchy rooted in *path*, instead of the current directory.

-P Don't follow symbolic links in the file hierarchy, instead consider the symbolic link itself in any comparisons. This is the default.

-r Remove any files in the file hierarchy that are not described in the specification.

-R *keywords*
Remove the specified (whitespace or comma separated) keywords from the current set of keywords. If `all` is specified, remove all of the other keywords.

-s *seed*
Display a single checksum to the standard error output that represents all of the files for which the keyword **cksum** was specified. The checksum is seeded with the specified value.

-u Modify the owner, group, permissions, and flags of existing files, the device type of devices, and symbolic link targets, to match the specification. Create any missing directories, devices or symbolic links. User, group, and permissions must all be specified for missing directories to be created. Note that unless the **-i** option is given, the schg and sappnd flags will not be set, even if specified. If **-m** is given, these flags will be reset. Exit with a status of 0 on success, 2 if the file hierarchy did not match the specification, and 1 if any other error occurred.

-U Same as **-u** except that a mismatch is not considered to be an error if it was corrected.

-W Don't attempt to set various file attributes such as the ownership, mode, flags, or time when creating new directories or changing existing entries. This option will be most useful when used in conjunction with **-u** or **-U**.

-x Don't descend below mount points in the file hierarchy.

-X *exclude-file*
The specified file contains `fnmatch`(3) patterns matching files to be excluded from the specification, one to a line. If the pattern contains a '`/`' character, it will be matched against entire pathnames (relative to the starting directory); otherwise, it will be matched against basenames only. Comments are permitted in the *exclude-list* file.

Specifications are mostly composed of "keywords", i.e. strings that that specify values relating to files. No keywords have default values, and if a keyword has no value set, no checks based on it are performed.

Currently supported keywords are as follows:

cksum The checksum of the file using the default algorithm specified by the `cksum`(1) utility.

device The device number to use for **block** or **char** file types. The argument must be one of the following forms:

format,*major*,*minor*
A device with *major* and *minor* fields, for an operating system specified with *format*. See below for valid formats.

format,*major*,*unit*,*subunit*
A device with *major*, *unit*, and *subunit* fields, for an operating system specified with *format*. (Currently this is only supported by the **bsdos** format.)

number
Opaque number (as stored on the file system).

The following values for *format* are recognized: **native**, **386bsd**, **4bsd**, **bsdos**, **freebsd**, **hpux**, **isc**, **linux**, **netbsd**, **osf1**, **sco**, **solaris**, **sunos**, **svr3**, **svr4**, and **ultrix**.

See mknod(8) for more details.

flags The file flags as a symbolic name. See chflags(1) for information on these names. If no flags are to be set the string none may be used to override the current default. Note that the schg and sappnd flags are treated specially (see the **-i** and **-m** options).

ignore Ignore any file hierarchy below this file.

gid The file group as a numeric value.

gname The file group as a symbolic name.

link The file the symbolic link is expected to reference.

md5 The MD5 cryptographic message digest of the file.

md5digest
Synonym for **md5**.

mode The current file's permissions as a numeric (octal) or symbolic value.

nlink The number of hard links the file is expected to have.

optional
The file is optional; don't complain about the file if it's not in the file hierarchy.

rmd160
The RMD-160 cryptographic message digest of the file.

rmd160digest
Synonym for **rmd160**.

sha1 The SHA-1 cryptographic message digest of the file.

sha1digest
Synonym for **sha1**.

sha256 The 256-bits SHA-2 cryptographic message digest of the file.

sha256digest
Synonym for **sha256**.

sha384 The 384-bits SHA-2 cryptographic message digest of the file.

sha384digest
Synonym for **sha384**.

sha512 The 512-bits SHA-2 cryptographic message digest of the file.

sha512digest
Synonym for **sha512**.

size The size, in bytes, of the file.

tags Comma delimited tags to be matched with **-E** and **-I**. These may be specified without leading or trailing commas, but will be stored internally with them.

time The last modification time of the file.

type The type of the file; may be set to any one of the following:

block	block special device
char	character special device
dir	directory
fifo	fifo
file	regular file
link	symbolic link
socket	socket

uid The file owner as a numeric value.

uname The file owner as a symbolic name.

The default set of keywords are **flags**, **gid**, **link**, **mode**, **nlink**, **size**, **time**, **type**, and **uid**.

There are four types of lines in a specification:

1. Set global values for a keyword. This consists of the string `/set` followed by whitespace, followed by sets of keyword/value pairs, separated by whitespace. Keyword/value pairs consist of a keyword, followed by an equals sign (‘`=`’), followed by a value, without whitespace characters. Once a keyword has been set, its value remains unchanged until either reset or unset.

2. Unset global values for a keyword. This consists of the string `/unset`, followed by whitespace, followed by one or more keywords, separated by whitespace. If `all` is specified, unset all of the keywords.

3. A file specification, consisting of a path name, followed by whitespace, followed by zero or more whitespace separated keyword/value pairs.

   The path name may be preceded by whitespace characters. The path name may contain any of the standard path name matching characters (‘`[`’, ‘`]`’, ‘`?`’ or ‘`*`’), in which case files in the hierarchy will be associated with the first pattern that they match. **mtree** uses `strsvis`(3) (in VIS_CSTYLE format) to encode path names containing non-printable characters. Whitespace characters are encoded as ‘`\s`’ (space), ‘`\t`’ (tab), and ‘`\n`’ (new line). ‘#’ characters in path names are escaped by a preceding backslash ‘\’ to distinguish them from comments.

   Each of the keyword/value pairs consist of a keyword, followed by an equals sign (‘`=`’), followed by the keyword's value, without whitespace characters. These values override, without changing, the global value of the corresponding keyword.

   The first path name entry listed must be a directory named ‘`.`’, as this ensures that intermixing full and relative path names will work consistently and correctly. Multiple entries for a directory named ‘`.`’ are permitted; the settings for the last such entry override those of the existing entry.

   A path name that contains a slash (‘`/`’) that is not the first character will be treated as a full path (relative to the root of the tree). All parent directories referenced in the path name must exist. The current directory path used by relative path names will be updated appropriately. Multiple entries for the same full path are permitted if the types are the same (unless **-M** is given, and then the types may differ); in this case the settings for the last entry take precedence.

A path name that does not contain a slash will be treated as a relative path. Specifying a directory will cause subsequent files to be searched for in that directory hierarchy.

4. A line containing only the string '`..`' which causes the current directory path (used by relative paths) to ascend one level.

Empty lines and lines whose first non-whitespace character is a hash mark ('#') are ignored.

The **mtree** utility exits with a status of 0 on success, 1 if any error occurred, and 2 if the file hierarchy did not match the specification.

FILES

`/etc/mtree` system specification directory

EXAMPLES

To detect system binaries that have been "trojan horsed", it is recommended that **mtree** be run on the file systems, and a copy of the results stored on a different machine, or, at least, in encrypted form. The seed for the **-s** option should not be an obvious value and the final checksum should not be stored on-line under any circumstances! Then, periodically, **mtree** should be run against the on-line specifications and the final checksum compared with the previous value. While it is possible for the bad guys to change the on-line specifications to conform to their modified binaries, it shouldn't be possible for them to make it produce the same final checksum value. If the final checksum value changes, the off-line copies of the specification can be used to detect which of the binaries have actually been modified.

The **-d** and **-u** options can be used in combination to create directory hierarchies for distributions and other such things.

SEE ALSO

`chflags`(1), `chgrp`(1), `chmod`(1), `cksum`(1), `stat`(2), `fnmatch`(3), `fts`(3), `strsvis`(3), `chown`(8), `mknod`(8)

HISTORY

The **mtree** utility appeared in 4.3BSD−Reno. The **optional** keyword appeared in NetBSD 1.2. The **-U** flag appeared in NetBSD 1.3. The **flags** and **md5** keywords, and **-i** and **-m** flags appeared in NetBSD 1.4. The **device**, **rmd160**, **sha1**, **tags**, and **all** keywords, **-D**, **-E**, **-I**, **-l**, **-L**, **-N**, **-P**, **-R**, **-W**, and **-X** flags, and support for full paths appeared in NetBSD 1.6. The **sha256**, **sha384**, and **sha512** keywords appeared in NetBSD 3.0.

NAME

multiboot — procedure for booting NetBSD/i386 from a Multiboot-compliant boot loader

DESCRIPTION

Multiboot is a specification that defines a protocol between a boot loader and a kernel. This protocol allows passing boot information between the two in a standard way, allowing any Multiboot-compliant boot loader to boot any Multiboot-compliant kernel. The NetBSD kernel supports Multiboot if it was compiled with **options MULTIBOOT** (the default in the 'GENERIC' and 'GENERIC_LAPTOP' configurations).

Unlike when using the native boot loader, the NetBSD kernel recognizes a set of command line arguments if booted through a Multiboot-compliant boot loader. This is because the Multiboot protocol is not complete enough to completely configure a NetBSD kernel.

The following arguments are recognized:

console	Specifies the console device name. Can be one of 'com' or 'pc'. If the former, *console_addr* and *console_speed* should be given too.
console_addr	Specifies the serial port address for the console. Defaults to the value of **options CONADDR** or '0x3f8' if this was not given.
console_speed	Specifies the serial port speed for the console. Defaults to the value of **options CONSPEED** or '9600' if this was not given.
root	Specifies the name of the device to be mounted as the root partition. It should not be needed because the kernel tries its best to guess which is the root partition (basing the decision on the device from which the kernel was loaded from). In cases where the automatic detection fails, this flag comes useful. Example: 'root=wd0e'.

Booting with GRUB Legacy

GRUB Legacy is the most popular bootloader that supports Multiboot. You can boot a NetBSD kernel (assuming it is compiled with Multiboot support) with a line similar to the following one:

```
kernel (fd0)/netbsd.gz -c console=pc root=wd0e
```

SEE ALSO

options(4)

HISTORY

multiboot support first appeared in NetBSD 4.0.

AUTHORS

multiboot support was added by Julio M. Merino Vidal ⟨jmmv@NetBSD.org⟩.

NAME

named−checkconf − named configuration file syntax checking tool

SYNOPSIS

named−checkconf [**−v**] [**−j**] [**−t** *directory*] {filename} [**−z**]

DESCRIPTION

named−checkconf checks the syntax, but not the semantics, of a named configuration file.

OPTIONS

−t *directory*

chroot to *directory* so that include directives in the configuration file are processed as if run by a similarly chrooted named.

−v

Print the version of the **named−checkconf** program and exit.

−z

Perform a check load the master zonefiles found in *named.conf*.

−j

When loading a zonefile read the journal if it exists.

filename

The name of the configuration file to be checked. If not specified, it defaults to */etc/named.conf*.

RETURN VALUES

named−checkconf returns an exit status of 1 if errors were detected and 0 otherwise.

SEE ALSO

named(8), BIND 9 Administrator Reference Manual.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

named−checkzone, named−compilezone − zone file validity checking or converting tool

SYNOPSIS

named−checkzone [**−d**] [**−j**] [**−q**] [**−v**] [**−c** *class*] [**−f** *format*] [**−F** *format*] [**−i** *mode*] [**−k** *mode*] [**−m** *mode*] [**−M** *mode*] [**−n** *mode*] [**−o** *filename*] [**−s** *style*] [**−S** *mode*] [**−t** *directory*] [**−w** *directory*] [**−D**] [**−W** *mode*] {zonename} {filename}

named−compilezone [**−d**] [**−j**] [**−q**] [**−v**] [**−c** *class*] [**−C** *mode*] [**−f** *format*] [**−F** *format*] [**−i** *mode*] [**−k** *mode*] [**−m** *mode*] [**−n** *mode*] [**−o** *filename*] [**−s** *style*] [**−t** *directory*] [**−w** *directory*] [**−D**] [**−W** *mode*] {zonename} {filename}

DESCRIPTION

named−checkzone checks the syntax and integrity of a zone file. It performs the same checks as **named** does when loading a zone. This makes **named−checkzone** useful for checking zone files before configuring them into a name server.

named−compilezone is similar to **named−checkzone**, but it always dumps the zone contents to a specified file in a specified format. Additionally, it applies stricter check levels by default, since the dump output will be used as an actual zone file loaded by **named**. When manaully specified otherwise, the check levels must at least be as strict as those specified in the **named** configuration file.

OPTIONS

−d
: Enable debugging.

−q
: Quiet mode − exit code only.

−v
: Print the version of the **named−checkzone** program and exit.

−j
: When loading the zone file read the journal if it exists.

−c *class*
: Specify the class of the zone. If not specified "IN" is assumed.

−i *mode*
: Perform post load zone integrity checks. Possible modes are **"full"** (default), **"full−sibling"**, **"local"**, **"local−sibling"** and **"none"**.

Mode **"full"** checks that MX records refer to A or AAAA record (both in−zone and out−of−zone hostnames). Mode **"local"** only checks MX records which refer to in−zone hostnames.

Mode **"full"** checks that SRV records refer to A or AAAA record (both in−zone and out−of−zone hostnames). Mode **"local"** only checks SRV records which refer to in−zone hostnames.

Mode **"full"** checks that delegation NS records refer to A or AAAA record (both in−zone and out−of−zone hostnames). It also checks that glue addresses records in the zone match those advertised by the child. Mode **"local"** only checks NS records which refer to in−zone hostnames or that some required glue exists, that is when the nameserver is in a child zone.

Mode **"full−sibling"** and **"local−sibling"** disable sibling glue checks but are otherwise the same as **"full"** and **"local"** respectively.

Mode **"none"** disables the checks.

−f *format*
: Specify the format of the zone file. Possible formats are **"text"** (default) and **"raw"**.

−F *format*

Specify the format of the output file specified. Possible formats are **"text"** (default) and **"raw"**. For **named−checkzone**, this does not cause any effects unless it dumps the zone contents.

−k *mode*
: Perform **"check−names"** checks with the specified failure mode. Possible modes are **"fail"** (default for **named−compilezone**), **"warn"** (default for **named−checkzone**) and **"ignore"**.

−m *mode*
: Specify whether MX records should be checked to see if they are addresses. Possible modes are **"fail"**, **"warn"** (default) and **"ignore"**.

−M *mode*
: Check if a MX record refers to a CNAME. Possible modes are **"fail"**, **"warn"** (default) and **"ignore"**.

−n *mode*
: Specify whether NS records should be checked to see if they are addresses. Possible modes are **"fail"** (default for **named−compilezone**), **"warn"** (default for **named−checkzone**) and **"ignore"**.

−o *filename*
: Write zone output to *filename*. This is mandatory for **named−compilezone**.

−s *style*
: Specify the style of the dumped zone file. Possible styles are **"full"** (default) and **"relative"**. The full format is most suitable for processing automatically by a separate script. On the other hand, the relative format is more human−readable and is thus suitable for editing by hand. For **named−checkzone** this does not cause any effects unless it dumps the zone contents. It also does not have any meaning if the output format is not text.

−S *mode*
: Check if a SRV record refers to a CNAME. Possible modes are **"fail"**, **"warn"** (default) and **"ignore"**.

−t *directory*
: chroot to *directory* so that include directives in the configuration file are processed as if run by a similarly chrooted named.

−w *directory*
: chdir to *directory* so that relative filenames in master file $INCLUDE directives work. This is similar to the directory clause in *named.conf*.

−D
: Dump zone file in canonical format. This is always enabled for **named−compilezone**.

−W *mode*
: Specify whether to check for non−terminal wildcards. Non−terminal wildcards are almost always the result of a failure to understand the wildcard matching algorithm (RFC 1034). Possible modes are **"warn"** (default) and **"ignore"**.

zonename
: The domain name of the zone being checked.

filename
: The name of the zone file.

RETURN VALUES

named−checkzone returns an exit status of 1 if errors were detected and 0 otherwise.

SEE ALSO

named(8), RFC 1035, BIND 9 Administrator Reference Manual.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

named − Internet domain name server

SYNOPSIS

named [**−4**] [**−6**] [**−c** *config−file*] [**−d** *debug−level*] [**−f**] [**−g**] [**−n** *#cpus*] [**−p** *port*] [**−s**] [**−t** *directory*] [**−u** *user*] [**−v**] [**−x** *cache−file*]

DESCRIPTION

named is a Domain Name System (DNS) server, part of the BIND 9 distribution from ISC. For more information on the DNS, see RFCs 1033, 1034, and 1035.

When invoked without arguments, **named** will read the default configuration file */etc/named.conf*, read any initial data, and listen for queries.

OPTIONS

−4
: Use IPv4 only even if the host machine is capable of IPv6. **−4** and **−6** are mutually exclusive.

−6
: Use IPv6 only even if the host machine is capable of IPv4. **−4** and **−6** are mutually exclusive.

−c *config−file*
: Use *config−file* as the configuration file instead of the default, */etc/named.conf*. To ensure that reloading the configuration file continues to work after the server has changed its working directory due to to a possible **directory** option in the configuration file, *config−file* should be an absolute pathname.

−d *debug−level*
: Set the daemon's debug level to *debug−level*. Debugging traces from **named** become more verbose as the debug level increases.

−f
: Run the server in the foreground (i.e. do not daemonize).

−g
: Run the server in the foreground and force all logging to *stderr*.

−n *#cpus*
: Create *#cpus* worker threads to take advantage of multiple CPUs. If not specified, **named** will try to determine the number of CPUs present and create one thread per CPU. If it is unable to determine the number of CPUs, a single worker thread will be created.

−p *port*
: Listen for queries on port *port*. If not specified, the default is port 53.

−s
: Write memory usage statistics to *stdout* on exit.

  Note: This option is mainly of interest to BIND 9 developers and may be removed or changed in a future release.

−t *directory*
: **chroot()** to *directory* after processing the command line arguments, but before reading the configuration file.

  Warning: This option should be used in conjunction with the **−u** option, as chrooting a process running as root doesn't enhance security on most systems; the way **chroot()** is defined allows a process with root privileges to escape a chroot jail.

−u *user*
: **setuid()** to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

  Note: On Linux, **named** uses the kernel's capability mechanism to drop all root privileges except the ability to **bind()** to a privileged port and set process resource limits. Unfortunately, this means that the **−u** option only works when **named** is run on kernel 2.2.18 or later, or kernel 2.3.99−pre3 or later, since previous kernels did not allow privileges to be retained after **setuid()**.

−v
: Report the version number and exit.

−x *cache−file*
: Load data from *cache−file* into the cache of the default view.

Warning: This option must not be used. It is only of interest to BIND 9 developers and may be removed or changed in a future release.

SIGNALS

In routine operation, signals should not be used to control the nameserver; **rndc** should be used instead.

SIGHUP
: Force a reload of the server.

SIGINT, SIGTERM
: Shut down the server.

The result of sending any other signals to the server is undefined.

CONFIGURATION

The **named** configuration file is too complex to describe in detail here. A complete description is provided in the BIND 9 Administrator Reference Manual.

FILES

/etc/named.conf
: The default configuration file.

/var/run/named.pid
: The default process−id file.

SEE ALSO

RFC 1033, RFC 1034, RFC 1035, **rndc**(8), **lwresd**(8), **named.conf**(5), BIND 9 Administrator Reference Manual.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME
ncdcs — update the CRC & size in an IBM Network Station 1000 bootable

SYNOPSIS
ncdcs *infile* [*outfile*]

DESCRIPTION
The **ncdcs** utility adds the cyclic redundancy check and image size information to a bootable NetBSD image. The image must contain an NCD firmware header at the very beginning of its text segment.

HISTORY
The **ncdcs** utility came from the Linux IBM Network Station 1000 port by Jochen Roth.

NAME

ndbootd — Sun Network Disk (ND) Protocol server

SYNOPSIS

ndbootd [**−s** *boot2*] [**−i** *interface*] [**−w** *windowsize*] [**−d**] *boot1*

DESCRIPTION

ndbootd is a server which supports the Sun Network Disk (ND) Protocol. This protocol was designed by Sun before they designed NFS. ND simply makes the raw blocks of a disk available to network clients. Contrast this with the true namespace and file abstractions that NFS provides.

The only reason you're likely to encounter ND nowadays is if you have an old Sun 2 machine, like the 2/120 or 2/50. The Sun 2 PROMs can only use ND to boot over the network. (Later, the Sun 3 PROMs would use RARP and TFTP to boot over the network.)

ndbootd is a very simple ND server that only supports client reads for booting. It exports a disk that the clients consider to be `/dev/ndp0` (ND public unit zero). The disk is available only to clients that are listed in `/etc/ethers` and have valid hostnames. (Sun 2 PROMs don't do RARP, but they do learn their IP address from the first ND response they receive from the server.)

boot1 is a file containing the mandatory first-stage network boot program, typically `/usr/mdec/bootyy`. The layout of the exported disk is:

- block 0: normally a Sun disklabel (but ignored by the PROM)
- blocks 1-15: the first-stage network boot program

With the **−s** *boot2* option, **ndbootd** will also make a second-stage network boot program available to clients, typically `/usr/mdec/netboot`. When *boot2* is a filename, that file is the single second-stage network boot program to be served to all clients.

When *boot2* is a directory name, typically `/tftpboot`, **ndbootd** finds a client's second-stage network boot program by turning its IP address into a filename in that directory, in the same manner later Sun 3 PROMs do when TFTPing (i.e., if a client has IP address 192.168.1.10, **ndbootd** expects to find `/tftpboot/C0A8010A.SUN2`).

When used in this last manner with an ND-aware first-stage boot program, **ndbootd** serves the same purpose in the Sun 2 netboot process as `tftpd`(8) serves in the Sun 3 netboot process.

Any second-stage network boot program always begins at block 16 of the exported disk, regardless of the length of the first-stage network boot program.

All first- and second-stage network boot programs must have all executable headers stripped off; they must be raw binary programs.

The remaining options are:

−i *interface*
: Only listen for ND clients on interface *interface*. Normally **ndbootd** listens for clients on the first non-loopback IP interface that is up and running.

−w *windowsize*
: This adjusts the window size of the ND protocol. This is the number of 1-kilobyte packets that can be transmitted before waiting for an acknowledgement. Defaults to 6.

−d
: Run in debug mode. Debugging output goes to standard error and the server will not fork.

FILES

`/etc/ethers`
`/etc/hosts`

SEE ALSO

`tftpd`(8)

BUGS

Whether or not there is a second-stage network boot program, the exported disk appears to all clients to have infinite length. The content of all blocks not used by the first- or second-stage network boot programs is undefined. All client reads of undefined blocks are silently allowed by the server.

NAME

ndiscvt — convert Windows® NDIS drivers for use with NetBSD

SYNOPSIS

ndiscvt [**-O**] [**-i** *inffile*] [**-n** *devname*] [**-o** *outfile*] **-s** *sysfile*

DESCRIPTION

The **ndiscvt** utility transforms a Windows® NDIS driver into a data file which is used to build an ndis compatibility driver module. Windows® drivers consist of two main parts: a `.SYS` file, which contains the actual driver executable code, and an `.INF` file, which provides the Windows® installer with device identifier information and a list of driver-specific registry keys. The **ndiscvt** utility can convert these files into a header file that is compiled into `if_ndis.c` to create an object code module that can be linked into the NetBSD kernel.

The `.INF` file is typically required since only it contains device identification data such as PCI vendor and device IDs or PCMCIA indentifier strings. The `.INF` file may be optionally omitted however, in which case the **ndiscvt** utility will only perform the conversion of the `.SYS` file. This is useful for debugging purposes only.

OPTIONS

The options are as follows:

-i *inffile*
: Open and parse the specified `.INF` file when performing conversion. The **ndiscvt** utility will parse this file and emit a device identification structure and registry key configuration structures which will be used by the ndis driver and ndisapi kernel subsystem. If this is omitted, **ndiscvt** will emit a dummy configuration structure only.

-n *devname*
: Specify an alternate name for the network device/interface which will be created when the driver is instantiated. If you need to load more than one NDIS driver into your system (i.e., if you have two different network cards in your system which require NDIS driver support), each module you create must have a unique name. Device can not be larger than `IFNAMSIZ`. If no name is specified, the driver will use the default a default name ("`ndis`").

-O
: Generate both an `ndis_driver_data.h` file and an `ndis_driver.data.o` file. The latter file will contain a copy of the Windows® `.SYS` driver image encoded as a NetBSD ELF object file (created with `objcopy`(1)). Turning the Windows® driver image directly into an object code file saves disk space and compilation time.

-o *outfile*
: Specify the output file in which to place the resulting data. This can be any file pathname. If *outfile* is a single dash ('**-**'), the data will be written to the standard output. The `if_ndis.c` module expects to find the driver data in a file called `ndis_driver_data.h`, so it is recommended that this name be used.

-s *sysfile*
: Open and parse the specified `.SYS` file. This file must contain a Windows® driver image. The **ndiscvt** utility will perform some manipulation of the sections within the executable file to make runtime linking within the kernel a little easier and then convert the image into a data array.

SEE ALSO

`ld`(1), `objcopy`(1), `ndis`(4)

HISTORY

The **`ndiscvt`** utility first appeared in FreeBSD 5.3.

AUTHORS

The **`ndiscvt`** utility was written by Bill Paul ⟨wpaul@windriver.com⟩. The `lex`(1) and `yacc`(1) `.INF` file parser was written by Matthew Dodd ⟨mdodd@FreeBSD.org⟩.

NAME

ndp — control/diagnose IPv6 neighbor discovery protocol

SYNOPSIS

ndp [**-nt**] *hostname*
ndp [**-nt**] **-a** | **-c** | **-p**
ndp [**-nt**] **-r**
ndp [**-nt**] **-H** | **-P** | **-R**
ndp [**-nt**] **-A** *wait*
ndp [**-nt**] **-d** *hostname*
ndp [**-nt**] **-f** *filename*
ndp [**-nt**] **-i** *interface* [*expressions ...*]
ndp [**-nt**] **-I** [*interface* | delete]
ndp [**-nt**] **-s** *nodename etheraddr* [temp] [proxy]

DESCRIPTION

The **ndp** command manipulates the address mapping table used by the Neighbor Discovery Protocol (NDP).

-a Dump the currently existing NDP entries. The following information will be printed:

Neighbor	IPv6 address of the neighbor.
Linklayer Address	Linklayer address of the neighbor. It could be "`(incomplete)`" when the address is not available.
Netif	Network interface associated with the neighbor cache entry.
Expire	The time until expiry of the entry. The entry could become "`permanent`", in which case it will never expire.
S	State of the neighbor cache entry, as a single letter:

N	Nostate
W	Waitdelete
I	Incomplete
R	Reachable
S	Stale
D	Delay
P	Probe
?	Unknown state (should never happen).

Flags	Flags on the neighbor cache entry, in a single letter. They are: Router, proxy neighbor advertisement ("p"). The field could be followed by a decimal number, which means the number of NS probes the node has sent during the current state.

-A *wait*
Repeat **-a** (dump NDP entries) every *wait* seconds.

-c Erase all the NDP entries.

-d Delete specified NDP entry.

-f Parse the file specified by *filename*.

-H Harmonize consistency between the routing table and the default router list; install the top entry of the list into the kernel routing table.

-I Shows the default interface used as the default route when there is no default router.

-I *interface*
Specifies the default *interface* to be used when there is no interface specified even though required.

-I delete
The current default interface will be deleted from the kernel.

-i *interface* [*expressions ...*]
View ND information for the specified interface. If additional arguments *expressions* are given, **ndp** sets or clears the flags or variables for the interface as specified in the expression. Each expression should be separated by white spaces or tab characters. Possible expressions are as follows. Some of the expressions can begin with the special character '-', which means the flag specified in the expression should be cleared. Note that you need **--** before **-foo** in this case.

nud Turn on or off NUD (Neighbor Unreachability Detection) on the interface. NUD is usually turned on by default.

accept_rtadv
Specify whether or not to accept Router Advertisement messages received on the *interface*. Note that the kernel does not accept Router Advertisement messages unless the net.inet6.ip6.accept_rtadv variable is non-0, even if the flag is on. This flag is set to 1 by default.

prefer_source
Prefer addresses on the *interface* as candidates of the source address for outgoing packets. The default value of this flag is off. For more details about the entire algorithm of source address selection, see the IMPLEMENTATION file supplied with the KAME kit.

disabled
Disable IPv6 operation on the interface. When disabled, the interface discards any IPv6 packets received on or being sent to the interface. In the sending case, an error of ENETDOWN will be returned to the application. This flag is typically set automatically in the kernel as a result of a certain failure of Duplicate Address Detection. While the flag can be set or cleared by hand with the **ndp** command, it is not generally advisable to modify this flag manually.

basereachable=(number)
Specify the BaseReachbleTimer on the interface in millisecond.

retrans=(number)
Specify the RetransTimer on the interface in millisecond.

curhlim=(number)
Specify the Cur Hop Limit on the interface.

-n Do not try to resolve numeric addresses to hostnames.

-p Show prefix list.

-P Flush all the entries in the prefix list.

-r Show default router list.

-R Flush all the entries in the default router list.

-s Register an NDP entry for a node. The entry will be permanent unless the word temp is given in the command. If the word proxy is given, this system will act as a proxy NDP server, responding to requests for *hostname* even though the host address is not its own.

-t Print timestamp on each entry, making it possible to merge output with `tcpdump`(8). Most useful when used with **-A**.

RETURN VALUES

The **ndp** command will exit with 0 on success, and non-zero on errors.

SEE ALSO

`arp`(8)

HISTORY

The **ndp** command first appeared in the WIDE Hydrangea IPv6 protocol stack kit.

NAME

netgroup_mkdb — generate the netgroup database

SYNOPSIS

netgroup_mkdb [**-o** *database*] [file]

DESCRIPTION

netgroup_mkdb creates a db(3) database for the specified file. If no file is specified, then /etc/netgroup is used. This database is installed into /var/db/netgroup.db. The file must be in the correct format (see netgroup(5)).

The options are as follows:

-o *database*
Put the output databases in the named file.

The databases are used by the C library netgroup routines (see getnetgrent(3)).

netgroup_mkdb exits zero on success, non-zero on failure.

FILES

/var/db/netgroup.db	The current netgroup database
/var/db/netgroup.db.tmp	A temporary file
/etc/netgroup	The current netgroup file

SEE ALSO

db(3), getnetgrent(3), netgroup(5)

BUGS

Because **netgroup_mkdb** guarantees not to install a partial destination file it must build a temporary file in the same file system and if successful use rename(2) to install over the destination file.

If **netgroup_mkdb** fails it will leave the previous version of the destination file intact.

NAME

newbtconf — multiple boot-up configurations

SYNOPSIS

newbtconf *new-conf-name* [*orig-conf-name*]
newbtconf init
newbtconf revert

DESCRIPTION

newbtconf is used to set up the system in such a way that the user is offered a selection of environments in which to boot the system up into. The most obvious application being for laptops to provide a network and non-network environment after a successful boot into multi-user mode.

Background

In order to accomplish this task, the files usually associated with establishing the current system's running configuration are replaced with symbolic links which are adjusted with each boot to point to the appropriate file for the desired run-time environment. This is accomplished by directing all of the symbolic links through a directory which itself is a symbolic link (`/etc/etc.current`), to the destination files. At each bootup, the selection made changes which directory `/etc/etc.current` points to.

Through doing this and reloading `/etc/rc.conf` in `/etc/rc` after the link has been established, the resulting run-time configuration is effectively controlled without the need to directly edit any files. The default boot-up environment is selected by manually directing which configuration directory `/etc/etc.default` points to. As opposed to `/etc/etc.current` (which is updated with every boot), `/etc/etc.default` is not automatically updated.

Getting Started

By default, NetBSD only has one boot-up configuration - that set in the file `/etc/rc.conf`. In order to initialize the system for operating in a manner which supports multiple boot configurations, **newbtconf** must be run with an argument of 'init'. This will create two symbolic links `/etc/etc.current` and `/etc/etc.default` to the directory `/etc/etc.network`. The following files are all moved into that directory and symbolic links put in their place, in `/etc`, pointing to `/etc/etc.current/<filename>`:

```
/etc/defaultdomain
/etc/fstab
/etc/ifconfig.*
/etc/inetd.conf
/etc/mrouted.conf
/etc/mygate
/etc/myname
/etc/netstart
/etc/nsswitch.conf
/etc/ntp.conf
/etc/rc.conf
/etc/rc.conf.d
/etc/resolv.conf
```

To test that this has been performed correctly, reboot your system into NetBSD. After the kernel has autoconfigured and tty flags have been set, a prompt should appear, preceded by the following like, looking like this:

```
[network]
Which configuration [network] ?
```

The []'s are used to indicate the default configuration, which can be selected by just pressing return. If there were other configurations available at this stage, you would have 30 seconds to enter that name and press **RETURN**.

Multiple Configurations

Once an initial configuration has been set up, we can proceed to set up further run time environments. This is done by invoking **newbtconf** with the name of the new configuration to be created. By default, this step will use the current configuration files as the basis for this setup unless a second parameter is given - that of the configuration to use as the basis for the new one. Upon completion, a new directory, `/etc/etc.<newname>`, will have been created, priming the directory with the appropriate files for editing. For example, if we do **newbtconf** *nonet network* it would create a directory named `/etc/etc.nonet` and copy all the files from `/etc/etc.network` into that directory. Upon rebooting, we should see:

```
[network] nonet
Which configuration [network] ?
```

To set up the system for booting into the "nonet" configuration, the files in `/etc/etc.nonet` need be edited.

If you wanted to make "nonet" the default configuration when booting, you would need delete the symbolic link `/etc/etc.default` and create a new symbolic link (with the same name) to `/etc/etc.nonet`. Booting up after having made such a change would result in the following being displayed:

```
network [nonet]
Which configuration [nonet] ?
```

No Network

Assuming that we performed the above command successfully, in order to successfully configure NetBSD to not configure interfaces (or generate no errors from attempting to do so), the following settings (at least) should be used in `/etc/etc.nonet/rc.conf`:

```
auto_ifconfig=NO
net_interfaces=NO
```

Of course other networking services, such as NTP, routed, etc, are all expected to be "NO". In general, the only setting that should be "YES" is syslogd, and perhaps cron (if your cron scripts don't need the network) or screenblank/wscons (if applicable). Other actions such as deleting any NFS mounts from `/etc/etc.nonet/fstab` would also need to be undertaken.

Reverting multiple boot configurations

Multiple boot configurations can be deactivated by running **newbtconf** with an argument of **revert**. All the symlinks mentioned above are then removed and the files they point to are copied to their default place. This effectively makes the currently selected configuration the only one active. The symbolic links `/etc/etc.current` and `/etc/etc.default` are also removed so upon rebooting no configuration selection menu is displayed. Note that the previously created configurations (in `/etc/etc.<name>`) are not removed.

FILES

`/etc/etc.current`	Symbolic link to current config directory.
`/etc/etc.default`	Symbolic link to default config directory.
`/etc/defaultdomain`	These files all become symbolic links.

```
/etc/fstab
/etc/ifconfig.*
/etc/inetd.conf
/etc/mrouted.conf
/etc/mygate
/etc/myname
/etc/netstart
/etc/nsswitch.conf
/etc/ntp.conf
/etc/rc.conf
/etc/rc.conf.d
/etc/resolv.conf
```

SEE ALSO

rc.conf(5), rc(8)

HISTORY

The **newbtconf** program first appeared in NetBSD 1.5.

AUTHORS

This shell script was written by Darren Reed ⟨darrenr@NetBSD.org⟩ with initial input from Matthew Green ⟨mrg@NetBSD.org⟩ on how to approach this problem.

BUGS

It presently does not display a count down timer whilst waiting for input to select which configuration and nor does it abort said timer when a key is first pressed.

The management of the overall collection of multiple configurations is much more manual than it ought to be. A general system configuration tool needs to be written to ease their management.

NAME

newdisk — Prepare a new disk to be usable for X680x0

SYNOPSIS

newdisk [**-vnfcp**] [**-m** *mboot*] *raw_device*

DESCRIPTION

newdisk prepares a new hard disk to be bootable from by X680x0. It should NOT be used for floppy disks.

It creates a disk mark for IOCS to determine the disk geometry, writes the primary boot program (mboot), and creates empty partition table. The option are as follows:

-v Verbose mode.

-n Dryrun mode. Nothing is written to the disk.

-f Force. Usually, when **newdisk** detects existing disk mark, it aborts with some error messages. **-f** option prevents this behaviour.

-c Check only. **newdisk** looks at the disk whether it is already marked.

-p Do not create the partition table.

-m *mboot*
Specifies the mboot program to be written.

FILES

`/usr/mdec/mboot` The default primary boot program.

SEE ALSO

boot(8), installboot(8)

HISTORY

The **newdisk** utility first appeared in NetBSD 1.5.

AUTHORS

newdisk was written by MINOURA Makoto ⟨minoura@NetBSD.org⟩.

NAME

newfs — construct a new file system

SYNOPSIS

newfs [**−FINZ**] [**−a** *maxcontig*] [**−B** *byte-order*] [**−b** *block-size*] [**−d** *maxbsize*]
[**−e** *maxbpg*] [**−f** *frag-size*] [**−g** *avgfilesize*] [**−h** *avgfpdir*]
[**−i** *bytes-per-inode*] [**−m** *free-space*] [**−n** *inodes*]
[**−O** *filesystem-format*] [**−o** *optimization*] [**−S** *sector-size*] [**−s** *size*]
[**−T** *disk-type*] [**−v** *volname*] [**−V** *verbose*] *special*

DESCRIPTION

newfs is used to initialize and clear file systems before first use. Before running **newfs** the disk must be labeled using disklabel(8). **newfs** builds a file system on the specified special device basing its defaults on the information in the disk label. Typically the defaults are reasonable, however **newfs** has numerous options to allow the defaults to be selectively overridden.

Options with numeric arguments may contain an optional (case-insensitive) suffix:

b	Bytes; causes no modification. (Default)
k	Kilo; multiply the argument by 1024
m	Mega; multiply the argument by 1048576
g	Giga; multiply the argument by 1073741824

The following options define the general layout policies.

−a *maxcontig*
This sets the obsolete maxcontig parameter.

−B *byte-order*
Specify the metadata byte order of the file system to be created. Valid byte orders are 'be' and 'le'. If no byte order is specified, the file system is created in host byte order.

−b *block-size*
The block size of the file system, in bytes. It must be a power of two. The smallest allowable size is 4096 bytes. The default size depends upon the size of the file system:

file system size	*block-size*
< 20 MB	4 KB
< 1024 MB	8 KB
>= 1024 MB	16 KB

−d *maxbsize*
Set the maximum extent size to *maxbsize*.

−e *maxbpg*
This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. The default is about one quarter of the total blocks in a cylinder group. See tunefs(8) for more details on how to set this option.

−F Create a file system image in *special*. The file system size needs to be specified with "**−s** *size*". No attempts to use or update the disk label will be made.

−f *frag-size*
The fragment size of the file system in bytes. It must be a power of two ranging in value between *block-size*/8 and *block-size*. The optimal *block-size*:*frag-size* ratio is 8:1. Other ratios are possible, but are not recommended, and may produce unpredictable results. The default size depends upon the size of the file system:

file system size	*frag-size*
< 20 MB	0.5 KB
< 1024 MB	1 KB
>= 1024 MB	2 KB

-g *avgfilesize*
The expected average file size for the file system.

-h *avgfpdir*
The expected average number of files per directory on the file system.

-I Do not require that the file system type listed in the disk label is `4.2BSD` or `Apple UFS`.

-i *bytes-per-inode*
This specifies the density of inodes in the file system. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given. The default is to create an inode for every (4 ∗ *frag-size*) bytes of data space:

file system size	*bytes-per-inode*
< 20 MB	2 KB
< 1024 MB	4 KB
>= 1024 MB	8 KB

-m *free-space*
The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 5%. See `tunefs`(8) for more details on how to set this option.

-N Causes the file system parameters to be printed out without really creating the file system.

-n *inodes*
This specifies the number of inodes for the filesystem. If both **-i** and **-n** are specified then **-n** takes precedence.

-O *filesystem-format*
Select the filesystem-format

0 4.3BSD; This option is primarily used to build root file systems that can be understood by older boot ROMs.
1 FFSv1; normal fast-filesystem (default).
2 FFSv2; enhanced fast-filesystem (suited for more than 1 Terabyte capacity, access control lists).

To create an LFS filesystem see `newfs_lfs`(8). To create a Linux Ext2 filesystem see `newfs_ext2fs`(8).

-o *optimization*
Optimization preference; either "space" or "time". The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the value of minfree (see above) is less than 5%, the default is to optimize for space; if the value of minfree is greater than or equal to 5%, the default is to optimize for time. See `tunefs`(8) for more details on how to set this option.

-s *size* The size of the file system in sectors. An 's' suffix will be interpreted as the number of sectors (the default). All other suffixes are interpreted as per other numeric arguments, except that the number is converted into sectors by dividing by the sector size (as specified by **-S** *secsize*) after suffix interpretation.

If no **-s** *size* is specified then the filesystem size defaults to that of the partition, or, if **-F** is specified, the existing file.

If *size* is negative the specified size is subtracted from the default size (reserving space at the end of the partition).

−T *disk-type*
Uses information for the specified disk from `/etc/disktab` instead of trying to get the information from the disk label.

−v *volname*
This specifies that an Apple UFS filesystem should be created with the given volume name.

−V *verbose*
This controls the amount of information written to stdout:

0	No output
1	Overall size and cylinder group details.
2	A progress bar (dots ending at right hand margin).
3	The first few super-block backup sector numbers are displayed before the progress bar.
4	All the super-block backup sector numbers are displayed (no progress bar).

The default is 3. If **−N** is specifed **newfs** stops before outputting the progress bar.

−Z Pre-zeros the file system image created with **−F**. This is necessary if the image is to be used by `vnd`(4) (which doesn't support file systems with 'holes').

The following option overrides the standard sizes for the disk geometry. The default value is taken from the disk label. Changing this default is useful only when using **newfs** to build a file system whose raw image will eventually be used on a different type of disk than the one on which it is initially created (for example on a write-once disk). Note that changing this value from its default will make it impossible for `fsck_ffs`(8) to find the alternative superblocks if the standard superblock is lost.

−S *sector-size*
The size of a sector in bytes (almost never anything but 512). Defaults to 512.

NOTES

The file system is created with 'random' inode generation numbers to improve NFS security.

The owner and group ids of the root node of the new file system are set to the effective uid and gid of the user initializing the file system.

For the **newfs** command to succeed, the disk label should first be updated such that the fstype field for the partition is set to `4.2BSD` or `Apple UFS`, unless **−F** or **−I** is used.

To create and populate a filesystem image within a file use the `makefs`(8) utility.

The partition size is found using `fstat`(2) not by inspecting the disklabel. The block size and fragment size will be written back to the disklabel only if the last character of *special* references the same partition as the minor device number. that provide disk like block and character devices.

SEE ALSO

`fstat`(2), `disktab`(5), `fs`(5), `disklabel`(8), `diskpart`(8), `dumpfs`(8), `fsck_ffs`(8), `makefs`(8), `mount`(8), `mount_mfs`(8), `newfs_ext2fs`(8), `newfs_lfs`(8), `newfs_msdos`(8), `tunefs`(8)

M. McKusick, W. Joy, S. Leffler, and R. Fabry, "A Fast File System for UNIX,", *ACM Transactions on Computer Systems 2*, 3, pp 181-197, August 1984, (reprinted in the BSD System Manager's Manual).

HISTORY

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

NAME

newfs_ext2fs — construct a new Ext2 file system

SYNOPSIS

newfs_ext2fs [**-FINZ**] [**-b** *block-size*] [**-f** *frag-size*] [**-i** *bytes-per-inode*]
[**-m** *free-space*] [**-n** *inodes*] [**-O** *filesystem-format*]
[**-S** *sector-size*] [**-s** *size*] [**-V** *verbose*] [**-v** *volname*] *special*

DESCRIPTION

newfs_ext2fs is used to initialize and clear Ext2 file systems before first use. Before running **newfs_ext2fs** the disk must be labeled using disklabel(8). **newfs_ext2fs** builds a file system on the specified special device basing its defaults on the information in the disk label. Typically the defaults are reasonable, however **newfs_ext2fs** has numerous options to allow the defaults to be selectively overridden.

Options with numeric arguments may contain an optional (case-insensitive) suffix:

b	Bytes; causes no modification. (Default)
k	Kilo; multiply the argument by 1024
m	Mega; multiply the argument by 1048576
g	Giga; multiply the argument by 1073741824

The following options define the general layout policies.

-b *block-size*
The block size of the file system, in bytes. It must be a power of two. The smallest allowable size is 1024 bytes. The default size depends upon the size of the file system:

file system size	*block-size*
<= 512 MB	1 KB
> 512 MB	4 KB

-F Create a file system image in *special*. The file system size needs to be specified with "**-s** *size*". No attempts to use or update the disk label will be made.

-f *frag-size*
The fragment size of the file system in bytes. It must be the same with blocksize because current ext2fs implementation doesn't support fragmentation.

-I Do not require that the file system type listed in the disk label is Linux Ext2.

-i *bytes-per-inode*
This specifies the density of inodes in the file system. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

-m *free-space*
The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 5%.

-N Causes the file system parameters to be printed out without really creating the file system.

-n *inodes*
This specifies the number of inodes for the file system. If both **-i** and **-n** are specified then **-n** takes precedence. The default number of inodes is calculated from a number of blocks in the file system.

-O *filesystem-format*
Select the filesystem-format

0 `GOOD_OLD_REV`; This option is primarily used to build root file systems that can be understood by old or dumb firmwares for bootstrap. (default)

1 `DYNAMIC_REV`; Various extended (and sometimes incompatible) features are enabled (though not all features are supported on NetBSD). Currently only the following features are supported:

RESIZE	Prepare some reserved structures which enable future file system resizing.
FTYPE	Store file types in directory entries to improve performance.
SPARSESUPER	Prepare superblock backups for the `fsck_ext2fs`(8) utility on not all but sparse block groups.
LARGEFILE	Enable files larger than 2G bytes.

−s *size* The size of the file system in sectors. An 's' suffix will be interpreted as the number of sectors (the default). All other suffixes are interpreted as per other numeric arguments, except that the number is converted into sectors by dividing by the sector size (as specified by **−S** *secsize*) after suffix interpretation.

If no **−s** *size* is specified then the filesystem size defaults to that of the partition, or, if **−F** is specified, the existing file.

If *size* is negative the specified size is subtracted from the default size (reserving space at the end of the partition).

−v *volname*
This specifies a volume name for the file system.

−V *verbose*
This controls the amount of information written to stdout:

0 No output
1 Overall size and cylinder group details.
2 A progress bar (dots ending at right hand margin).
3 The first few super-block backup sector numbers are displayed before the progress bar.
4 All the super-block backup sector numbers are displayed (no progress bar).

The default is 3. If **−N** is specifed **newfs_ext2fs** stops before outputting the progress bar.

−Z Pre-zeros the file system image created with **−F**. This is necessary if the image is to be used by `vnd`(4) (which doesn't support file systems with 'holes').

The following option overrides the standard sizes for the disk geometry. The default value is taken from the disk label. Changing this default is useful only when using **newfs_ext2fs** to build a file system whose raw image will eventually be used on a different type of disk than the one on which it is initially created (for example on a write-once disk). Note that changing this value from its default will make it impossible for `fsck_ext2fs`(8) to find the alternative superblocks if the standard superblock is lost.

−S *sector-size*
The size of a sector in bytes (almost never anything but 512). Defaults to 512.

NOTES

There is no option to specify the metadata byte order on the file system to be created because native Ext2 file system is always little endian even on big endian hosts.

The file system is created with 'random' inode generation numbers to improve NFS security.

The owner and group ids of the root node and reserved blocks of the new file system are set to the effective uid and gid of the user initializing the file system.

For the **newfs_ext2fs** command to succeed, the disk label should first be updated such that the fstype field for the partition is set to `Linux Ext2`, unless **-F** or **-I** is used.

The partition size is found using `fstat`(2) not by inspecting the disklabel. The block size and fragment size will be written back to the disklabel only if the last character of *special* references the same partition as the minor device number. that provide disk like block and character devices.

SEE ALSO

`fstat`(2), `disklabel`(5), `disktab`(5), `fs`(5), `disklabel`(8), `diskpart`(8), `fsck_ext2fs`(8), `mount`(8), `mount_ext2fs`(8), `newfs`(8),

http://e2fsprogs.sourceforge.net/ext2intro.html
Remy Card, Theodore Ts'o, and Stephen Tweedie, "Design and Implementation of the Second Extended Filesystem", *The Proceedings of the First Dutch International Symposium on Linux*.

HISTORY

The **newfs_ext2fs** command first appeared in NetBSD 5.0.

AUTHORS

The **newfs_ext2fs** command was written by Izumi Tsutsui ⟨tsutsui@NetBSD.org⟩.

BUGS

The **newfs_ext2fs** command is still experimental and there are few sanity checks.

The **newfs_ext2fs** command doesn't have options to specify each REV1 file system feature independently.

The **newfs_ext2fs** command doesn't support the bad block list accounted by the bad blocks inode.

Many newer Ext2 file system features (especially journaling) are not supported yet.

Some features in file systems created by the **newfs_ext2fs** command might not be recognized properly by the `fsck_ext2fs`(8) utility.

There is no native tool in the NetBSD distribution for resizing Ext2 file systems yet.

NAME

newfs_lfs — construct a new LFS file system

SYNOPSIS

newfs_lfs [*newfs_lfs-options*] *special*

DESCRIPTION

newfs_lfs builds a log-structured file system on the specified special device basing its defaults on the information in the disk label. Before running **newfs_lfs** the disk must be labeled using disklabel(8), the proper fstype is 4.4LFS. Reasonable values for the fsize, bsize, and sgs fields are 1024, 8192, and 7 respectively.

The following options define the general layout policies.

-A Attempt to compute the appropriate segment size using the formula *4 ∗ bandwidth ∗ access time*. The disk is tested for twenty seconds to discover its bandwidth and seek time.

-B *logical-segment-size*
The logical segment size of the file system in bytes. If not specified, the segment size is computed by left-shifting the partition label's block size by the amount indicated in the partition table's segshift. If the disklabel indicates a zero block size or segment shift, a compile-time default segment size of 1M is used.

-b *block-size*
The block size of the file system in bytes. If not specified, the block size is taken from the partition label, or if the partition label indicates 0, a compile-time default of 8K is used.

-F Force creation of an LFS even on a partition labeled as another type. **newfs_lfs** will use compile-time default values for block and fragment size, and segment shift, unless these are overridden by command-line flags.

-f *fragment-size*
The fragment size of the file system in bytes. If not specified, the fragment size is taken from the partition label, or if the partition label indicates 0, a compile-time default of 1K is used.

-I *interleave*
Specify the interleave between segments. The default is zero.

-i The size of an inode block, in bytes. The default is to use the same size as a fragment, or in a v1 filesystem, the same size as a data block.

-L Create a log-structured file system (LFS). This is the default, and this option is provided for compatibility only.

-M *nsegs* Specify *lfs_minfreeseg*, the number of segments left out of the amount allocated to user data. A higher number increases cleaner performance, while a lower number gives more usable space. The default is based on the size of the filesystem, either 5% of the total number of segments or 20 segments, whichever is larger.

-m *free space %*
The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

-N Do not actually create the filesystem.

-O *offset*
Start the first segment this many sectors from the beginning of the partition. The default is zero.

-R *nsegs* Specify *lfs_resvseg*, the number of segments set aside for the exclusive use of the cleaner. A larger figure reduces the likelihood of running out of clean segments, but if is too close to *lfs_minfreeseg*, the cleaner will run without ceasing when the filesystem becomes close to full. The default is the larger of 15 or the quantity *lfs_minfreeseg* / 2 + 1 .

-r *ident* For a v2 filesystem, specify the roll-forward identifier for the filesystem. This identifier, a 32-bit numeric quantity, should be different from that of any LFS that may previously have existed on the same disk. By default the identifier is chosen at random.

-s *size* The size of the file system in sectors.

-v *version*
Make a filesystem with the specified disk layout version. Valid options are 1 or 2 (the default). *Note*, however, that LFS version 1 is deprecated.

SEE ALSO

disktab(5), disklabel(8), diskpart(8), dumplfs(8)

M. Seltzer, K. Bostic, M. McKusick, and C. Staelin, "An Implementation of a Log-Structured File System for UNIX", *Proceedings of the Winter 1993 USENIX Conference*, pp. 315-331, January 25-29, 1993.

J. Matthews, D. Roselli, A. Costello, R. Wang, and T. Anderson, "Improving the Performance of Log-Structured File Systems with Adaptive Methods", *Proceedings of the Sixteenth ACM SOSP*, October 1997.

HISTORY

A **newlfs** command appeared in 4.4BSD, and was renamed to **newfs_lfs** for NetBSD 1.4.

NAME

newfs_msdos — construct a new MS-DOS (FAT) file system

SYNOPSIS

newfs_msdos [**-N**] [**-B** *boot*] [**-F** *FAT-type*] [**-I** *volid*] [**-L** *label*] [**-O** *OEM*]
[**-S** *sector-size*] [**-a** *FAT-size*] [**-b** *block-size*]
[**-c** *cluster-size*] [**-e** *dirents*] [**-f** *format*] [**-h** *heads*] [**-i** *info*]
[**-k** *backup*] [**-m** *media*] [**-n** *FATs*] [**-o** *hidden*] [**-r** *reserved*]
[**-s** *total*] [**-u** *track-size*] *special* [*disktype*]

DESCRIPTION

The **newfs_msdos** utility creates a FAT12, FAT16, or FAT32 file system on device *special*, using disktab(5) entry *disktype* to determine geometry, if required.

The options are as follow:

-N Don't create a file system: just print out parameters.

-B *boot*
Get bootstrap from file.

-F *FAT-type*
FAT type (one of 12, 16, or 32).

-I *volid*
Volume ID.

-L *label*
Volume label (up to 11 characters). The label should consist of only those characters permitted in regular DOS (8+3) filenames. The default is "NO_NAME".

-O *OEM*
OEM string (up to 8 characters). The default is "NetBSD".

-S *sector-size*
Number of bytes per sector. Acceptable values are powers of 2 in the range 512 through 32768.

-a *FAT-size*
Number of sectors per FAT.

-b *block-size*
File system block size (bytes per cluster). This should resolve to an acceptable number of sectors per cluster (see below).

-c *cluster-size*
Sectors per cluster. Acceptable values are powers of 2 in the range 1 through 128.

-e *dirents*
Number of root directory entries (FAT12 and FAT16 only).

-f *format*
Specify a standard (floppy disk) format. The standard formats are (capacities in kilobytes): 160, 180, 320, 360, 640, 720, 1200, 1232, 1440, 2880.

-h *heads*
Number of drive heads.

-i *info*
: Location of the file system info sector (FAT32 only). A value of 0xffff signifies no info sector.

-k *backup*
: Location of the backup boot sector (FAT32 only). A value of 0xffff signifies no backup sector.

-m *media*
: Media descriptor (acceptable range 0xf0 to 0xff).

-n *FATs*
: Number of FATs. Acceptable values are 1 to 16 inclusive. The default is 2.

-o *hidden*
: Number of hidden sectors.

-r *reserved*
: Number of reserved sectors.

-s *total*
: File system size.

-u *track-size*
: Number of sectors per track.

If **newfs_msdos** receives a SIGINFO signal (see the **status** argument for stty(1)), a line will be written to the standard error output indicating the name of the device currently being formatted, the sector number being written, and the total number of sectors to be written.

NOTES

FAT file system parameters occupy a "Boot Sector BPB (BIOS Parameter Block)" in the first of the "reserved" sectors which precede the actual file system. For reference purposes, this structure is presented below.

```
struct bsbpb {
    u_int16_t  bps;           /* [-S] bytes per sector */
    u_int8_t   spc;           /* [-c] sectors per cluster */
    u_int16_t  res;           /* [-r] reserved sectors */
    u_int8_t   nft;           /* [-n] number of FATs */
    u_int16_t  rde;           /* [-e] root directory entries */
    u_int16_t  sec;           /* [-s] total sectors */
    u_int8_t   mid;           /* [-m] media descriptor */
    u_int16_t  spf;           /* [-a] sectors per FAT */
    u_int16_t  spt;           /* [-u] sectors per track */
    u_int16_t  hds;           /* [-h] drive heads */
    u_int32_t  hid;           /* [-o] hidden sectors */
    u_int32_t  bsec;          /* [-s] big total sectors */
};
/* FAT32 extensions */
struct bsxbpb {
    u_int32_t  bspf;          /* [-a] big sectors per FAT */
    u_int16_t  xflg;          /* control flags */
    u_int16_t  vers;          /* file system version */
    u_int32_t  rdcl;          /* root directory start cluster */
    u_int16_t  infs;          /* [-i] file system info sector */
    u_int16_t  bkbs;          /* [-k] backup boot sector */
};
```

EXAMPLES

```
newfs_msdos /dev/rwd1a
```

Create a file system, using default parameters, on `/dev/rwd1a`.

```
newfs_msdos -f 1440 -L foo /dev/rfd0a
```

Create a standard 1.44M file system, with volume label *foo*, on `/dev/rfd0a`.

DIAGNOSTICS

Exit status is 0 on success and 1 on error.

SEE ALSO

`disktab`(5), `disklabel`(8), `fdisk`(8), `newfs`(8)

HISTORY

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

AUTHORS

Robert Nordier ⟨rnordier@FreeBSD.org⟩.

NAME

newfs_sysvbfs — construct a new System V Boot File System

SYNOPSIS

newfs_sysvbfs *special*

DESCRIPTION

newfs_sysvbfs builds a System V boot file system on the specified special device basing its defaults on the information in the disk label. Before running **newfs_sysvbfs** the disk must be labeled using disklabel(8); the proper fstype is "SysVBFS".

SEE ALSO

disklabel(5), disktab(5), disklabel(8), diskpart(8)

HISTORY

A **newfs_sysvbfs** command first appeared in NetBSD 4.0.

BUGS

The sysvbfs support is still experimental and there are few sanity checks.

NAME

newfs_udf — construct a new UDF file system

SYNOPSIS

newfs_udf [**-c**] [**-F**] [**-L** *loglabel*] [**-M**] [**-v** *min_udf*] [**-V** *max_udf*] [**-P** *discid*] [**-s** *size*] [**-S** *setlabel*] [**-t** *gmtoff*] *special*

DESCRIPTION

The **newfs_udf** utility creates an UDF file system on device *special* suitable for the media currently inserted.

The options are as follow:

-c Perform a crude surface check first to weed out disc faults on rewritable media.

-F Force file system construction on non-empty recordable media.

-L *loglabel*
Set the disc logical label to the specified *loglabel*.

-M Disable metadata partition flavour selection.

-v *min_udf*
Select *min_udf* as the minimum UDF version to be supported. Notation "0x201" for UDF version 2.01.

-V *max_udf*
Select *max_udf* as the maximum UDF version to be supported. Notation "0x250" for UDF version 2.50.

-P *discid*
Set the phyisical disc label to the specified *discid*. For strict conformance and interchange dont set this manually.

-s *size*
Ignored for now.

-S *setlabel*
Set the disc set label to the specified *setlabel*. For strict conformance and interchange dont set this manually.

-t *gmtoff*
Use the specified *gmtoff* as gmt time offset for recording times on the disc.

NOTES

The UDF file system is defined for the entire optical medium. It can only function on the entire CD/DVD/BD so the raw partition has to be specified for read/write actions. For **newfs_udf** this means specifying the raw device with the raw partition. i.e. `/dev/rcd0d` or `/dev/rcd0c.`

Some rewritable optical media needs to be formatted first before it can be used by UDF. This can be done using `mmcformat`(8).

The default UDF version is version 2.01 that can be specified if required as "0x201".

EXAMPLES

```
      newfs_udf -S "Encyclopedia" -L "volume 2" -P "copy-nr-1" /dev/rcd0d
```

Create a file system, using the specified names on the device `/dev/rcd0d` with the default UDF version.

```
dd if=/dev/zero of=bigdisk.2048.udf seek=9999999 count=1
vnconfig -c vnd0 bigdisk.2048.udf 2048/1/1/1
newfs_udf -L bigdisk /dev/rvnd0d
```

Create a 4.8 Gb sparse file and configure it using `vnconfig`(8) to be a 2048 sector size disc and create a new UDF file system on `/dev/rvnd0d`.

```
newfs_udf -L "My USB stick" /dev/rsd0d
```

Create a new UDF file system on the inserted USB stick using its 'native' sectorsize of 512.

SEE ALSO

`disktab`(5), `disklabel`(8), `mmcformat`(8), `newfs`(8)

HISTORY

The **`newfs_udf`** command first appeared in NetBSD 5.0.

AUTHORS

Reinoud Zandijk ⟨reinoud@NetBSD.org⟩.

NAME

newsyslog — maintain system log files to manageable sizes

SYNOPSIS

newsyslog [**-nrsvF**] [**-f** *config_file*] [file ...]

DESCRIPTION

newsyslog is a program that should be scheduled to run periodically by cron(8). When it is executed it archives log files if necessary. If a log file is determined to require archiving, **newsyslog** rearranges the files so that "*logfile*" is empty, "*logfile*.0" has the last period's logs in it, "*logfile*.1" has the next to last period's logs in it and so on, up to a user-specified number of archived logs. Optionally the archived logs can be compressed to save space.

A log can be archived for three reasons:

1. It is larger than the configured size (in kilobytes).
2. A configured number of hours have elapsed since the log was last archived.
3. The configured time for rotation of the log occurred within the last 60 minutes.

The granularity of **newsyslog** is dependent on how often it is scheduled to run by cron(8). It is recommended that **newsyslog** be run once hourly.

When starting up, **newsyslog** reads in a configuration file to determine which logs may potentially be archived. By default, this configuration file is /etc/newsyslog.conf. Each line of the file contains information about a particular log file that should be handled by **newsyslog**. Each line has six mandatory fields and three optional fields, with whitespace separating each field. Blank lines or lines beginning with "#" are ignored. The fields of the configuration file are as follows:

logfile_name
Name of the system log file to be archived.

owner:group
This optional field specifies the owner and group for the archive file. The ":" is essential, even if the *owner* or *group* field is left blank. The field may be numeric, or a name which is present in /etc/passwd or /etc/group. For backward compatibility, "." is usable in lieu of ":", however use of this feature is discouraged.

mode Specify the mode of the log file and archives.

ngen Specify the number of archive files to be kept besides the log file itself.

size When the size of the log file reaches *size* kilobytes, the log file will be trimmed as described above. If this field is replaced by an asterisk ('*'), then the size of the log file is not taken into account when determining when to trim the log file.

when The *when* field can consist of an interval, a specific time, or both. If the *when* field is an asterisk ('*') log rotation will depend only on the contents of the *size* field. Otherwise, the *when* field consists of an optional interval in hours, optionally followed by an '@'-sign and a time in a restricted ISO 8601 format or by an '$'-sign and a time specification for logfile rotation at a fixed time once per day, per week or per month.

If a time is specified, the log file will only be trimmed if **newsyslog** is run within one hour of the specified time. If an interval is specified, the log file will be trimmed if that many hours have passed since the last rotation. When both a time and an interval are specified, the log will be trimmed if either condition is met.

There is no provision for specification of a timezone. There is little point in specifying an explicit minutes or seconds component in the current implementation, since the only comparison is 'within the hour'.

ISO 8601 restricted time format

The lead-in character for a restricted ISO 8601 time is an '`@`'-sign. The particular format of the time in restricted ISO 8601 is: [[[[[*cc*]*yy*]*mm*]*dd*][`T`[*hh*[*mm*[*ss*]]]]]. Optional date fields default to the appropriate component of the current date; optional time fields default to midnight; hence if today is January 22, 1999, the following date specifications are all equivalent:

```
'19990122T000000'
'990122T000000'
'0122T000000'
'22T000000'
'T000000'
'T0000'
'T00'
'22T'
'T'
''
```

Day, week and month time format

The lead-in character for day, week and month specification is a '`$`'-sign. The particular format of day, week and month specification is: [*Dhh*], [*Ww*[*Dhh*]] and [*Mdd*[*Dhh*]] respectively. Optional time fields default to midnight. The ranges for day and hour specifications are:

hh	hours, range 0 ... 23
w	day of week, range 0 ... 6, 0 = Sunday
dd	day of month, range 1 ... 31, or the letter *L* or *l* to specify the last day of the month.

Some examples:

$D0	rotate every night at midnight
$D23	rotate every day at 23:00 hr
$W0D23	rotate every week on Sunday at 23:00 hr
$W5D16	rotate every week on Friday at 16:00 hr
$MLD0	rotate at the last day of every month at midnight
$M5D6	rotate on every 5th day of month at 6:00 hr

flags This field specifies any special processing that is required. These flags are parsed in a case insensitive manner. Individual flags and their meanings:

- This flag means nothing - it is used as a spacer when no flags are set.

b The file is a binary file or is not in `syslogd`(8) format: the ASCII message which **`newsyslog`** inserts to indicate that the logs have been trimmed should not be included.

c Create an empty log file if none currently exists.

n No signal should be sent when the log is trimmed.

p The first historical log file (i.e. the historical log file with the suffix ".0") should not be compressed.

j Archived log files should be compressed with `bzip2`(1) to save space.

z Archived log files should be compressed with `gzip`(1) to save space.

path_to_pid_file
This optional field specifies the file name to read to find the daemon process id. If this field is missing, it defaults to the `/var/run/syslogd.pid` file. A signal of type *sigtype* is sent to the process id contained in this *path_to_pid_file* file. This field must start with '/' in order to be recognized properly.

sigtype
This optional field specifies the type of signal to be sent to the daemon process. This may be a numeric or symbolic value. By default a SIGHUP (hang-up) will be sent.

OPTIONS

The following options can be used with newsyslog:

-f *config_file*
Use *config_file* instead of `/etc/newsyslog.conf` as the configuration file.

-n Do not trim the logs, but print out what would be done if this option were not specified: **-n** implies **-v**.

-r Remove the restriction that **newsyslog** must be running as root. When running as a regular user, **newsyslog** will not be able to send a HUP signal to `syslogd`(8), so this option should be used only when debugging or trimming user generated logs.

-s Do not signal daemon processes.

-v Run in verbose mode. In this mode each action that is taken will be printed.

-F Force trimming of the logs, even if the trim conditions have not been met. This option is useful for diagnosing system problems by providing you with fresh logs.

If additional command line arguments are given, **newsyslog** will only examine log files that match those arguments; otherwise, it will examine all files listed in the configuration file.

FILES

`/etc/newsyslog.conf` **newsyslog** configuration file.

SEE ALSO

`bzip2`(1), `gzip`(1), `syslog`(3), `syslogd`(8)

NAME

nfsd — remote NFS server

SYNOPSIS

nfsd [**−6rut**] [**−n** *num_threads*]

DESCRIPTION

nfsd runs on a server machine to service NFS requests from client machines. At least one **nfsd** must be running for a machine to operate as a server.

Unless otherwise specified, four servers for UDP transport are started.

The following options are available:

−r Register the NFS service with rpcbind(8) without creating any servers. This option can be used along with the **−u** or **−t** options to re-register NFS if the portmap server is restarted.

−n Specifies how many server threads to create. The default is 4. A server should run enough threads to handle the maximum level of concurrency from its clients.

−6 Listen to IPv6 requests as well as IPv4 requests. If IPv6 support is not available, nfsd will silently continue and just use IPv4.

−t Serve TCP NFS clients.

−u Serve UDP NFS clients.

For example, "`nfsd -t -u -n 6`" serves UDP and TCP transports using six threads.

nfsd listens for service requests at the port indicated in the NFS server specification; see *Network File System Protocol Specification*, RFC 1094 and *NFS: Network File System Version 3 Protocol Specification*.

The **nfsd** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

nfsstat(1), nfssvc(2), mountd(8), rpcbind(8)

HISTORY

The **nfsd** utility first appeared in 4.4BSD.

NAME

nis, **yp** — description of the NIS (formerly YP) subsystem

SYNOPSIS

ypbind [**−ypset**]
ypbind [**−ypsetme**]

ypset [**−h** *host*] [**−d** *domain*] *server*

yppoll [**−h** *host*] [**−d** *domain*] *mapname*

ypcat [**−kt**] [**−d** *domainname*] *mapname*
ypcat −x

ypmatch [**−kt**] [**−d** *domainname*] *key ... mapname*
ypmatch −x

ypwhich [**−d** *domain*] [[**−t**] **−m** [*mname*] | *host*]
ypwhich −x

ypserv [**−d**] [**−x**]

yppush [**−d** *domainname*] [**−h** *hostname*] [**−v**] *mapname*

ypxfr [**−bcf**] [**−d** *domain*] [**−h** *host*] [**−s** *domain*] [**−C** *tid prog ipadd port*]
mapname

ypinit −m [*domainname*]
ypinit −s *master_server* [*domainname*]

yptest

rpc.yppasswdd [**−noshell**] [**−nogecos**] [**−nopw**] [**−m** *arg1 arg2 ...*]

DESCRIPTION

The NIS subsystem allows network management of passwd and group file entries through the functions getpwent(3) and getgrent(3). NIS also provides hooks for other client programs, such as amd(8) and bootparamd(8), that can use NIS maps.

Password maps in standard YP are insecure, because the pw_passwd field is accessible by any user. A common solution to this is to generate a secure map (using "makedbm -s") which can only be accessed by a client bound to a privileged port. To activate the secure map, see the appropriate comment in /var/yp/Makefile.yp.

The NIS subsystem is conditionally started in /etc/rc. See the /etc/rc.conf file for configuration variables.

SEE ALSO

domainname(1), ypcat(1), ypmatch(1), ypwhich(1), ypclnt(3), group(5), hosts_access(5), nsswitch.conf(5), passwd(5), rc.conf(5), rc(8), ypbind(8), ypinit(8), yppoll(8), yppush(8), ypserv(8), ypset(8), yptest(8), ypxfr(8)

HISTORY

The NIS client subsystem was originally written by Theo de Raadt to be compatible with Sun's implementation. The NIS server suite was originally written by Mats O Jansson.

BUGS

If `ypbind`(8) cannot find a server, the system behaves the same way as Sun’s code: it hangs.

The ‘secure map’ feature is not compatible with non-BSD implementations as found e.g. in Solaris.

NAME

nologin — politely refuse a login

SYNOPSIS

nologin

DESCRIPTION

nologin displays a message that an account is not available and returns a non-zero exit code. It is intended as a replacement shell field for accounts that have been disabled.

SEE ALSO

login(1)

HISTORY

The **nologin** command appeared in 4.4BSD, a free re-implementation was contributed in NetBSD 1.5 by Hubert Feyrer to avoid bloat through the copyright comment.

NAME

nslint - perform consistency checks on dns files

SYNOPSIS

nslint [**-d**] [**-b** *named.boot*] [**-B** *nslint.boot*]
nslint [**-d**] [**-c** *named.conf*] [**-C** *nslint.conf*]

DESCRIPTION

Nslint reads the nameserver configuration files and performs a number of consistency checks on the dns records. If any problems are discovered, error messages are displayed on *stderr* and **nslint** exits with a non-zero status.

Here is a short list of errors **nslint** detects:

Records that are malformed.

Names that contain dots but are missing a trailing dot.

PTR records with names that are missing a trailing dot.

Names that contain illegal characters (rfc1034).

A records without matching **PTR** records

PTR records without matching **A** records

Names with more than one address on the same subnet.

Addresses in use by more than one name.

Names with **CNAME** and other records (rfc1033).

Unknown service and/or protocol keywords in **WKS** records.

Missing quotes.

OPTIONS

-b Specify an alternate *named.boot* file. The default is */etc/named.boot*.

-c Specify an alternate *named.conf* file. The default is */etc/named.conf*.

-B Specify an alternate *nslint.boot* file. The default is *nslint.boot* in the last **directory** line processed in *named.boot* (or the current working directory). This file is processed like a second *named.boot*. The most common use is to tell **nslint** about **A** records that match **PTR** records that point outside the domains listed in *named.boot*.

-C Specify an alternate *nslint.conf* file. The default is *nslint.conf* in the last **directory** line processed in *named.conf* (or the current working directory). This file is processed like a second *named.conf*.

-d Raise the debugging level. Debugging information is displayed on *stdout*.

Nslint knows how to read old style *named.boot* and BIND 8's new *named.conf* files. If both files exist, **nslint** will prefer *named.conf* (on the theory that you forgot to delete *named.boot* when you upgraded to BIND 8).

ADVANCED CONFIGURATION

There are some cases where it is necessary to use the advanced configuration features of **nslint**. Advanced configuration is done with the *nslint.boot* file.

The most common is when a site has a demilitarized zone (DMZ). The problem here is that the DMZ network will have **PTR** records for hosts outside its domain. For example lets say we have *128.0.rev* with:

```
1.1    604800  in    ptr    gateway.lbl.gov.
2.1    604800  in    ptr    gateway.es.net.
```

Obviously we will define an **A** record for *gateway.lbl.gov* pointing to *128.0.1.1* but we will get errors because there is no **A** record defined for *gateway.es.net*. The solution is to create a *nslint.boot* file (in the same directory as the other dns files) with:

```
primary es.net          nslint.es.net
```

And then create the file *nslint.es.net* with:

```
gateway 1    in    a    128.0.1.2
```

Another problem occurs when there is a **CNAME** that points to a host outside the local domains. Let's say we have *info.lbl.gov* pointing to *larry.es.net*:

```
info   604800  in    cname   larry.es.net.
```

In this case we would need:

```
primary es.net          nslint.es.net
```

in *nslint.boot* and:

```
larry   1    in    txt    "place holder"
```

nslint.es.net.

One last problem when a pseudo host is setup to allow two more more actual hosts provide a service. For, let's say that *lbl.gov* contains:

```
server  604800  in    a    128.0.6.6
server  604800  in    a    128.0.6.94
;
tom     604800  in    a    128.0.6.6
tom     604800  in    mx 0   lbl.gov.
;
jerry   604800  in    a    128.0.6.94
jerry   604800  in    mx 0   lbl.gov.
```

In this case **nslint** would complain about missing **PTR** records and ip addresses in use by more than one host. To suppress these warnings, add you would the lines:

```
primary lbl.gov              nslint.lbl.gov
primary 0.128.in-addr.arpa     nslint.128.0.rev
```

to *nslint.boot* and create *nslint.lbl.gov* with:

```
server  1    in    allowdupa    128.0.6.6
server  1    in    allowdupa    128.0.6.94
```

and create *nslint.128.0.rev* with:

```
6.6    604800  in    ptr    server.lbl.gov.
94.6    604800  in    ptr    server.lbl.gov.
```

In this example, the **allowdupa** keyword tells **nslint** that it's ok for *128.0.6.6* and *128.0.6.94* to be shared by *server.lbl.gov*, *tom.lbl.gov*, and *jerry.lbl.gov*.

One last **nslint** feature helps detect hosts that have mistakenly had two ip addresses assigned on the same subnet. This can happen when two different people request an ip address for the same hostname or when someone forgets an address has been assigned and requests a new number.

To detect such **A** records, add a **nslint** section to your *nslint.conf* containing something similar to:

```
nslint {
        network "128.0.6/22";
        network "128.0.6 255.255.252.0";
};
```

The two network lines in this example are equivalent ways of saying the same thing; that subnet *128.0.6* has a 22 bit wide subnet mask.

If you are using *nslint.boot*, the syntax would be:

```
network 128.0.6/22
network 128.0.6 255.255.252.0
```

Again this shows two ways of saying the same thing.

Using information from the above **network** statement, **nslint** would would flag the following **A** records as being in error:

```
server  1     in    a     128.0.6.48
server  1     in    a     128.0.7.16
```

Note that if you specify any **network** lines in your *nslint.conf* or *nslint.boot* files, **nslint** requires you to include lines for all networks; otherwise you might forget to add **network** lines for new networks.

FILES

/etc/named.boot - default named configuration file
nslint.boot - default nslint configuration file

SEE ALSO

named(8), rfc1033, rfc1034

AUTHOR

Craig Leres of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA.

The current version is available via anonymous ftp:

ftp://ftp.ee.lbl.gov/nslint.tar.gz

BUGS

Please send bug reports to nslint@ee.lbl.gov.

Not everyone is guaranteed to agree with all the checks done.

NAME

nslookup — query Internet name servers interactively

SYNOPSIS

nslookup [**−option** *. . .*] [*host-to-find* | **−**[*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.

The options listed under the "`set`" command below can be specified in the `.nslookuprc` file in the user's home directory if they are listed one per line. 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:

```
nslookup -query=hinfo  -timeout=10
```

INTERACTIVE COMMANDS

Commands may be interrupted at any time by typing a control-C. To exit, type a control-D (`EOF`) or type `exit`. The command line length must be less than 256 characters. To treat a built-in command as a host name, precede it with an escape character ('\'). *N.B.:* unrecognized command will be interpreted as a host name.

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 default domain name is appended to the name. (This behavior depends on the state of the **set** options **domain**, **srchlist**, **defname**, and **search**.)

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 Changes the default server to the server for the root of the domain name space. Currently, the host `ns.internic.net` is used. (This command is a synonym for "**lserver ns.internic.net**".) The name of the root server can be changed with the "**set root**" command.

finger [*name*] [**>** *filename*]

finger [*name*] [**>>** *filename*]

Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and returned address information (see the "**set querytype**=A" command). The *name* is optional. **>** and **>>** can be used to redirect output in the usual manner.

ls [*option*] *domain* [**>** *filename*]

ls [*option*] *domain* [**>>** *filename*]

List the information available for *domain*, optionally creating or appending to *filename*. The default output contains host names and their Internet addresses. *Option* can be one of the following:

-t *querytype*
lists all records of the specified type (see *querytype* below).

-a lists aliases of hosts in the domain; synonym for "**-t** CNAME".

-d lists all records for the domain; synonym for "**-t** ANY".

-h lists CPU and operating system information for the domain; synonym for "**-t** HINFO".

-s lists well-known services of hosts in the domain; synonym for "**-t** WKS".

When output is directed to a file, hash marks are printed for every 50 records received from the server.

view *filename*

Sorts and lists the output of previous **ls** command(s) with more(1).

help

? Prints a brief summary of commands.

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
CHAOS	the Chaos class
HESIOD	the MIT Athena Hesiod class
ANY	wildcard (any of the above)

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 exhaustive debugging mode on. Essentially all fields of every packet are printed.

(Default = **nod2**)

domain=*name*
Change the default domain name to *name*. The default domain name is appended to a lookup request depending on the state of the **defname** and **search** options. The domain search list contains the parents of the default domain if it has at least two components in its name. For example, if the default domain is CC.Berkeley.EDU, the search list is CC.Berkeley.EDU and Berkeley.EDU. Use the "**set srchlist**" command to specify a different list. Use the "**set all**" command to display the list.

(Default = value from hostname(1), /etc/resolv.conf, or LOCALDOMAIN; abbreviation = **do**)

srchlist=*name1*/*name2*/*...*
Change the default domain name to *name1* and the domain search list to *name1*, *name2*, etc. A maximum of 6 names separated by slashes ('/') can be specified. For example,

```
set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU
```

sets the domain to lcs.MIT.EDU and the search list to the three names. This command overrides the default domain name and search list of the "**set domain**" command. Use the "**set all**" command to display the list.

(Default = value based on hostname(1), /etc/resolv.conf, or LOCALDOMAIN; abbreviation = **srchl**)

[**no**]**defname**
If set, append the default domain name to a single-component lookup request (i.e., one that does not contain a period).

(Default = **defname**; abbreviation = [**no**]**defname**)

[**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**; abbreviation = [**no**]**sea**)

port=*value*
Change the default TCP/UDP name server port to *value*.

(Default = 53; abbreviation = **po**)

querytype=*value*

type=*value*
Change the type of information query to one of:

A	the host's Internet address.
CNAME	the canonical name for an alias.

`HINFO`	the host CPU and operating system type.
`MINFO`	the mailbox or mail list information.
`MX`	the mail exchanger.
`NS`	the name server for the named zone.
`PTR`	the host name if the query is an Internet address; otherwise, the pointer to other information.
`SOA`	the domain's "start-of-authority" information.
`TXT`	the text information.
`UINFO`	the user information.
`WKS`	the supported well-known services.

Other types (`ANY`, `AXFR`, `MB`, `MD`, `MF`, `NULL`) are described in the RFC-1035 document.

(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`*. When a reply to a request is not received within a certain amount of time (changed with "**`set timeout`**"), the timeout period is doubled and the request is resent. The retry value controls how many times a request is resent before giving up.

(Default = 4, abbreviation = **`ret`**)

`root`=*`host`*
Change the name of the root server to *`host`*. This affects the "**`root`**" command.

(Default = **`ns.internic.net.`**; abbreviation = **`ro`**)

`timeout`=*`number`*
Change the initial timeout interval for waiting for a reply to *`number`* seconds. Each retry doubles the timeout period.

(Default = 5 seconds; abbreviation = **`ti`**)

[**`no`**]**`vc`** Always use a virtual circuit when sending requests to the server.

(Default = **`novc`**; abbreviation = [**`no`**]**`v`**)

[**`no`**]**`ignoretc`**
Ignore packet truncation errors.

(Default = **`noignoretc`**; abbreviation = [**`no`**]**`ig`**)

ENVIRONMENT

`HOSTALIASES`	file containing host aliases
`LOCALDOMAIN`	overrides default domain

FILES

`/etc/resolv.conf`	initial domain name and name server addresses
`$HOME/.nslookuprc`	user's initial options
`/usr/share/misc/nslookup.help`	summary of commands

DIAGNOSTICS

If the lookup request was not successful, an error message is printed. Possible errors are:

`Timed out`
: The server did not respond to a request after a certain amount of time (changed with "**set timeout**=*value*") and a certain number of retries (changed with "**set retry**=*value*").

`No response from server`
: No name server is running on the server machine.

`No records`
: The server does not have resource records of the current query type for the host, although the host name is valid. The query type is specified with the "**set querytype**" command.

`Non-existent domain`
: The host or domain name does not exist.

`Connection refused`

`Network is unreachable`
: The connection to the name or finger server could not be made at the current time. This error commonly occurs with **ls** and **finger** requests.

`Server failure`
: The name server found an internal inconsistency in its database and could not return a valid answer.

`Refused`
: The name server refused to service the request.

`Format error`
: The name server found that the request packet was not in the proper format. It may indicate an error in **nslookup**.

SEE ALSO

resolver(3), resolv.conf(5), named(8)

P.V. Mockapetris, *Domain Names - Concepts and Facilities*, RFC 1034, Nov 1, 1987.

P.V. Mockapetris, *Domain Names - Implementation and Specification*, RFC 1035, Nov 1, 1987.

AUTHORS

Andrew Cherenson

NAME

nsupdate − Dynamic DNS update utility

SYNOPSIS

nsupdate [**−d**] [[**−y** *[hmac:]keyname:secret*] | [**−k** *keyfile*]] [**−t** *timeout*] [**−u** *udptimeout*] [**−r** *udpretries*] [**−v**] [filename]

DESCRIPTION

nsupdate is used to submit Dynamic DNS Update requests as defined in RFC2136 to a name server. This allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Zones that are under dynamic control via **nsupdate** or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

The resource records that are dynamically added or removed with **nsupdate** have to be in the same zone. Requests are sent to the zone's master server. This is identified by the MNAME field of the zone's SOA record.

The **−d** option makes **nsupdate** operate in debug mode. This provides tracing information about the update requests that are made and the replies received from the name server.

Transaction signatures can be used to authenticate the Dynamic DNS updates. These use the TSIG resource record type described in RFC2845 or the SIG(0) record described in RFC3535 and RFC2931. TSIG relies on a shared secret that should only be known to **nsupdate** and the name server. Currently, the only supported encryption algorithm for TSIG is HMAC−MD5, which is defined in RFC 2104. Once other algorithms are defined for TSIG, applications will need to ensure they select the appropriate algorithm as well as the key when authenticating each other. For instance suitable **key** and **server** statements would be added to */etc/named.conf* so that the name server can associate the appropriate secret key and algorithm with the IP address of the client application that will be using TSIG authentication. SIG(0) uses public key cryptography. To use a SIG(0) key, the public key must be stored in a KEY record in a zone served by the name server. **nsupdate** does not read */etc/named.conf*.

nsupdate uses the **−y** or **−k** option to provide the shared secret needed to generate a TSIG record for authenticating Dynamic DNS update requests, default type HMAC−MD5. These options are mutually exclusive. With the **−k** option, **nsupdate** reads the shared secret from the file *keyfile*, whose name is of the form *K{name}.+157.+{random}.private*. For historical reasons, the file *K{name}.+157.+{random}.key* must also be present. When the **−y** option is used, a signature is generated from [*hmac:*]*keyname:secret. keyname* is the name of the key, and *secret* is the base64 encoded shared secret. Use of the **−y** option is discouraged because the shared secret is supplied as a command line argument in clear text. This may be visible in the output from **ps**(1) or in a history file maintained by the user's shell.

The **−k** may also be used to specify a SIG(0) key used to authenticate Dynamic DNS update requests. In this case, the key specified is not an HMAC−MD5 key.

By default **nsupdate** uses UDP to send update requests to the name server unless they are too large to fit in a UDP request in which case TCP will be used. The **−v** option makes **nsupdate** use a TCP connection. This may be preferable when a batch of update requests is made.

The **−t** option sets the maximum time a update request can take before it is aborted. The default is 300 seconds. Zero can be used to disable the timeout.

The **−u** option sets the UDP retry interval. The default is 3 seconds. If zero the interval will be computed from the timeout interval and number of UDP retries.

The **−r** option sets the number of UDP retries. The default is 3. If zero only one update request will be made.

INPUT FORMAT

nsupdate reads input from *filename* or standard input. Each command is supplied on exactly one line of input. Some commands are for administrative purposes. The others are either update instructions or prerequisite checks on the contents of the zone. These checks set conditions that some name or set of resource records (RRset) either exists or is absent from the zone. These conditions must be met if the entire update request is to succeed. Updates will be rejected if the tests for the prerequisite conditions fail.

Every update request consists of zero or more prerequisites and zero or more updates. This allows a suitably authenticated update request to proceed if some specified resource records are present or missing from the zone. A blank input line (or the **send** command) causes the accumulated commands to be sent as one Dynamic DNS update request to the name server.

The command formats and their meaning are as follows:

server {servername} [port]
: Sends all dynamic update requests to the name server *servername*. When no server statement is provided, **nsupdate** will send updates to the master server of the correct zone. The MNAME field of that zone's SOA record will identify the master server for that zone. *port* is the port number on *servername* where the dynamic update requests get sent. If no port number is specified, the default DNS port number of 53 is used.

local {address} [port]
: Sends all dynamic update requests using the local *address*. When no local statement is provided, **nsupdate** will send updates using an address and port chosen by the system. *port* can additionally be used to make requests come from a specific port. If no port number is specified, the system will assign one.

zone {zonename}
: Specifies that all updates are to be made to the zone *zonename*. If no *zone* statement is provided, **nsupdate** will attempt determine the correct zone to update based on the rest of the input.

class {classname}
: Specify the default class. If no *class* is specified the default class is *IN*.

key {name} {secret}
: Specifies that all updates are to be TSIG signed using the *keyname keysecret* pair. The **key** command overrides any key specified on the command line via **−y** or **−k**.

prereq nxdomain {domain−name}
: Requires that no resource record of any type exists with name *domain−name*.

prereq yxdomain {domain−name}
: Requires that *domain−name* exists (has as at least one resource record, of any type).

prereq nxrrset {domain−name} [class] {type}
: Requires that no resource record exists of the specified *type*, *class* and *domain−name*. If *class* is omitted, IN (internet) is assumed.

prereq yxrrset {domain−name} [class] {type}
: This requires that a resource record of the specified *type*, *class* and *domain−name* must exist. If *class* is omitted, IN (internet) is assumed.

prereq yxrrset {domain−name} [class] {type} {data...}
: The *data* from each set of prerequisites of this form sharing a common *type*, *class*, and *domain−name* are combined to form a set of RRs. This set of RRs must exactly match the set of RRs existing in the zone at the given *type*, *class*, and *domain−name*. The *data* are written in the standard text representation of the resource record's RDATA.

update delete {domain−name} [ttl] [class] [type [data...]]
: Deletes any resource records named *domain−name*. If *type* and *data* is provided, only matching resource records will be removed. The internet class is assumed if *class* is not supplied. The *ttl* is ignored, and is only allowed for compatibility.

update add {domain−name} {ttl} [class] {type} {data...}
: Adds a new resource record with the specified *ttl*, *class* and *data*.

show

Displays the current message, containing all of the prerequisites and updates specified since the last send.

send
Sends the current message. This is equivalent to entering a blank line.

answer
Displays the answer.

Lines beginning with a semicolon are comments and are ignored.

EXAMPLES

The examples below show how **nsupdate** could be used to insert and delete resource records from the **example.com** zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for **example.com**.

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
> send
```

Any A records for **oldhost.example.com** are deleted. and an A record for **newhost.example.com** it IP address 172.16.1.1 is added. The newly−added record has a 1 day TTL (86400 seconds)

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.example.com
> send
```

The prerequisite condition gets the name server to check that there are no resource records of any type for **nickname.example.com**. If there are, the update request fails. If this name does not exist, a CNAME for it is added. This ensures that when the CNAME is added, it cannot conflict with the long−standing rule in RFC1034 that a name must not exist as any other record type if it exists as a CNAME. (The rule has been updated for DNSSEC in RFC2535 to allow CNAMEs to have RRSIG, DNSKEY and NSEC records.)

FILES

/etc/resolv.conf
used to identify default name server

K{name}.+157.+{random}.key
base−64 encoding of HMAC−MD5 key created by **dnssec−keygen**(8).

K{name}.+157.+{random}.private
base−64 encoding of HMAC−MD5 key created by **dnssec−keygen**(8).

SEE ALSO

RFC2136(), **RFC3007**(), **RFC2104**(), **RFC2845**(), **RFC1034**(), **RFC2535**(), **RFC2931**(), **named**(8), **dnssec−keygen**(8).

BUGS

The TSIG key is redundantly stored in two separate files. This is a consequence of nsupdate using the DST library for its cryptographic operations, and may change in future releases.

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

see html/copyright.html

This manual page was *AutoGen*-erated from the **ntpd** option definitions.

NAME

ntpdate — set the date and time via NTP

SYNOPSIS

ntpdate [**−bBdoqsuv**] [**−a** *key*] [**−e** *authdelay*] [**−k** *keyfile*] [**−o** *version*]
[**−p** *samples*] [**−t** *timeout*] [*server ...*]

DESCRIPTION

ntpdate sets the local date and time by polling the Network Time Protocol (NTP) server(s) given as the *server* arguments to determine the correct time. It must be run as root on the local host. A number of samples are obtained from each of the servers specified and a subset of the NTP clock filter and selection algorithms are applied to select the best of these. Note that the accuracy and reliability of **ntpdate** depends on the number of servers, the number of polls each time it is run and the interval between runs.

ntpdate can be run manually as necessary to set the host clock, or it can be run from the host startup script to set the clock at boot time. This is useful in some cases to set the clock initially before starting the NTP daemon ntpd. It is also possible to run **ntpdate** from a cron script. However, it is important to note that **ntpdate** with contrived cron scripts is no substitute for the NTP daemon, which uses sophisticated algorithms to maximize accuracy and reliability while minimizing resource use. Finally, since **ntpdate** does not discipline the host clock frequency as does ntpd, the accuracy using **ntpdate** is limited.

Time adjustments are made by **ntpdate** in one of two ways. If **ntpdate** determines the clock is in error more than 0.5 second it will simply step the time by calling the system settimeofday(2) routine. If the error is less than 0.5 seconds, it will slew the time by calling the system adjtime(2) routine. The latter technique is less disruptive and more accurate when the error is small, and works quite well when **ntpdate** is run by cron every hour or two.

ntpdate will decline to set the date if an NTP server daemon (e.g., ntpd) is running on the same host. When running **ntpdate** on a regular basis from cron as an alternative to running a daemon, doing so once every hour or two will result in precise enough timekeeping to avoid stepping the clock.

If NetInfo support is compiled into **ntpdate**, then the server argument is optional if **ntpdate** can find a time server in the NetInfo configuration for ntpd

COMMAND LINE OPTIONS

−a *key*
Enable the authentication function and specify the key identifier to be used for authentication as the argument *key* **ntpdate**. The keys and key identifiers must match in both the client and server key files. The default is to disable the authentication function.

−B Force the time to always be slewed using the adjtime() system call, even if the measured offset is greater than +-128 ms. The default is to step the time using settimeofday() if the offset is greater than +-128 ms. Note that, if the offset is much greater than +-128 ms in this case, that it can take a long time (hours) to slew the clock to the correct value. During this time. the host should not be used to synchronize clients.

−b Force the time to be stepped using the settimeofday() system call, rather than slewed (default) using the adjtime() system call. This option should be used when called from a startup file at boot time.

−d Enable the debugging mode, in which **ntpdate** will go through all the steps, but not adjust the local clock. Information useful for general debugging will also be printed.

−e *authdelay*
Specify the processing delay to perform an authentication function as the value *authdelay* , in seconds and fraction (see ntpd for details). This number is usually small enough to be negligible

for most purposes, though specifying a value may improve timekeeping on very slow CPU's.

−k *keyfile*
: Specify the path for the authentication key file as the string *keyfile* The default is `/etc/ntp.keys`. This file should be in the format described in `ntpd`

−o *version*
: Specify the NTP version for outgoing packets as the integer *version* , which can be 1 or 2. The default is 3. This allows **ntpdate** to be used with older NTP versions.

−p *samples*
: Specify the number of samples to be acquired from each server as the integer *samples* , with values from 1 to 8 inclusive. The default is 4.

−q Query only - don't set the clock.

−s Divert logging output from the standard output (default) to the system `syslog` facility. This is designed primarily for convenience of `cron` scripts.

−t *timeout*
: Specify the maximum time waiting for a server response as the value *timeout* , in seconds and fraction. The value is is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

−u Direct **ntpdate** to use an unprivileged port for outgoing packets. This is most useful when behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronise with hosts beyond the firewall. Note that the **−d** option always uses unprivileged ports.

−v Be verbose. This option will cause **ntpdate** string to be logged.

FILES

`/etc/ntp.keys` encryption keys used by **ntpdate**.

AUTHORS

David L. Mills (mills@udel.edu)

BUGS

The slew adjustment is actually 50% larger than the measured offset, since this (it is argued) will tend to keep a badly drifting clock more accurate. This is probably not a good idea and may cause a troubling hunt for some values of the kernel variables `tick` and `tickadj`.

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

see /usr/share/doc/html/ntp/copyright.html

This manual page was *AutoGen*-erated from the **ntpdc** 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

ntptime — read kernel time variables

SYNOPSIS

ntptime [**-chr**] [**-e** *est_error*] [**-f** *frequency*] [**-m** *max_error*] [**-o** *offset*]
[**-s** *status*] [**-t** *time_constant*]

DESCRIPTION

This program is useful only with special kernels described in the *A Kernel Model for Precision Timekeeping* page in `/usr/share/doc/html/ntp/kern.html`. It reads and displays time-related kernel variables using the `ntp_gettime`(2) system call. A similar display can be obtained using the **ntpdc** program and `kerninfo` command.

OPTIONS

-c Display the execution time of **ntptime** itself.

-e *est_error*
Specify estimated error, in microseconds.

-f *frequency*
Specify frequency offset, in parts per million.

-h Display times in Unix timeval format. Default is NTP format.

-l Specify the leap bits as a code from 0 to 3.

-m *max_error*
Display help information.

-o *offset*
Specify clock offset, in microseconds.

-r Display Unix and NTP times in raw format.

-s *status*
Specify clock status. Better know what you are doing.

-t *time_constant*
Specify time constant, an integer in the range 0-4.

AUTHORS

David L. Mills (mills@udel.edu)

NAME

ntptrace — trace a chain of NTP servers back to the primary source

SYNOPSIS

ntptrace [**−vdn**] [**−r** *retries*] [**−t** *timeout*] [*server*]

DESCRIPTION

ntptrace determines where a given Network Time Protocol (NTP) server gets its time from, and follows the chain of NTP servers back to their master time source. If given no arguments, it starts with `localhost`. Here is an example of the output from **ntptrace**:

```
% ntptrace
localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2ozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid 'WWVB'
```

On each line, the fields are (left to right): the host name, the host stratum, the time offset between that host and the local host (as measured by **ntptrace** ; this is why it is not always zero for `localhost`), the host synchronization distance, and (only for stratum-1 servers) the reference clock ID. All times are given in seconds. Note that the stratum is the server hop count to the primary source, while the synchronization distance is the estimated error relative to the primary source. These terms are precisely defined in RFC-1305.

OPTIONS

−d Turns on some debugging output.

−n Turns off the printing of host names; instead, host IP addresses are given. This may be useful if a nameserver is down.

−r *retries*
Sets the number of retransmission attempts for each host (default = 5).

−t *timeout*
Sets the retransmission timeout (in seconds) (default = 2).

−v Prints verbose information about the NTP servers.

AUTHORS

David L. Mills (mills@udel.edu)

BUGS

This program makes no attempt to improve accuracy by doing multiple samples.

NAME

ofctl — display the OpenPROM or OpenFirmware device tree

SYNOPSIS

ofctl [**-p**] [**-f** *file*] [*node*]

DESCRIPTION

ofctl provides an interface for displaying the OpenPROM or OpenFirmware device tree and node properties. Without any arguments, **ofctl** will dump the full tree. When given the name of a specific node, **ofctl** will display that node and its child nodes.

The options are as follows:

-f *file*
: On systems with OpenPROM, use *file* instead of the default `/dev/openprom`. On systems with OpenFirmware, use *file* instead of the default `/dev/openfirm`.

-p
: Display each node's properties.

FILES

/dev/openprom	The openprom device on systems with OpenPROM.
/dev/openfirm	The openfirm device on systems with OpenFirmware.

SEE ALSO

eeprom(8)

NAME

ofwboot, **ofwboot.elf**, **ofwboot.xcf** — Open Firmware boot command

SYNOPSIS

ofwboot

DESCRIPTION

Open Firmware is a FORTH-like command interpreter started by the BootROM after the power-on self test (POST). This command interpreter allows the user flexibility in choosing how their machine boots an operating system. NetBSD uses Open Firmware to initialize many of the devices in a system and uses it to load the primary bootloader, **ofwboot**.

The information in this man page should only serve as a guideline for users. Apple has made many revisions to Open Firmware, and the earlier versions had many problems and inconsistencies. You may find that a boot command that works on one model will not work on another.

In this man page, only one Open Firmware command will be described, **boot**, because it is used to pass arguments to **ofwboot**. The Open Firmware **boot** command takes up to three arguments:

boot [*boot-device* [*boot-file*]] [*options*]

boot-device	primary bootloader location
boot-file	kernel location
options	flags passed to the kernel (see below)

boot-device

The first argument, *boot-device*, actually designates the primary bootloader location and its name in the form:

```
      boot-device:[partition-num],[bootloader-filename]
```

A typical example, from a PowerBook (FireWire), is

```
      /pci@f2000000/mac-io@17/ata-4@1f000/@0:9,ofwboot.xcf
```

Note that colon (':') delimits the device to the left, and comma (',') separates the boot loader filename from the first part. For Open Firmware versions before 3, the primary bootloader is installed in partition “zero”, and it is not necessary to specify the bootloader-filename. For Open Firmware version 3, you must specify the bootloader-filename.

Open Firmware stores aliases to common devices in NVRAM. In the example above, `/pci@f2000000/mac-io@17/ata-4@1f000/@0` is the path on a PowerBook (FireWire) to the built-in ATA/100 hard drive. Use the **devalias** command in Open Firmware to print out a list of common device names on a particular model. The command above could then be simplified to:

```
      hd:9,ofwboot.xcf
```

boot-loader-file-name is usually **ofwboot.xcf**. (See also the **FILES** section for further discussion.)

If omitted, the Open Firmware variable *boot-device* is used.

boot-file

It may be necessary to specify the *boot-file* if Open Firmware does not know where to find the kernel. The default is to load the file named **netbsd** on partition “a” from the device used to load the primary bootloader.

For systems with Open Firmware versions less than 3 which are set up using **sysinst**, the *boot-file* argument is not necessary. Systems with Open Firmware version 3 may need to specify the *boot-file*.

The syntax is similar to the `boot-device` argument:

```
     [boot-file-device:partition-num/][kernel-name]
```

This is a little different, since a kernel-name may be specified without listing a boot-file-device and partition-num. Additionally, a boot-file-device and partition-num may need to be specified, while using the default kernel-name.

If no kernel-name is specified, the primary bootloader will try to find kernels named either *netbsd* or *netbsd.gz* on the boot-device or (if specified) boot-file-device.

options

Possible options are:

−a	ask for the boot device
−s	single-user mode boot
−d	debug mode
exit	exit to Open Firmware after processing arguments

ENVIRONMENT

If set, the following Open Firmware variables will be used to determine which *boot-device* and *boot-file* Open Firmware should use when booting a system. If the user specifies arguments on the command line, these values are overridden.

boot-device	used as the first argument
boot-file	used as the second argument
auto-boot?	setting this variable to *false* will present the user with an Open Firmware command prompt after power-on reset. A value of *true* will automatically boot the system using the variables *boot-device* and *boot-file*. (This is not really related to the boot command, but is included for completeness.)

To restore these variables to their default values, use the **set-default** Open Firmware command: **set-default** *boot-device*

FILES

The three files **ofwboot**, **ofwboot.elf**, and **ofwboot.xcf** are the same program, in different executable formats.

ofwboot	**ofwboot** is installed via `installboot`(8) on systems with Open Firmware versions less than 3. It is not necessary to specify this file name, as it is stored in a special location on the disk, partition "zero". For example, the following command might be used to boot from a SCSI device with ID 2: **0 >boot scsi-int/sd@2:0**.
ofwboot.xcf	**ofwboot.xcf** is in XCOFF format. This file is used on all Open Firmware 3 systems, and on Open Firmware systems prior to 3 when the bootloader is not installed in partition "zero", such as from an ISO-9660 format CD-ROM.
ofwboot.elf	**ofwboot.elf** is in `elf`(5) format and only functions on systems with Open Firmware version 3. To avoid confusion, all users should be using **ofwboot.xcf**, as **ofwboot.elf** offers no additional functionality. It is only included for historical reasons.
boot.fs	This 1.44 MB disk image contains everything necessary to boot and install NetBSD. It includes the partition "zero" bootloader (**ofwboot**), an INSTALL kernel (with limited device drivers), and the **sysinst** utility in a RAM disk. Since Open Firmware does not care what media files are loaded from, only whether they are supported and in the correct format, this disk image may be placed on media other than floppy disks, such as hard drives or Zip disks. Use `dd`(1) on Unix, or **DiskCopy** on MacOS 9.1 or later, or **suntar** on any MacOS version to copy this image onto the media.

netbsd	production kernel, using the GENERIC set of devices which supports almost all hardware available for this platform.
netbsd_GENERIC_MD.gz	GENERIC kernel (the same as *netbsd*), with RAM disk and **sysinst** included.
macppccd.iso	bootable CDROM image for all supported systems. Usually located at `ftp://ftp.NetBSD.org/pub/NetBSD/iso/{RELEASE}/macppccd.iso`

EXAMPLES

Boot an Open Firmware 3 system, with *netbsd* installed on partition "a":

```
0 > boot hd:,ofwboot.xcf
```

Boot into single user mode:

```
0 > boot hd:,ofwboot.xcf netbsd -s
```

Boot from bootable CDROM with Open Firmware 3 or higher:

```
0 > boot cd:,\ofwboot.xcf netbsd.macppc
```

Boot from bootable CDROM (internal SCSI, id=3) of NetBSD 1.5 release with Open Firmware versions prior to 3:

```
0 > boot scsi/sd@3:0,OFWBOOT.XCF NETBSD.MACPPC
```

Boot from floppy disk:

```
0 > boot fd:0
```

Boot from network, with bootps, `bootptab`(5), `tftpd`(8), and `nfsd`(8) server available:

```
0 > boot enet:0
```

Boot from network, but use internal root partition of second drive:

```
0 > boot enet:0 ultra1:0
```

Boot MacOS, looking for the first available bootable disk:

```
0 > boot hd:,\\:tbxi
```

Boot MacOS X residing on partition 10:

```
0 > boot hd:10,\\:tbxi
```

ERRORS

```
DEFAULT CATCH!, code=FF00300 at %SRR0: FF80AD38 %SRR1: 00001070
```
Could be "device not found" or I/O errors on the device. The numbers are just for example.

```
Can't LOAD from this device
```
Open Firmware found the device, but it is not supported by **load**.

```
0 > boot yy:0/netbsd
RESETing to change Configuration!
```
yy:0 doesn't exist, so Open Firmware ignores the string and uses the default parameters to boot MacOS; the MacOS boot routine then clears some of the Open Firmware variables.

```
0 > boot ata/ata-disk@0:9 specified partition is not bootable
 ok
```
As it says.

```
0 > boot ata/ata-disk@0:0
>> NetBSD/macppc OpenFirmware Boot, Revision 1.3
>> (root@nazuha, Fri Jun  8 22:21:55 JST 2001)
no active package3337696/
```
and hangs: See the real-base part in the FAQ.

SEE ALSO

`installboot`(8)

`INSTALL.html`
`http://www.NetBSD.org/ports/macppc/faq.html`
`http://www.NetBSD.org/docs/network/netboot/`

STANDARDS

IEEE Std 1275-1994 ("Open Firmware")

BUGS

`ofwboot` can only boot from devices recognized by Open Firmware.

Early PowerMacintosh systems (particularly the 7500) seem to have problems with netbooting. Adding an arp entry at the tftp server with

```
arp -s booting-host-name its-ethernet-address
```
may resolve this problem (see `arp`(8)).

```
0 > boot CLAIM failed
 ok
```
Once boot failed, successive boots may not be possible. You need to type **`reset-all`** or power-cycle to initialize Open Firmware.

NAME

oqmgr − old Postfix queue manager

SYNOPSIS

oqmgr [generic Postfix daemon options]

DESCRIPTION

The **oqmgr**(8) daemon awaits the arrival of incoming mail and arranges for its delivery via Postfix delivery processes. The actual mail routing strategy is delegated to the **trivial-rewrite**(8) daemon. This program expects to be run from the **master**(8) process manager.

Mail addressed to the local **double-bounce** address is logged and discarded. This stops potential loops caused by undeliverable bounce notifications.

MAIL QUEUES

The **oqmgr**(8) daemon maintains the following queues:

incoming
: Inbound mail from the network, or mail picked up by the local **pickup**(8) agent from the **maildrop** directory.

active
: Messages that the queue manager has opened for delivery. Only a limited number of messages is allowed to enter the **active** queue (leaky bucket strategy, for a fixed delivery rate).

deferred
: Mail that could not be delivered upon the first attempt. The queue manager implements exponential backoff by doubling the time between delivery attempts.

corrupt
: Unreadable or damaged queue files are moved here for inspection.

hold
: Messages that are kept "on hold" are kept here until someone sets them free.

DELIVERY STATUS REPORTS

The **oqmgr**(8) daemon keeps an eye on per-message delivery status reports in the following directories. Each status report file has the same name as the corresponding message file:

bounce
: Per-recipient status information about why mail is bounced. These files are maintained by the **bounce**(8) daemon.

defer
: Per-recipient status information about why mail is delayed. These files are maintained by the **defer**(8) daemon.

trace
: Per-recipient status information as requested with the Postfix "**sendmail -v**" or "**sendmail -bv**" command. These files are maintained by the **trace**(8) daemon.

The **oqmgr**(8) daemon is responsible for asking the **bounce**(8), **defer**(8) or **trace**(8) daemons to send delivery reports.

STRATEGIES

The queue manager implements a variety of strategies for either opening queue files (input) or for message delivery (output).

leaky bucket
: This strategy limits the number of messages in the **active** queue and prevents the queue manager from running out of memory under heavy load.

fairness
: When the **active** queue has room, the queue manager takes one message from the **incoming** queue and one from the **deferred** queue. This prevents a large mail backlog from blocking the delivery of new mail.

slow start
: This strategy eliminates "thundering herd" problems by slowly adjusting the number of parallel deliveries to the same destination.

round robin
The queue manager sorts delivery requests by destination. Round-robin selection prevents one destination from dominating deliveries to other destinations.

exponential backoff
Mail that cannot be delivered upon the first attempt is deferred. The time interval between delivery attempts is doubled after each attempt.

destination status cache
The queue manager avoids unnecessary delivery attempts by maintaining a short-term, in-memory list of unreachable destinations.

TRIGGERS

On an idle system, the queue manager waits for the arrival of trigger events, or it waits for a timer to go off. A trigger is a one-byte message. Depending on the message received, the queue manager performs one of the following actions (the message is followed by the symbolic constant used internally by the software):

D (QMGR_REQ_SCAN_DEFERRED)
Start a deferred queue scan. If a deferred queue scan is already in progress, that scan will be restarted as soon as it finishes.

I (QMGR_REQ_SCAN_INCOMING)
Start an incoming queue scan. If an incoming queue scan is already in progress, that scan will be restarted as soon as it finishes.

A (QMGR_REQ_SCAN_ALL)
Ignore deferred queue file time stamps. The request affects the next deferred queue scan.

F (QMGR_REQ_FLUSH_DEAD)
Purge all information about dead transports and destinations.

W (TRIGGER_REQ_WAKEUP)
Wakeup call, This is used by the master server to instantiate servers that should not go away forever. The action is to start an incoming queue scan.

The **oqmgr**(8) daemon reads an entire buffer worth of triggers. Multiple identical trigger requests are collapsed into one, and trigger requests are sorted so that **A** and **F** precede **D** and **I**. Thus, in order to force a deferred queue run, one would request **A F D**; in order to notify the queue manager of the arrival of new mail one would request **I**.

STANDARDS

RFC 3463 (Enhanced status codes)
RFC 3464 (Delivery status notifications)

SECURITY

The **oqmgr**(8) daemon is not security sensitive. It reads single-character messages from untrusted local users, and thus may be susceptible to denial of service attacks. The **oqmgr**(8) daemon does not talk to the outside world, and it can be run at fixed low privilege in a chrooted environment.

DIAGNOSTICS

Problems and transactions are logged to the **syslog**(8) daemon. Corrupted message files are saved to the **corrupt** queue for further inspection.

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces and of other trouble.

BUGS

A single queue manager process has to compete for disk access with multiple front-end processes such as **cleanup**(8). A sudden burst of inbound mail can negatively impact outbound delivery rates.

CONFIGURATION PARAMETERS

Changes to **main.cf** are not picked up automatically, as **oqmgr**(8) is a persistent process. Use the command "**postfix reload**" after a configuration change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

In the text below, *transport* is the first field in a **master.cf** entry.

COMPATIBILITY CONTROLS

allow_min_user (no)
Allow a recipient address to have '-' as the first character.

ACTIVE QUEUE CONTROLS

qmgr_clog_warn_time (300s)
The minimal delay between warnings that a specific destination is clogging up the Postfix active queue.

qmgr_message_active_limit (20000)
The maximal number of messages in the active queue.

qmgr_message_recipient_limit (20000)
The maximal number of recipients held in memory by the Postfix queue manager, and the maximal size of the size of the short-term, in-memory "dead" destination status cache.

DELIVERY CONCURRENCY CONTROLS

qmgr_fudge_factor (100)
Obsolete feature: the percentage of delivery resources that a busy mail system will use up for delivery of a large mailing list message.

initial_destination_concurrency (5)
The initial per-destination concurrency level for parallel delivery to the same destination.

default_destination_concurrency_limit (20)
The default maximal number of parallel deliveries to the same destination.

*transport***_destination_concurrency_limit**
Idem, for delivery via the named message *transport*.

RECIPIENT SCHEDULING CONTROLS

default_destination_recipient_limit (50)
The default maximal number of recipients per message delivery.

*transport***_destination_recipient_limit**
Idem, for delivery via the named message *transport*.

OTHER RESOURCE AND RATE CONTROLS

minimal_backoff_time (version dependent)
The minimal time between attempts to deliver a deferred message.

maximal_backoff_time (4000s)
The maximal time between attempts to deliver a deferred message.

maximal_queue_lifetime (5d)
The maximal time a message is queued before it is sent back as undeliverable.

queue_run_delay (version dependent)
The time between deferred queue scans by the queue manager.

transport_retry_time (60s)
The time between attempts by the Postfix queue manager to contact a malfunctioning message delivery transport.

Available in Postfix version 2.1 and later:

bounce_queue_lifetime (5d)
The maximal time a bounce message is queued before it is considered undeliverable.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

defer_transports (empty)
The names of message delivery transports that should not deliver mail unless someone issues "**sendmail -q**" or equivalent.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

helpful_warnings (yes)
Log warnings about problematic configuration settings, and provide helpful suggestions.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

FILES

/var/spool/postfix/incoming, incoming queue
/var/spool/postfix/active, active queue
/var/spool/postfix/deferred, deferred queue
/var/spool/postfix/bounce, non-delivery status
/var/spool/postfix/defer, non-delivery status
/var/spool/postfix/trace, delivery status

SEE ALSO

trivial-rewrite(8), address routing
bounce(8), delivery status reports
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
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

NAME

pac — printer/plotter accounting information

SYNOPSIS

pac [**−cmrs**] [**−P** *printer*] [**−p** *price*] [*name ...*]

DESCRIPTION

pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars.

Options and operands available:

−P*printer*
Accounting is done for the named printer. Normally, accounting is done for the default printer (site dependent) or the value of the environment variable PRINTER is used.

−c flag causes the output to be sorted by cost; usually the output is sorted alphabetically by name.

−m flag causes the host name to be ignored in the accounting file. This allows for a user on multiple machines to have all of his printing charges grouped together.

−p*price* The value *price* is used for the cost in dollars instead of the default value of 0.02 or the price specified in /etc/printcap.

−r Reverse the sorting order.

−s Accounting information is summarized on the summary accounting file; this summarization is necessary since on a busy system, the accounting file can grow by several lines per day.

names Statistics are only printed for user(s) *name*; usually, statistics are printed for every user who has used any paper.

OUTPUT FORMAT

pac formats the output into simple table, using four columns - number of feets or pages (column "pages/feet"), how many copies were made (column "runs"), total price for this print (column "price") and user login with host name (column "login" or "host name and login"). If argument *name* was not used and hence **pac** is printing information for all users, a summary line with print totals (runs, pages, price) is appended.

Note that **pac** on other system might print the price as price per copy.

FILES

/var/account/?acct	raw accounting files
/var/account/?_sum	summary accounting files
/etc/printcap	printer capability data base

SEE ALSO

printcap(5)

HISTORY

The **pac** command appeared in 4.0BSD.

BUGS

The relationship between the computed price and reality is as yet unknown.

NAME

pam — Pluggable Authentication Modules framework

DESCRIPTION

The Pluggable Authentication Modules (PAM) framework is a system of libraries that perform authentication tasks for services and applications. Applications that use the PAM API may have their authentication behavior configured by the system administrator though the use of the service's PAM configuration file.

PAM modules provide four classes of functionality:

account	Account verification services such as password expiration and access control.
auth	Authentication services. This usually takes the form of a challenge-response conversation. However, PAM can also support, with appropriate hardware support, biometric devices, smartcards, and so forth.
password	Password (or, more generally, authentication token) change and update services.
session	Session management services. These are tasks that are performed before access to a service is granted and after access to a service is withdrawn. These may include updating activity logs or setting up and tearing down credential forwarding agents.

A primary feature of PAM is the notion of "stacking" different modules together to form a processing chain for the task. This allows fairly precise control over how a particular authentication task is performed, and under what conditions. PAM module configurations may also inherit stacks from other module configurations, providing some degree of centralized administration.

SEE ALSO

`login`(1), `passwd`(1), `su`(1), `pam`(3), `pam.conf`(5), `pam_chroot`(8), `pam_deny`(8), `pam_echo`(8), `pam_exec`(8), `pam_ftpusers`(8), `pam_group`(8), `pam_guest`(8), `pam_krb5`(8), `pam_ksu`(8), `pam_lastlog`(8), `pam_login_access`(8), `pam_nologin`(8), `pam_permit`(8), `pam_radius`(8), `pam_rhosts`(8), `pam_rootok`(8), `pam_securetty`(8), `pam_self`(8), `pam_skey`(8), `pam_ssh`(8), `pam_unix`(8)

HISTORY

The Pluggable Authentication Module framework was originally developed by SunSoft, described in DCE/OSF-RFC 86.0, and first deployed in Solaris 2.6. It was later incorporated into the X/Open Single Sign-On Service (XSSO) Pluggable Authentication Modules specifiation.

The Pluggable Authentication Module framework first appeared in NetBSD 3.0.

NAME

pam_afslog — AFS credentials PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_afslog [*arguments*]

DESCRIPTION

The **pam_afslog** authentication service module for PAM provides functionality for only one PAM category: authentication (*module-type* of "auth").

The **pam_sm_authenticate**() function does nothing and thus the module should be used with an *control-flag* of "optional".

The value of the module comes from its **pam_sm_setcred**() function. If the *afslog* parameter is enabled in krb5.conf(5), then the module will take Kerberos 5 credentials from the cache created by pam_krb5(8) and convert them into AFS tokens, after first creating a PAG (Process Authentication Group) if necessary.

SEE ALSO

kafs(3), pam.conf(5), pam(8), pam_krb5(8)

HISTORY

The **pam_afslog** module was developed for NetBSD by Tyler C. Sarna. The **pam_afslog** module appeared in NetBSD 4.0.

NAME

pam_chroot — Chroot PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_chroot [*arguments*]

DESCRIPTION

The chroot service module for PAM chroots users into either a predetermined directory or one derived from their home directory. If a user's home directory as specified in the *passwd* structure returned by getpwnam(3) contains the string "/./", the portion of the directory name to the left of that string is used as the chroot directory, and the portion to the right will be the current working directory inside the chroot tree. Otherwise, the directories specified by the **dir** and **cwd** options (see below) are used.

also_root Do not hold user ID 0 exempt from the chroot requirement.

always Report a failure if a chroot directory could not be derived from the user's home directory, and the **dir** option was not specified.

cwd=*directory*
Specify the directory to chdir(2) into after a successful chroot(2) call.

dir=*directory*
Specify the chroot directory to use if one could not be derived from the user's home directory.

SEE ALSO

pam.conf(5), pam(8)

AUTHORS

The **pam_chroot** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_deny — Deny PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_deny [*options*]

DESCRIPTION

The deny authentication service module for PAM provides functionality for all the PAM categories: authentication, account management, session management and password management. In terms of the *module-type* parameter, these are the "auth", "account", "session", and "password" features.

The Deny module will universally deny all requests. It is primarily of use during testing, and to "null-out" unwanted functionality.

The following options may be passed to the module:

debug syslog(3) debugging information at LOG_DEBUG level.

no_warn suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

SEE ALSO

syslog(3), pam.conf(5), pam(8)

NAME

pam_echo — Echo PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_echo [*arguments*]

DESCRIPTION

The echo service module for PAM displays its arguments to the user, separated by spaces, using the current conversation function.

If the **%** character occurs anywhere in the arguments to **pam_echo**, it is assumed to introduce one of the following escape sequences:

%H The name of the host on which the client runs (PAM_RHOST).

%s The current service name (PAM_SERVICE).

%t The name of the controlling tty (PAM_TTY).

%U The applicant's user name (PAM_RUSER).

%u The target account's user name (PAM_USER).

Any other two-character sequence beginning with **%** expands to the character following the **%** character.

SEE ALSO

pam.conf(5), pam(8)

AUTHORS

The **pam_echo** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_exec — Exec PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_exec [*arguments*]

DESCRIPTION

The exec service module for PAM executes the program designated by its first argument, with its remaining arguments as command-line arguments. The child's environment is set to the current PAM environment list, as returned by pam_getenvlist(3). In addition, the following PAM items are exported as environment variables: PAM_RHOST, PAM_RUSER, PAM_SERVICE, PAM_TTY, and PAM_USER.

SEE ALSO

pam_get_item(3), pam.conf(5), pam(8)

AUTHORS

The **pam_exec** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_ftpusers — ftpusers PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_ftpusers [*options*]

DESCRIPTION

The ftpusers service module for PAM provides functionality for only one PAM category: account management. In terms of the *module-type* parameter, this is the "account" feature.

Ftpusers Account Management Module

The ftpusers account management component (**pam_sm_acct_mgmt**()), succeeds if and only if the user is listed in /etc/ftpusers.

The following options may be passed to the authentication module:

debug syslog(3) debugging information at LOG_DEBUG level.

no_warn suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

disallow reverse the semantics; **pam_ftpusers** will succeed if and only if the user is not listed in /etc/ftpusers.

SEE ALSO

ftpusers(5), pam.conf(5), ftpd(8), pam(8)

AUTHORS

The **pam_ftpusers** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

BUGS

The current version of this module parses an older format of the ftpusers(5) file and should not be used. ftpd(8) will keep using its built-in ftpusers(5) parsing code until the parser code in the pam module is fixed.

NAME

pam_group — Group PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_group [*arguments*]

DESCRIPTION

The group service module for PAM accepts or rejects users based on their membership in a particular file group.

The following options may be passed to the **pam_group** module:

deny Reverse the meaning of the test, i.e., reject the applicant if and only if he or she is a member of the specified group. This can be useful to exclude certain groups of users from certain services.

fail_safe If the specified group does not exist, or has no members, act as if it does exist and the applicant is a member.

group=*groupname*
Specify the name of the group to check. The default is “wheel”.

root_only Skip this module entirely if the target account is not the superuser account.

authenticate
The user is asked to authenticate using his own password.

SEE ALSO

pam.conf(5), pam(8)

AUTHORS

The **pam_group** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 (“CBOSS”), as part of the DARPA CHATS research program.

NAME

pam_guest — Guest PAM module

SYNOPSIS

[*service-name*] *module-type control-flag* pam_guest [*arguments*]

DESCRIPTION

The guest service module for PAM allows guest logins. If successful, the **pam_guest** module sets the PAM environment variable `GUEST` to the login name. The application can check this variable using `pam_getenv`(3) to differentiate guest logins from normal logins.

The following options may be passed to the **pam_guest** module:

guests=*list*	Comma-separated list of guest account names. The default is "`guest`". A typical value for `ftpd`(8) would be "`anonymous,ftp`".
nopass	Omits the password prompt if the target account is on the list of guest accounts.
pass_as_ruser	The password typed in by the user is exported as the `PAM_RUSER` item. This is useful for applications like `ftpd`(8) where guest users are encouraged to use their email address as password.
pass_is_user	Requires the guest user to type in the guest account name as password.

SEE ALSO

`pam_get_item`(3), `pam_getenv`(3), `pam.conf`(5), `pam`(8)

AUTHORS

The **pam_guest** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_krb5 — Kerberos 5 PAM module

SYNOPSIS

[*service-name*] *module-type control-flag* pam_krb5 [*arguments*]

DESCRIPTION

The Kerberos 5 service module for PAM provides functionality for three PAM categories: authentication, account management, and password management. It also provides null functions for session management.

Kerberos 5 Authentication Module

The Kerberos 5 authentication component provides functions to verify the identity of a user (**pam_sm_authenticate**()) and to set user specific credentials (**pam_sm_setcred**()). **pam_sm_authenticate**() converts the supplied username into a Kerberos principal, by appending the default local realm name. It also supports usernames with explicit realm names. If a realm name is supplied, then upon a successful return, it changes the username by mapping the principal name into a local username (calling **krb5_aname_to_localname**()). This typically just means the realm name is stripped.

It prompts the user for a password and obtains a new Kerberos TGT for the principal. The TGT is verified by obtaining a service ticket for the local host.

When prompting for the current password, the authentication module will use the prompt "`Password for <principal>:`".

The **pam_sm_setcred**() function stores the newly acquired credentials in a credentials cache, and sets the environment variable KRB5CCNAME appropriately. The credentials cache should be destroyed by the user at logout with kdestroy(1).

The following options may be passed to the authentication module:

debug syslog(3) debugging information at LOG_DEBUG level.

no_warn suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

use_first_pass If the authentication module is not the first in the stack, and a previous module obtained the user's password, that password is used to authenticate the user. If this fails, the authentication module returns failure without prompting the user for a password. This option has no effect if the authentication module is the first in the stack, or if no previous modules obtained the user's password.

try_first_pass This option is similar to the **use_first_pass** option, except that if the previously obtained password fails, the user is prompted for another password.

renewable=*timeperiod*
Obtain renewable Kerberos credentials for the user. The renewable time can be specified, or it defaults to one month. Since spaces are not allowed in the pam configuration time, underscores are used to form parseable times (e.g., 1_month).

forwardable Obtain forwardable Kerberos credentials for the user.

no_ccache Do not save the obtained credentials in a credentials cache. This is a useful option if the authentication module is used for services such as ftp or pop, where the user would not be able to destroy them. [This is not a recommendation to use the module for those services.]

ccache=*name*	Use *name* as the credentials cache. *name* must be in the form *type*:*residual*. The special tokens '%u', to designate the decimal UID of the user; and '%p', to designate the current process ID; can be used in *name*.

Kerberos 5 Account Management Module

The Kerberos 5 account management component provides a function to perform account management, **pam_sm_acct_mgmt**(). The function verifies that the authenticated principal is allowed to login to the local user account by calling **krb5_kuserok**() (which checks the user's .k5login file).

Kerberos 5 Password Management Module

The Kerberos 5 password management component provides a function to change passwords (**pam_sm_chauthtok**()). The username supplied (the user running the passwd(1) command, or the username given as an argument) is mapped into a Kerberos principal name, using the same technique as in the authentication module. Note that if a realm name was explicitly supplied during authentication, but not during a password change, the mapping done by the password management module may not result in the same principal as was used for authentication.

Unlike when changing a UNIX password, the password management module will allow any user to change any principal's password (if the user knows the principal's old password, of course). Also unlike UNIX, root is always prompted for the principal's old password.

The password management module uses the same heuristics as kpasswd(1) to determine how to contact the Kerberos password server.

The following options may be passed to the password management module:

debug	syslog(3) debugging information at LOG_DEBUG level.
use_first_pass	If the password management module is not the first in the stack, and a previous module obtained the user's old password, that password is used to authenticate the user. If this fails, the password management module returns failure without prompting the user for the old password. If successful, the new password entered to the previous module is also used as the new Kerberos password. If the new password fails, the password management module returns failure without prompting the user for a new password.
try_first_pass	This option is similar to the **use_first_pass** option, except that if the previously obtained old or new passwords fail, the user is prompted for them.

Kerberos 5 Session Management Module

The Kerberos 5 session management component provides functions to initiate (**pam_sm_open_session**()) and terminate (**pam_sm_close_session**()) sessions. Since session management is not defined under Kerberos 5, both of these functions simply return success. They are provided only because of the naming conventions for PAM modules.

ENVIRONMENT

KRB5CCNAME Location of the credentials cache.

FILES

/tmp/krb5cc_*uid*	default credentials cache (*uid* is the decimal UID of the user).
$HOME/.k5login	file containing Kerberos principals that are allowed access.

SEE ALSO

kdestroy(1), passwd(1), syslog(3), pam.conf(5), pam(8)

NOTES

Applications should not call **pam_authenticate**() more than once between calls to **pam_start**() and **pam_end**() when using the Kerberos 5 PAM module.

SECURITY CONSIDERATIONS

The **pam_krb5** module implements what is fundamentally a password authentication scheme. It does not use a Kerberos 5 exchange between client and server, but rather authenticates the password provided by the client against the Kerberos KDC. Therefore, care should be taken to only use this module over a secure session (secure TTY, encrypted session, etc.) , otherwise the user's Kerberos 5 password could be compromised.

NAME

pam_ksu — Kerberos 5 SU PAM module

SYNOPSIS

[*service-name*] *module-type control-flag* pam_ksu [*options*]

DESCRIPTION

The Kerberos 5 SU authentication service module for PAM provides functionality for only one PAM category: authentication. In terms of the *module-type* parameter, this is the "auth" feature. The module is specifically designed to be used with the su(1) utility.

Kerberos 5 SU Authentication Module

The Kerberos 5 SU authentication component provides functions to verify the identity of a user (**pam_sm_authenticate**()), and determine whether or not the user is authorized to obtain the privileges of the target account. If the target account is "root", then the Kerberos 5 principal used for authentication and authorization will be the "root" instance of the current user, e.g. "user/root@REAL.M". Otherwise, the principal will simply be the current user's default principal, e.g. "user@REAL.M".

The user is prompted for a password if necessary. Authorization is performed by comparing the Kerberos 5 principal with those listed in the .k5login file in the target account's home directory (e.g. /root/.k5login for root).

The following options may be passed to the authentication module:

debug	syslog(3) debugging information at LOG_DEBUG level.
use_first_pass	If the authentication module is not the first in the stack, and a previous module obtained the user's password, that password is used to authenticate the user. If this fails, the authentication module returns failure without prompting the user for a password. This option has no effect if the authentication module is the first in the stack, or if no previous modules obtained the user's password.
try_first_pass	This option is similar to the **use_first_pass** option, except that if the previously obtained password fails, the user is prompted for another password.

SEE ALSO

su(1), syslog(3), pam.conf(5), pam(8)

NAME

pam_lastlog — login accounting PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_lastlog [*options*]

DESCRIPTION

The login accounting service module for PAM provides functionality for only one PAM category: session management. In terms of the *module-type* parameter, this is the "session" feature.

Login Accounting Session Management Module

The login accounting session management component provides functions to initiate (**pam_sm_open_session**()) and terminate (**pam_sm_close_session**()) sessions. The **pam_sm_open_session**() function records the session in the utmp(5), utmpx(5), wtmp(5), wtmpx(5), lastlog(5), and lastlogx(5) databases. The **pam_sm_close_session**() function does nothing.

The following options may be passed to the authentication module:

debug	syslog(3) debugging information at LOG_DEBUG level.
no_nested	Don't update records or print messages if the user is "nested", i.e. logged in on the same tty on top of another user.
no_warn	suppress warning messages to the user.
no_fail	Ignore I/O failures.

SEE ALSO

last(1), w(1), login(3), loginx(3), logout(3), logoutx(3), pam.conf(5), utmp(5), utmpx(5), lastlogin(8), pam(8)

AUTHORS

The **pam_lastlog** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_login_access — login.access PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_login_access [*options*]

DESCRIPTION

The login.access service module for PAM provides functionality for only one PAM category: account management. In terms of the *module-type* parameter, this is the "account" feature.

Login.access Account Management Module

The login.access account management component (**pam_sm_acct_mgmt**()), returns success if and only the user is allowed to log in on the specified tty (in the case of a local login) or from the specified remote host (in the case of a remote login), according to the restrictions listed in /etc/login.access.

SEE ALSO

login.access(5), pam.conf(5), pam(8)

AUTHORS

The login.access(5) access control scheme was designed and implemented by Wietse Venema.

The **pam_login_access** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_nologin — NoLogin PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_nologin [*options*]

DESCRIPTION

The NoLogin authentication service module for PAM provides functionality for only one PAM category: authentication. In terms of the *module-type* parameter, this is the "auth" feature. It also provides a null function for session management.

NoLogin Authentication Module

The NoLogin authentication component (**pam_sm_authenticate**()), always returns success for the superuser, and returns success for all other users if the file /etc/nologin does not exist. If /etc/nologin does exist, then its contents are echoed to non-superusers before failure is returned. If a "nologin" capability is specified in login.conf(5), then the file thus specified is used instead. This usually defaults to /etc/nologin.

The following options may be passed to the authentication module:

debug syslog(3) debugging information at LOG_DEBUG level.

no_warn suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

SEE ALSO

syslog(3), login.conf(5), pam.conf(5), nologin(8), pam(8)

NAME

`pam_permit` — Promiscuous PAM module

SYNOPSIS

`[`*`service-name`*`]` *`module-type`* *`control-flag`* `pam_permit` `[`*`options`*`]`

DESCRIPTION

The Promiscuous authentication service module for PAM provides functionality for all the PAM categories: authentication, account management, session management and password management. In terms of the *`module-type`* parameter, these are the "`auth`", "`account`", "`session`", and "`password`" features.

The Promiscuous module will universally allow all requests. It is primarily of use during testing, and to silence "noisy" PAM-enabled applications.

The following options may be passed to the module:

`debug` `syslog`(3) debugging information at `LOG_DEBUG` level.

SEE ALSO

`syslog`(3), `pam.conf`(5), `pam`(8)

NAME

pam_radius — RADIUS PAM module

SYNOPSIS

[*service-name*] *module-type control-flag* pam_radius [*options*]

DESCRIPTION

The RADIUS service module for PAM provides authentication services based upon the RADIUS (Remote Authentication Dial In User Service) protocol.

The **pam_radius** module accepts these optional parameters:

use_first_pass
: causes **pam_radius** to use a previously entered password instead of prompting for a new one. If no password has been entered then authentication fails.

try_first_pass
: causes **pam_radius** to use a previously entered password, if one is available. If no password has been entered, **pam_radius** prompts for one as usual.

echo_pass
: causes echoing to be left on if **pam_radius** prompts for a password.

conf=*pathname*
: specifies a non-standard location for the RADIUS client configuration file (normally located in /etc/radius.conf).

nas_id=*identifier*
: specifies a NAS identifier to send instead of the hostname.

template_user=*username*
: specifies a user whose passwd(5) entry will be used as a template to create the session environment if the supplied username does not exist in the local password database. The user will be authenticated with the supplied username and password, but his credentials to the system will be presented as the ones for *username*, i.e., his login class, home directory, resource limits, etc. will be set to ones defined for *username*.

  If this option is omitted, and there is no username in the system databases equal to the supplied one (as determined by call to getpwnam(3)), the authentication will fail.

nas_ipaddr[=*address*]
: specifies a NAS IP address to be sent. If option is present, but there is no value provided then IP address corresponding to the current hostname will be used.

FILES

/etc/radius.conf The standard RADIUS client configuration file for **pam_radius**

SEE ALSO

passwd(5), radius.conf(5), pam(8)

HISTORY

The **pam_radius** module first appeared in FreeBSD 3.1. The **pam_radius** manual page first appeared in FreeBSD 3.3.

AUTHORS

The **pam_radius** manual page was written by Andrzej Bialecki ⟨abial@FreeBSD.org⟩.

The **pam_radius** module was written by John D. Polstra ⟨jdp@FreeBSD.org⟩.

NAME

pam_rhosts — rhosts PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_rhosts [*options*]

DESCRIPTION

The rhosts authentication service module for PAM provides functionality for only one PAM category: authentication. In terms of the *module-type* parameter, this is the "auth" feature.

Rhosts Authentication Module

The Rhosts authentication component (**pam_sm_authenticate**()), returns success if and only if the target user's UID is not 0 and the remote host and user are listed in /etc/hosts.equiv or in the target user's ~/.rhosts.

The following options may be passed to the authentication module:

debug	syslog(3) debugging information at LOG_DEBUG level.
no_warn	suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.
allow_root	do not automatically fail if the target user's UID is 0.

SEE ALSO

hosts.equiv(5), pam.conf(5), pam(8)

AUTHORS

The **pam_rhosts** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

NAME

pam_rootok — RootOK PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_rootok [*options*]

DESCRIPTION

The RootOK authentication service module for PAM provides functionality for only one PAM category: authentication. In terms of the *module-type* parameter, this is the "auth" feature. It also provides a null function for session management.

RootOK Authentication Module

The RootOK authentication component (**pam_sm_authenticate**()), always returns success for the superuser; i.e., if getuid(2) returns 0.

The following options may be passed to the authentication module:

debug	syslog(3) debugging information at LOG_DEBUG level.
no_warn	suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

SEE ALSO

getuid(2), pam.conf(5), pam(8)

NAME

pam_securetty — SecureTTY PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_securetty [*options*]

DESCRIPTION

The SecureTTY service module for PAM provides functionality for only one PAM category: account management. In terms of the *module-type* parameter, this is the "account" feature. It also provides null functions for authentication and session management.

SecureTTY Account Management Module

The SecureTTY account management component (**pam_sm_acct_mgmt**()), returns failure if the user is attempting to authenticate as superuser, and the process is attached to an insecure TTY. In all other cases, the module returns success.

A TTY is considered secure if it is listed in /etc/ttys and has the TTY_SECURE flag set.

The following options may be passed to the authentication module:

debug syslog(3) debugging information at LOG_DEBUG level.

no_warn suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

SEE ALSO

getttynam(3), syslog(3), pam.conf(5), ttys(5), pam(8)

NAME

pam_self — Self PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_self [*options*]

DESCRIPTION

The Self authentication service module for PAM provides functionality for only one PAM category: authentication. In terms of the *module-type* parameter, this is the “auth” feature.

Self Authentication Module

The Self authentication component (**pam_sm_authenticate**()), returns success if and only if the target user's user ID is identical with the current real user ID. If the current real user ID is zero, authentication will fail, unless the **allow_root** option was specified.

The following options may be passed to the authentication module:

debug	syslog(3) debugging information at LOG_DEBUG level.
no_warn	suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.
allow_root	do not automatically fail if the current real user ID is 0.

SEE ALSO

getuid(2), pam.conf(5), pam(8)

AUTHORS

The **pam_self** module and this manual page were developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 (“CBOSS”), as part of the DARPA CHATS research program.

NAME

pam_skey — S/Key PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_skey [*options*]

DESCRIPTION

The *S/Key* service module for PAM provides authentication services based on the *S/Key* One Time Password (OTP) authentication system.

The **pam_skey** module has no optional parameters.

FILES

`/etc/skeykeys` database of information for the S/Key system.

SEE ALSO

skey(1), skeyinit(1), pam(8)

NAME

pam_ssh — authentication and session management with SSH private keys

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_ssh [*options*]

DESCRIPTION

The SSH authentication service module for PAM provides functionality for two PAM categories: authentication and session management. In terms of the *module-type* parameter, they are the "auth" and "session" features.

SSH Authentication Module

The SSH authentication component provides a function to verify the identity of a user (**pam_sm_authenticate**()), by prompting the user for a passphrase and verifying that it can decrypt the target user's SSH key using that passphrase.

The following options may be passed to the authentication module:

use_first_pass If the authentication module is not the first in the stack, and a previous module obtained the user's password, that password is used to authenticate the user. If this fails, the authentication module returns failure without prompting the user for a password. This option has no effect if the authentication module is the first in the stack, or if no previous modules obtained the user's password.

try_first_pass This option is similar to the **use_first_pass** option, except that if the previously obtained password fails, the user is prompted for another password.

SSH Session Management Module

The SSH session management component provides functions to initiate (**pam_sm_open_session**()) and terminate (**pam_sm_close_session**()) sessions. The **pam_sm_open_session**() function starts an SSH agent, passing it any private keys it decrypted during the authentication phase, and sets the environment variables the agent specifies. The **pam_sm_close_session**() function kills the previously started SSH agent by sending it a SIGTERM.

The following options may be passed to the session management module:

want_agent Start an agent even if no keys were decrypted during the authentication phase.

FILES

$HOME/.ssh/identity	SSH1 RSA key
$HOME/.ssh/id_rsa	SSH2 RSA key
$HOME/.ssh/id_dsa	SSH2 DSA key

SEE ALSO

ssh-agent(1), pam.conf(5), pam(8)

AUTHORS

The **pam_ssh** module was originally written by Andrew J. Korty ⟨ajk@iu.edu⟩. The current implementation was developed for the FreeBSD Project by ThinkSec AS and NAI Labs, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program. This manual page was written by Mark R V Murray ⟨markm@FreeBSD.org⟩.

SECURITY CONSIDERATIONS

The **pam_ssh** module implements what is fundamentally a password authentication scheme. Care should be taken to only use this module over a secure session (secure TTY, encrypted session, etc.), otherwise the user's SSH passphrase could be compromised.

Additional consideration should be given to the use of **pam_ssh**. Users often assume that file permissions are sufficient to protect their SSH keys, and thus use weak or no passphrases. Since the system administrator has no effective means of enforcing SSH passphrase quality, this has the potential to expose the system to security risks.

NAME

pam_unix — UNIX PAM module

SYNOPSIS

[*service-name*] *module-type* *control-flag* pam_unix [*options*]

DESCRIPTION

The UNIX authentication service module for PAM provides functionality for two PAM categories: authentication and account management. In terms of the *module-type* parameter, they are the "auth" and "account" features. It also provides a null function for session management.

UNIX **Authentication Module**

The UNIX authentication component provides functions to verify the identity of a user (**pam_sm_authenticate**()), which obtains the relevant passwd(5) entry. It prompts the user for a password and verifies that this is correct with crypt(3).

The following options may be passed to the authentication module:

debug	syslog(3) debugging information at LOG_DEBUG level.
use_first_pass	If the authentication module is not the first in the stack, and a previous module obtained the user's password, that password is used to authenticate the user. If this fails, the authentication module returns failure without prompting the user for a password. This option has no effect if the authentication module is the first in the stack, or if no previous modules obtained the user's password.
try_first_pass	This option is similar to the **use_first_pass** option, except that if the previously obtained password fails, the user is prompted for another password.
auth_as_self	This option will require the user to authenticate himself as the user given by getlogin(2), not as the account they are attempting to access. This is primarily for services like su(1), where the user's ability to retype their own password might be deemed sufficient.
nullok	If the password database has no password for the entity being authenticated, then this option will forgo password prompting, and silently allow authentication to succeed.

UNIX **Account Management Module**

The UNIX account management component provides a function to perform account management, **pam_sm_acct_mgmt**(). The function verifies that the authenticated user is allowed to login to the local user account by checking the password expiry date.

The following options may be passed to the management module:

debug	syslog(3) debugging information at LOG_DEBUG level.

UNIX **Password Management Module**

The UNIX password management component provides a function to perform account management, **pam_sm_chauthtok**(). The function changes the user's password.

The following options may be passed to the password module:

debug	syslog(3) debugging information at LOG_DEBUG level.
no_warn	suppress warning messages to the user. These messages include reasons why the user's authentication attempt was declined.

passwd_db=*name* Change the user's password only the specified password database. Valid password database names are:

files	local password file
nis	NIS password database

FILES

`/etc/master.passwd` default UNIX password database.

SEE ALSO

passwd(1), getlogin(2), crypt(3), getpwent(3), syslog(3), nsswitch.conf(5), passwd(5), nis(8), pam(8)

NAME

paxctl — list and modify PaX flags associated with an ELF program

SYNOPSIS

paxctl *flags program ...*

DESCRIPTION

The **paxctl** utility is used to list and manipulate PaX flags associated with an ELF program.

Each flag can be prefixed either with a "+" or a "-" sign to add or remove the flag, respectively.

The following flags are available:

a	Explicitly disable PaX ASLR for *program*.
A	Explicitly enable PaX ASLR for *program*.
g	Explicitly disable PaX Segvguard for *program*.
G	Explicitly enable PaX Segvguard for *program*.
m	Explicitly disable PaX MPROTECT (mprotect(2) restrictions) for *program*.
M	Explicitly enable PaX MPROTECT (mprotect(2) restrictions) for *program*.

To view existing flags on a file, execute **paxctl** without any flags.

SEE ALSO

sysctl(3), options(4), security(8), sysctl(8)

HISTORY

The **paxctl** utility first appeared in NetBSD 4.0.

The **paxctl** utility is modeled after a tool of the same name available for Linux from the PaX project.

AUTHORS

Elad Efrat ⟨elad@NetBSD.org⟩
Christos Zoulas ⟨christos@NetBSD.org⟩

BUGS

The **paxctl** utility currently uses elf(5) "note" sections to mark executables as PaX Segvguard enabled. This will be done using fileassoc(9) in the future so that we can control who does the marking.

NAME

pbsdboot — load and boot NetBSD/hpcmips kernel from Windows CE

SYNOPSIS

pbsdboot.exe

DESCRIPTION

pbsdboot is a program runs on Windows CE. It loads and executes the specified NetBSD/netbsd kernel.

The menu options (for **pbsdboot** itself) are as follows:

kernel	Select Kernel Path.
Frame buffer	Select Frame Buffer type.
Option	option for pass kernel.

The options for NetBSD kernel are as follows:

-d break into the kernel debugger.

-m use miniroot in memory.

-s single user mode.

-h use serial console.check also serial port on.and connect your terminal with 9600bps,8bit,non-parity,VT100 mode.

-a ask for name:kernel ask root/dump device,filesystem.

-b=DEV
change boot device to DEV(wd0,sd0,nfs,etc)

FILES

`/msdos/pbsdboot.exe` You will find this program Windows CE readable disk partition.

SEE ALSO

`reboot`(2)

HISTORY

The **pbsdboot** utility first appeared in NetBSD 1.5.

BUGS

pbsdboot reads the entire kernel image at once, and requires enough free area on the main memory.

NAME

pcictl — a program to manipulate the PCI bus

SYNOPSIS

pcictl *pcibus command* [*arg* [...]]

DESCRIPTION

pcictl allows a user or system administrator to access various resources on a PCI bus.

The following commands are available:

list [**-n**] [**-b** *bus*] [**-d** *device*] [**-f** *function*]

List the devices in the PCI domain, either as names or, if **-n** is given, as numbers. The bus, device, and function numbers may be specified by flags. If the bus is not specified, it defaults to the bus number of the PCI bus specified on the command line. Any other locator not specified defaults to a wildcard, or may be explicitly wildcarded by specifying "any".

dump [**-b** *bus*] **-d** *device* [**-f** *function*]

Dump the PCI configuration space for the specified device located at the specified bus, device, and function. If the bus is not specified, it defaults to the bus number of the PCI bus specified on the command line. If the function is not specified, it defaults to 0.

FILES

`/dev/pci*` - PCI bus device nodes

SEE ALSO

pci(3), pci(4), drvctl(8)

HISTORY

The **pcictl** command first appeared in NetBSD 1.6.

NAME

pcifs — refuse-based virtual file system to display devices on PCI bus

SYNOPSIS

pcifs [**-v**] [**-b** *bus*] *mount_point*

DESCRIPTION

The **pcifs** utility can be used to mount a virtual file system which under the mount point, showing the layout of devices on the PCI bus.

The following arguments can be used:

-b *bus*
Use the numeric bus number from the argument for the PCI bus number to list.

-v Produce verbose output

The **pcifs** utility makes use of the virtdir(3) virtual directory routines.

The refuse(3) library is used to provide the file system features.

The mandatory parameter is the local mount point.

The pcictl(8) utility is used to retrieve the information on PCI devices.

SEE ALSO

puffs(3), refuse(3), virtdir(3), pcictl(8)

HISTORY

The **pcifs** utility first appeared in NetBSD 5.0.

AUTHORS

Alistair Crooks ⟨agc@NetBSD.org⟩

NAME

pcnfsd, rpc.pcnfsd − (PC)NFS authentication and print request server

SYNOPSIS

/usr/sbin/rpc.pcnfsd

DESCRIPTION

pcnfsd is an RPC server that supports Sun ONC clients on PC (DOS, OS/2, Macintosh, and other) systems. This page describes version two of the **pcnfsd** server.

rpc.pcnfsd may be started from **/etc/rc.local** or by the **inetd**(8) superdaemon. It reads the configuration file **/etc/pcnfsd.conf** if present, and then services RPC requests directed to program number 150001. This release of the **pcnfsd** daemon supports both version 1 and version 2 of the pcnfsd protocol. Consult the **rpcgen** source file **pcnfsd.x** for details of the protocols.

The requests serviced by **pcnfsd** fall into three categories: authentication, printing, and other. Only the authentication and printing services have administrative significance.

AUTHENTICATION

When **pcnfsd** receives a **PCNFSD_AUTH** or **PCNFSD2_AUTH** request, it will "log in" the user by validating the username and password and returning the corresponding uid, gids, home directory, and umask. If **pcnfsd** was built with the **WTMP** compile-time option, it will also append a record to the **wtmp**(5) data base. If you do not wish to record PC "logins" in this way, you should add a line of the form

wtmp off

to the **/etc/pcnfsd.conf** file.

PRINTING

pcnfsd supports a printing model based on the use of NFS to transfer the actual print data from the client to the server. The client system issues a **PCNFSD_PR_INIT** or **PCNFSD2_PR_INIT** request, and the server returns the path to a spool directory which the client may use and which is exported by NFS. **pcnfsd** creates a subdirectory for each of its clients: the parent directory is normally **/export/pcnfs** and the subdirectory is the hostname of the client system. If you wish to use a different parent directory, you should add a line of the form

spooldir *path*

to the **/etc/pcnfsd.conf** file.

Once a client has mounted the spool directory using NFS and has transferred print data to a file in this directory, it will issue a **PCNFSD_PR_START** or **PCNFSD2_PR_START** request. **pcnfsd** handles this, and most other print-related requests, by constructing a command based on the printing services of the server operating system and executing the command using the identity of the PC user. Since this involves set-user-id privileges, **pcnfsd** must be run as root.

Every print request from the client includes the name of the printer which is to be used. In SunOS, this name corresponds to a printer definition in the **/etc/printcap**(5) database. If you wish to define a non-standard way of processing print data, you should define a new printer and arrange for the client to print to this printer. There are two ways of setting up a new printer. The first involves the addition of an entry to **/etc/printcap**(5) and the creation of filters to perform the required processing. This is outside the scope of this discussion. In addition, **pcnfsd** includes a mechanism by which you can define virtual printers known only to **pcnfsd** clients. Each printer is defined by a line in the **/etc/pcnfsd.conf** file of the following form

printer *name alias-for command*

name is the name of the printer you want to define. *alias-for* is the name of a "real" printer which corresponds to this printer. For example, a request to display the queue for *name* will be translated into the corresponding request for the printer *alias-for*. If you have defined a printer in such a way that there is no "real" printer to which it corresponds, use a single "-" for this field. (See the definition of the printer **test** below for an example.) *command* is a command which will be executed whenever a file is printed on *name*. This command is executed by the Bourne shell, **/bin/sh** using the **-c** option. For complex operations you should construct an executable shell program and invoke that in *command*. Within *command* the following tokens will be replaced:

Token	**Substitution**
$FILE	Replaced by the full path name of the print data file. When the command has been executed, the file will be unlinked.
$USER	Replaced by the username of the user logged in to the client system.
$HOST	Replaced by the host name of the client system.

Consider the following example **/etc/pcnfsd.conf** file:

```
printer rotated lw /usr/local/bin/enscript -2r $FILE
printer test - /usr/bin/cp $FILE /usr/tmp/$HOST-$USER
```

If a client system prints a job on the printer **rotated** the utility **enscript** will be invoked to pre-process the file $FILE. In this case, the **-2r** option causes the file to be printed in two-column rotated format on the default PostScript® printer. If the client requests a list of the print queue for the printer **rotated** the **pcnfsd** daemon will translate this into a request for a listing for the printer **lw.**

The printer **test** is used only for testing. Any file sent to this printer will be copied into **/usr/tmp.** Any request to list the queue, check the status, etc. of printer **test** will be rejected because the *alias-for* has been specified as "-".

FILES

/etc/pcnfsd.conf	configuration file
/export/pcnfs	default print spool directory

SEE ALSO

lpr(1), **lpc**(8)

HISTORY

The **pcnfsd** source code is distributed by Sun Microsystems, Inc. with their PC/NFS product under terms described in common.h in that source code. Those terms require that you be informed that this version of **pcnfsd** was modified to run on NetBSD and is NOT supported by Sun.

NAME

pdisk — Apple partition table editor

SYNOPSIS

pdisk [**−acdfhilLrv**] [**−-abbr**] [**−-compute_size**] [**−-debug**] [**−-fname**] [**−-help**] [**−-interactive**] [**−-list** *device*] [**−-logical**] [**−-readonly**] [**−-version**] [*device ...*]

DESCRIPTION

pdisk is a menu driven program which partitions disks using the standard Apple disk partitioning scheme described in "Inside Macintosh: Devices". It does not support the Intel/DOS partitioning scheme supported by fdisk(8).

Supported options are:

−a	
−-abbr	Abbreviate the partition types shown in the partition list.
−c	
−-compute_size	Causes **pdisk** to always ignore the device size listed in the partition table and compute the device size by other means.
−d	
−-debug	Turns on debugging. Doesn't add that much output, but does add a new command 'x' to the editing commands that accesses an eclectic bunch of undocumented functionality.
−f	
−-fname	Show HFS volume names instead of partition name when available.
−h	
−-help	Prints a short help message.
−i	
−-interactive	Causes **pdisk** to go into an interactive mode similar to the MacOS version of the program.
−l	
−-list *device*	List the partition tables for the specified *devices*.
−L	
−-logical	Show partition limits in logical blocks. Default is physical blocks.
−r	
−-readonly	Prevents **pdisk** from writing to the device.
−v	
−-version	Prints the version number of **pdisk**.

Editing Partition Tables

An argument which is simply the name of a *device* indicates that **pdisk** should edit the partition table of that device.

The current top level editing commands are:

C	(create with type also specified)
c	create new partition
d	delete a partition
h	command help
i	initialize partition map
n	(re)name a partition
P	(print ordered by base address)
p	print the partition table

q	quit editing (don't save changes)
r	reorder partition entry in map
s	change size of partition map
t	change the type of an existing partition
w	write the partition table

Commands which take arguments prompt for each argument in turn. You can also type any number of the arguments separated by spaces and those prompts will be skipped. The only exception to typeahead are the confirmation prompts on the **i** and **w** commands, since if we expect you to confirm the decision, we shouldn't undermine that by allowing you to be precipitate about it.

Partitions are always specified by their number, which is the index of the partition entry in the partition map. Most of the commands will change the index numbers of all partitions after the affected partition. You are advised to print the table as frequently as necessary.

The **c** (create new partition) command is the only one with complicated arguments. The first argument is the base address (in blocks) of the partition. Besides a raw number, you can also specify a partition number followed by the letter ‘p’ to indicate that the first block of the new partition should be the same as the first block of that existing free space partition. The second argument is the length of the partition in blocks. This can be a raw number or can be a partition number followed by the letter ‘p’ to use the size of that partition or can be a number followed by ‘k’, ‘m’, or ‘g’ to indicate the size in kilobytes, megabytes, or gigabytes respectively. (These are powers of 1024, of course, not powers of 1000.) The third argument is the name of the partition. This can be a single word without quotes, or a string surrounded by single or double quotes. The type of the created partition will be Apple_UNIX_SVR2, which is the correct type for use with NetBSD. This command will prompt for the unix filesystem slice to set in the Block Zero Block bits.

The **C** command is similar to the **c** command, with the addition of a partition type argument after the other arguments. Choosing a type of Apple_UNIX_SVR2 will prompt for the unix filesystem slice to set in the Block Zero Block bits.

The **i** (initalize) command prompts for the size of the device.

The **n** (name) command allows the name of a partition to be changed. Note that the various "Apple_Driver" partitions depend on the name field for proper functioning. We are not aware of any other partition types with this limitation.

The **r** (reorder) command allows the index number of partitions to be changed. The index numbers are constrained to be a contiguous sequence.

The **t** (change partition type) command allows the type of a partition to be changed. Changing the type to Apple_UNIX_SVR2 will prompt for the unix filesystem slice to set in the Block Zero Block bits.

The **w** (write) command writes the partition map out. In order to use the new partition map you must reboot.

SEE ALSO

`fdisk`(8), `newfs`(8)

HISTORY

The **pdisk** utility was originally developed for MkLinux.

AUTHORS

Eryk Vershen

BUGS

Some people believe there should really be just one disk partitioning utility.

Filesystem volume names are out of place in a partition utility. This utility supports HFS volume names, but not volume names of any other filesystem types.

The **--logical** option has not been heavily tested.

pdisk will first try to use `lseek`(2) with `SEEK_END` to compute the size of the device. If this fails, it will try a binary search using `lseek`(2) and `read`(2) to find the end of the device. This has been observed to fail on some raw disk devices. As a workaround, try using the block device instead. **pdisk** should probably read the disklabel using the `DIOCGDINFO` `ioctl`(2) to get the device size instead.

NAME

pfctl — control the packet filter (PF) and network address translation (NAT) device

SYNOPSIS

pfctl [**−AdeghmNnOoqRrvz**] [**−a** *anchor*] [**−D** *macro*=*value*] [**−F** *modifier*] [**−f** *file*] [**−i** *interface*] [**−k** *host*] [**−p** *device*] [**−s** *modifier*] [**−t** *table* **−T** *command* [*address ...*]] [**−x** *level*]

DESCRIPTION

The **pfctl** utility communicates with the packet filter device using the ioctl interface described in pf(4). It allows ruleset and parameter configuration and retrieval of status information from the packet filter.

Packet filtering restricts the types of packets that pass through network interfaces entering or leaving the host based on filter rules as described in pf.conf(5). The packet filter can also replace addresses and ports of packets. Replacing source addresses and ports of outgoing packets is called NAT (Network Address Translation) and is used to connect an internal network (usually reserved address space) to an external one (the Internet) by making all connections to external hosts appear to come from the gateway. Replacing destination addresses and ports of incoming packets is used to redirect connections to different hosts and/or ports. A combination of both translations, bidirectional NAT, is also supported. Translation rules are described in pf.conf(5).

When the variable *pf* is set to YES in rc.conf(5), the rule file specified with the variable *pf_rules* is loaded automatically by the rc(8) scripts and the packet filter is enabled.

The packet filter does not itself forward packets between interfaces. Forwarding can be enabled by setting the sysctl(8) variables *net.inet.ip.forwarding* and/or *net.inet6.ip6.forwarding* to 1. Set them permanently in /etc/sysctl.conf.

The **pfctl** utility provides several commands. The options are as follows:

−A Load only the queue rules present in the rule file. Other rules and options are ignored.

−a *anchor*

Apply flags **−f**, **−F**, and **−s** only to the rules in the specified *anchor*. In addition to the main ruleset, **pfctl** can load and manipulate additional rulesets by name, called anchors. The main ruleset is the default anchor.

Anchors are referenced by name and may be nested, with the various components of the anchor path separated by '/' characters, similar to how file system hierarchies are laid out. The last component of the anchor path is where ruleset operations are performed.

Evaluation of *anchor* rules from the main ruleset is described in pf.conf(5).

For example, the following will show all filter rules (see the **−s** flag below) inside the anchor authpf/smith(1234), which would have been created for user smith by authpf(8), PID 1234:

```
# pfctl -a "authpf/smith(1234)" -s rules
```

Private tables can also be put inside anchors, either by having table statements in the pf.conf(5) file that is loaded in the anchor, or by using regular table commands, as in:

```
# pfctl -a foo/bar -t mytable -T add 1.2.3.4 5.6.7.8
```

When a rule referring to a table is loaded in an anchor, the rule will use the private table if one is defined, and then fall back to the table defined in the main ruleset, if there is one. This is similar to C rules for variable scope. It is possible to create distinct tables with the same name in the global ruleset and in an anchor, but this is often bad design and a warning will be issued in that case.

−D *macro*=*value*
Define *macro* to be set to *value* on the command line. Overrides the definition of *macro* in the ruleset.

−d Disable the packet filter.

−e Enable the packet filter.

−F *modifier*
Flush the filter parameters specified by *modifier* (may be abbreviated):

−F nat	Flush the NAT rules.
−F queue	Flush the queue rules.
−F rules	Flush the filter rules.
−F state	Flush the state table (NAT and filter).
−F Sources	Flush the source tracking table.
−F info	Flush the filter information (statistics that are not bound to rules).
−F Tables	Flush the tables.
−F osfp	Flush the passive operating system fingerprints.
−F all	Flush all of the above.

−f *file*
Load the rules contained in *file*. This *file* may contain macros, tables, options, and normalization, queueing, translation, and filtering rules. With the exception of macros and tables, the statements must appear in that order.

−g Include output helpful for debugging.

−h Help.

−i *interface*
Restrict the operation to the given *interface*.

−k *host*
Kill all of the state entries originating from the specified *host*. A second **−k** *host* option may be specified, which will kill all the state entries from the first *host* to the second *host*. For example, to kill all of the state entries originating from host:

```
# pfctl -k host
```

To kill all of the state entries from host1 to host2:

```
# pfctl -k host1 -k host2
```

−m Merge in explicitly given options without resetting those which are omitted. Allows single options to be modified without disturbing the others:

```
# echo "set loginterface fxp0" | pfctl -mf -
```

−N Load only the NAT rules present in the rule file. Other rules and options are ignored.

−n Do not actually load rules, just parse them.

−O Load only the options present in the rule file. Other rules and options are ignored.

−o Enable the ruleset optimizer. The ruleset optimizer attempts to improve rulesets by removing rule duplication and making better use of rule ordering. Specifically, it does four things:

1. remove duplicate rules

2. remove rules that are a subset of another rule
3. combine multiple rules into a table when advantageous
4. re-order the rules to improve evaluation performance

A second **−o** may be specified to use the currently loaded ruleset as a feedback profile to tailor the optimization of the *quick* rules to the actual network behavior.

It is important to note that the ruleset optimizer will modify the ruleset to improve performance. A side effect of the ruleset modification is that per-rule accounting statistics will have different meanings than before. If per-rule accounting is important for billing purposes or whatnot, either the ruleset optimizer should not be used or a *label* field should be added to all of the accounting rules to act as optimization barriers.

−p *device*
Use the device file *device* instead of the default `/dev/pf`.

−q Only print errors and warnings.

−R Load only the filter rules present in the rule file. Other rules and options are ignored.

−r Perform reverse DNS lookups on states when displaying them.

−s *modifier*
Show the filter parameters specified by *modifier* (may be abbreviated):

−s nat	Show the currently loaded NAT rules.
−s queue	Show the currently loaded queue rules. When used together with **−v**, per-queue statistics are also shown. When used together with **−v −v**, **pfctl** will loop and show updated queue statistics every five seconds, including measured bandwidth and packets per second.
−s rules	Show the currently loaded filter rules. When used together with **−v**, the per-rule statistics (number of evaluations, packets and bytes) are also shown. Note that the “skip step” optimization done automatically by the kernel will skip evaluation of rules where possible. Packets passed statefully are counted in the rule that created the state (even though the rule isn't evaluated more than once for the entire connection).
−s Anchors	Show the currently loaded anchors directly attached to the main ruleset. If **−a** *anchor* is specified as well, the anchors loaded directly below the given *anchor* are shown instead. If **−v** is specified, all anchors attached under the target anchor will be displayed recursively.
−s state	Show the contents of the state table.
−s Sources	Show the contents of the source tracking table.
−s info	Show filter information (statistics and counters). When used together with **−v**, source tracking statistics are also shown.
−s labels	Show per-rule statistics (label, evaluations, packets, bytes) of filter rules with labels, useful for accounting.
−s timeouts	Show the current global timeouts.
−s memory	Show the current pool memory hard limits.
−s Tables	Show the list of tables.
−s osfp	Show the list of operating system fingerprints.
−s Interfaces	Show the list of interfaces and interface drivers available to PF. When used together with a double **−v**, interface statistics are also shown. **−i** can be used to select an interface or a group of interfaces.

−s all	Show all of the above, except for the lists of interfaces and operating system fingerprints.

−T *command* [*address ...*]

Specify the *command* (may be abbreviated) to apply to the table. Commands include:

−T kill	Kill a table.
−T flush	Flush all addresses of a table.
−T add	Add one or more addresses in a table. Automatically create a nonexisting table.
−T delete	Delete one or more addresses from a table.
−T replace	Replace the addresses of the table. Automatically create a nonexisting table.
−T show	Show the content (addresses) of a table.
−T test	Test if the given addresses match a table.
−T zero	Clear all the statistics of a table.
−T load	Load only the table definitions from pf.conf(5). This is used in conjunction with the **−f** flag, as in:

```
# pfctl -Tl -f pf.conf
```

For the **add**, **delete**, **replace**, and **test** commands, the list of addresses can be specified either directly on the command line and/or in an unformatted text file, using the **−f** flag. Comments starting with a '#' are allowed in the text file. With these commands, the **−v** flag can also be used once or twice, in which case **pfctl** will print the detailed result of the operation for each individual address, prefixed by one of the following letters:

A	The address/network has been added.
C	The address/network has been changed (negated).
D	The address/network has been deleted.
M	The address matches (**test** operation only).
X	The address/network is duplicated and therefore ignored.
Y	The address/network cannot be added/deleted due to conflicting '!' attributes.
Z	The address/network has been cleared (statistics).

Each table maintains a set of counters that can be retrieved using the **−v** flag of **pfctl**. For example, the following commands define a wide open firewall which will keep track of packets going to or coming from the OpenBSD FTP server. The following commands configure the firewall and send 10 pings to the FTP server:

```
# printf "table <test> { ftp.NetBSD.org }\n \
    pass out to <test> keep state\n" | pfctl -f-
# ping -qc10 ftp.NetBSD.org
```

We can now use the table **show** command to output, for each address and packet direction, the number of packets and bytes that are being passed or blocked by rules referencing the table. The time at which the current accounting started is also shown with the "Cleared" line.

```
# pfctl -t test -vTshow
   129.128.5.191
    Cleared:     Thu Feb 13 18:55:18 2003
    In/Block:    [ Packets: 0        Bytes: 0        ]
    In/Pass:     [ Packets: 10       Bytes: 840      ]
    Out/Block:   [ Packets: 0        Bytes: 0        ]
    Out/Pass:    [ Packets: 10       Bytes: 840      ]
```

Similarly, it is possible to view global information about the tables by using the **−v** modifier twice and the **−s Tables** command. This will display the number of addresses on each table, the number of rules which reference the table, and the global packet statistics for the whole table:

```
# pfctl -vvsTables
--a-r-  test
    Addresses:   1
    Cleared:     Thu Feb 13 18:55:18 2003
    References:  [ Anchors: 0        Rules: 1        ]
    Evaluations: [ NoMatch: 3496     Match: 1        ]
    In/Block:    [ Packets: 0        Bytes: 0        ]
    In/Pass:     [ Packets: 10       Bytes: 840      ]
    In/XPass:    [ Packets: 0        Bytes: 0        ]
    Out/Block:   [ Packets: 0        Bytes: 0        ]
    Out/Pass:    [ Packets: 10       Bytes: 840      ]
    Out/XPass:   [ Packets: 0        Bytes: 0        ]
```

As we can see here, only one packet − the initial ping request − matched the table, but all packets passing as the result of the state are correctly accounted for. Reloading the table(s) or ruleset will not affect packet accounting in any way. The two "XPass" counters are incremented instead of the "Pass" counters when a "stateful" packet is passed but doesn't match the table anymore. This will happen in our example if someone flushes the table while the `ping`(8) command is running.

When used with a single **−v**, **pfctl** will only display the first line containing the table flags and name. The flags are defined as follows:

c	For constant tables, which cannot be altered outside `pf.conf`(5).
p	For persistent tables, which don't get automatically killed when no rules refer to them.
a	For tables which are part of the *active* tableset. Tables without this flag do not really exist, cannot contain addresses, and are only listed if the **−g** flag is given.
i	For tables which are part of the *inactive* tableset. This flag can only be witnessed briefly during the loading of `pf.conf`(5).
r	For tables which are referenced (used) by rules.
h	This flag is set when a table in the main ruleset is hidden by one or more tables of the same name from anchors attached below it.

−t *table*
Specify the name of the table.

−v Produce more verbose output. A second use of **−v** will produce even more verbose output including ruleset warnings. See the previous section for its effect on table commands.

−x *level*
Set the debug *level* (may be abbreviated) to one of the following:

−x none	Don't generate debug messages.
−x urgent	Generate debug messages only for serious errors.
−x misc	Generate debug messages for various errors.
−x loud	Generate debug messages for common conditions.

−z Clear per-rule statistics.

FILES

`/etc/pf.conf`	Packet filter rules file.
`/etc/pf.os`	Passive operating system fingerprint database.

SEE ALSO

`pf`(4), `pf.conf`(5), `pf.os`(5), `rc.conf`(5), `authpf`(8), `ftp-proxy`(8), `rc`(8), `sysctl`(8)

HISTORY

The **pfctl** program and the pf(4) filter mechanism first appeared in OpenBSD 3.0.

NAME

pflogd — packet filter logging daemon

SYNOPSIS

pflogd [**-Dx**] [**-d** *delay*] [**-f** *filename*] [**-s** *snaplen*] [*expression*]

DESCRIPTION

pflogd is a background daemon which reads packets logged by pf(4) to the packet logging interface pflog0 and writes the packets to a logfile (normally /var/log/pflog) in tcpdump(8) binary format. These logs can be reviewed later using the **-r** option of tcpdump(8), hopefully offline in case there are bugs in the packet parsing code of tcpdump(8).

pflogd closes and then re-opens the log file when it receives SIGHUP, permitting newsyslog(8) to rotate logfiles automatically. SIGALRM causes **pflogd** to flush the current logfile buffers to the disk, thus making the most recent logs available. The buffers are also flushed every *delay* seconds.

If the log file contains data after a restart or a SIGHUP, new logs are appended to the existing file. If the existing log file was created with a different snaplen, **pflogd** temporarily uses the old snaplen to keep the log file consistent.

pflogd tries to preserve the integrity of the log file against I/O errors. Furthermore, integrity of an existing log file is verified before appending. If there is an invalid log file or an I/O error, logging is suspended until a SIGHUP or a SIGALRM is received.

The options are as follows:

-D Debugging mode. **pflogd** does not disassociate from the controlling terminal.

-d *delay*
Time in seconds to delay between automatic flushes of the file. This may be specified with a value between 5 and 3600 seconds. If not specified, the default is 60 seconds.

-f *filename*
Log output filename. Default is /var/log/pflog.

-s *snaplen*
Analyze at most the first *snaplen* bytes of data from each packet rather than the default of 96. The default of 96 is adequate for IP, ICMP, TCP, and UDP headers but may truncate protocol information for other protocols. Other file parsers may desire a higher snaplen.

-x Check the integrity of an existing log file, and return.

expression
Selects which packets will be dumped, using the regular language of tcpdump(8).

FILES

/var/run/pflogd.pid	Process ID of the currently running **pflogd**.
/var/log/pflog	Default log file.

EXAMPLES

Log specific tcp packets to a different log file with a large snaplen (useful with a log-all rule to dump complete sessions):

```
# pflogd -s 1600 -f suspicious.log port 80 and host evilhost
```

Display binary logs:

```
# tcpdump -n -e -ttt -r /var/log/pflog
```

Display the logs in real time (this does not interfere with the operation of **pflogd**):

```
# tcpdump -n -e -ttt -i pflog0
```

Tcpdump has been extended to be able to filter on the pfloghdr structure defined in ⟨*net/if_pflog.h*⟩. Tcpdump can restrict the output to packets logged on a specified interface, a rule number, a reason, a direction, an IP family or an action.

ip	Address family equals IPv4.
ip6	Address family equals IPv6.
ifname kue0	Interface name equals "kue0".
on kue0	Interface name equals "kue0".
rulenum 10	Rule number equals 10.
reason match	Reason equals match. Also accepts "bad-offset", "fragment", "bad-timestamp", "short", "normalize" and "memory".
action pass	Action equals pass. Also accepts "block".
inbound	The direction was inbound.
outbound	The direction was outbound.

Display the logs in real time of inbound packets that were blocked on the wi0 interface:

```
# tcpdump -n -e -ttt -i pflog0 inbound and action block and on wi0
```

SEE ALSO

pcap(3), pf(4), pflog(4), pf.conf(5), newsyslog(8), tcpdump(8)

HISTORY

The **pflogd** command appeared in OpenBSD 3.0.

AUTHORS

Can Erkin Acar

NAME

pickup − Postfix local mail pickup

SYNOPSIS

pickup [generic Postfix daemon options]

DESCRIPTION

The **pickup**(8) daemon waits for hints that new mail has been dropped into the **maildrop** directory, and feeds it into the **cleanup**(8) daemon. Ill-formatted files are deleted without notifying the originator. This program expects to be run from the **master**(8) process manager.

STANDARDS

None. The **pickup**(8) daemon does not interact with the outside world.

SECURITY

The **pickup**(8) daemon is moderately security sensitive. It runs with fixed low privilege and can run in a chrooted environment. However, the program reads files from potentially hostile users. The **pickup**(8) daemon opens no files for writing, is careful about what files it opens for reading, and does not actually touch any data that is sent to its public service endpoint.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

The **pickup**(8) daemon copies mail from file to the **cleanup**(8) daemon. It could avoid message copying overhead by sending a file descriptor instead of file data, but then the already complex **cleanup**(8) daemon would have to deal with unfiltered user data.

CONFIGURATION PARAMETERS

As the **pickup**(8) daemon is a relatively long-running process, up to an hour may pass before a **main.cf** change takes effect. Use the command "**postfix reload**" command to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

CONTENT INSPECTION CONTROLS

content_filter (empty)
The name of a mail delivery transport that filters mail after it is queued.

receive_override_options (empty)
Enable or disable recipient validation, built-in content filtering, or address mapping.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

line_length_limit (2048)
Upon input, long lines are chopped up into pieces of at most this length; upon delivery, long lines are reconstructed.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

cleanup(8), message canonicalization
sendmail(1), Sendmail-compatible interface
postdrop(1), mail posting agent
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
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

ping — send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

ping [**-adDfLnoPqQrRv**] [**-c** *count*] [**-E** *policy*] [**-g** *gateway*] [**-h** *host*]
[**-i** *interval*] [**-I** *srcaddr*] [**-l** *preload*] [**-p** *pattern*] [**-s** *packetsize*]
[**-t** *tos*] [**-T** *ttl*] [**-w** *deadline*] *host*

DESCRIPTION

ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (''pings'') have an IP and ICMP header, followed by a "struct timeval" and then an arbitrary number of ''pad'' bytes used to fill out the packet. The options are as follows:

-a Emit an audible beep (by sending an ascii BEL character to the standard error output) after each non-duplicate response is received. This is disabled for flood pings as it would probably cause temporary insanity.

-c *count*
Stop after sending (and waiting the specified delay to receive) *count* ECHO_RESPONSE packets.

-d Set the SO_DEBUG option on the socket being used.

-D Set the Don't Fragment bit in the IP header. This can be used to determine the path MTU.

-E *policy*
Use IPsec policy specification string *policy* for packets. For the format of specification string, please refer ipsec_set_policy(3). Please note that this option is same as **-P** in KAME/FreeBSD and KAME/BSDI (as **-P** was already occupied in NetBSD).

-f Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period ''.'' is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. Only the super-user may use this option. *This can be very hard on a network and should be used with caution.*

-g *gateway*
Use Loose Source Routing to send the ECHO_REQUEST packets via *gateway*.

-i *interval*
Wait *interval* seconds *between sending each packet*. The default is to wait for one second between each packet, except when the -f option is used the wait interval is 0.01 seconds.

-I *srcaddr*
Set the source IP address to *srcaddr* which can be a hostname or an IP number. For multicast datagrams, it also specifies the outgoing interface.

-h *host*
is an alternate way of specifying the target host instead of as the last argument.

-l *preload*
If *preload* is specified, **ping** sends that many packets as fast as possible before falling into its normal mode of behavior. Only the super-user may use this option.

-L Disable loopback when sending to multicast destinations, so the transmitting host doesn't see the ICMP requests.

−n Numeric output only. No attempt will be made to look up symbolic names for host addresses.

−o Exit successfully after receiving one reply packet.

−p *pattern*
You may specify up to 16 ‘‘pad’’ bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, “`−p ff`” will cause the sent packet to be filled with all ones.

−P Use a pseudo-random sequence for the data instead of the default, fixed sequence of incrementing 8-bit integers. This is useful to foil compression on PPP and other links.

−q Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

−Q Do not display responses such as Network Unreachable ICMP messages concerning the ECHO_REQUESTs sent.

−r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by `routed`(8)).

−R Record Route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. This should show the path to the target host and back, which is especially useful in the case of asymmetric routing. Note that the IP header is only large enough for nine such addresses, and only seven when using the **−g** option. This is why it was necessary to invent `traceroute`(8). Many hosts ignore or discard this option.

−s *packetsize*
Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data. The maximum allowed value is 65467 bytes.

−T *ttl*
Use the specified time-to-live.

−t *tos*
Use the specified hexadecimal type of service.

−v Verbose output. ICMP packets other than ECHO_RESPONSE that are received are listed.

−w *deadline*
Specifies a timeout, in seconds, before ping exits regardless of how many packets have been sent or received.

When using **ping** for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be ‘‘pinged’’.

Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers.

When the specified number of packets have been sent (and received) or if the program is terminated with a `SIGINT`, a brief summary is displayed. The summary information can be displayed while **ping** is running by sending it a `SIGINFO` signal (see the “status” argument for `stty`(1) for more information).

ping continually sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. On a trusted system with IP Security Options enabled, if the network idiom is not MONO, **ping** also prints a second line containing the hexadecimal representation of the IP security option in the ECHO_RESPONSE. If the **−c** count option is given, only that number of requests is sent. No output is produced if there is no response. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the specified number of packets have been sent (and received) or if the program is terminated with an interrupt (SIGINT), a brief summary is displayed. When not using the **−f** (flood) option, the first interrupt, usually generated by control-C or DEL, causes **ping** to wait for its outstanding requests to return. It will wait no longer than the longest round trip time encountered by previous, successful pings. The second interrupt stops ping immediately.

This program is intended for use in network testing, measurement and management. Because of the load it can impose on the network, it is unwise to use **ping** during normal operations or from automated scripts.

ICMP PACKET DETAILS

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a *packetsize* is given, this indicated the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least eight bytes large, **ping** uses the first eight bytes of this space to include a timestamp to compute round trip times. If less than eight bytes of pad are specified, no round trip times are given.

DUPLICATE AND DAMAGED PACKETS

ping will report duplicate and damaged packets. Duplicate packets should never occur, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the **ping** packet's path (in the network or in the hosts).

TRYING DIFFERENT DATA PATTERNS

The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that doesn't have sufficient ‘‘transitions’’, such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn't necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either can't be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the **−p** option of **ping**.

TTL DETAILS

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one.

The TCP/IP specification states that the TTL field for TCP packets should be set to 60, but many systems use smaller values (4.3BSD uses 30, 4.2BSD used 15) .

The maximum possible value of this field is 255, and most UNIX systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you will find you can ''ping'' some hosts, but not reach them with `telnet`(1) or `ftp`(1).

In normal operation ping prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

- Not change it; this is what Berkeley UNIX systems did before the 4.3BSD−Tahoe release. In this case the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.
- Set it to 255; this is what current Berkeley UNIX systems do. In this case the TTL value in the received packet will be 255 minus the number of routers in the path *from* the remote system *to* the **`ping`***ing* host.
- Set it to some other value. Some machines use the same value for ICMP packets that they use for TCP packets, for example either 30 or 60. Others may use completely wild values.

EXIT STATUS

`ping` returns 0 on success (the host is alive), and non-zero if the arguments are incorrect or the host is not responding.

SEE ALSO

`netstat`(1), `icmp`(4), `inet`(4), `ip`(4), `ifconfig`(8), `routed`(8), `spray`(8), `traceroute`(8)

HISTORY

The **`ping`** command appeared in 4.3BSD. IPsec support was added by WIDE/KAME project.

BUGS

Flood pinging is not recommended in general, and flood pinging a broadcast or multicast address should only be done under very controlled conditions.

The **`ping`** program has evolved differently under different operating systems, and in some cases the same flag performs a different function under different operating systems. The **`−t`** flag conflicts with FreeBSD. The **`−a`**, **`−c`**, **`−i`**, **`−I`**, **`−l`**, **`−p`**, **`−P`**, **`−s`**, and **`−t`** flags conflict with **Solaris**.

Some hosts and gateways ignore the RECORD_ROUTE option.

The maximum IP header length is too small for options like RECORD_ROUTE to be completely useful. There's not much that that can be done about this, however.

NAME

ping6 — send ICMPv6 ECHO_REQUEST packets to network hosts

SYNOPSIS

ping6 [**-dfHmnNqRtvwW**] [**-a** *addrtype*] [**-b** *bufsiz*] [**-c** *count*] [**-g** *gateway*]
[**-h** *hoplimit*] [**-I** *interface*] [**-i** *wait*] [**-l** *preload*] [**-p** *pattern*]
[**-P** *policy*] [**-S** *sourceaddr*] [**-s** *packetsize*] [*hops ...*] *host*

DESCRIPTION

ping6 uses the ICMPv6 protocol's mandatory ICMP6_ECHO_REQUEST datagram to elicit an ICMP6_ECHO_REPLY from a host or gateway. ICMP6_ECHO_REQUEST datagrams (“pings”) have an IPv6 header, and ICMPv6 header formatted as documented in RFC 2463. The options are as follows:

-a *addrtype*
Generate ICMPv6 Node Information Node Addresses query, rather than echo-request. *addrtype* must be a string constructed of the following characters.

a requests unicast addresses from all of the responder's interfaces. If the character is omitted, only those addresses which belong to the interface which has the responder's address are requests.

c requests responder's IPv4-compatible and IPv4-mapped addresses.

g requests responder's global-scope addresses.

s requests responder's site-local addresses.

l requests responder's link-local addresses.

A requests responder's anycast addresses. Without this character, the responder will return unicast addresses only. With this character, the responder will return anycast addresses only. Note that the specification does not specify how to get responder's anycast addresses. This is an experimental option.

-b *bufsiz*
Set socket buffer size.

-c *count*
Stop after sending (and receiving) *count* ECHO_RESPONSE packets.

-d Set the SO_DEBUG option on the socket being used.

-f Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period “.” is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. Only the super-user may use this option. *This can be very hard on a network and should be used with caution.*

-g *gateway*
Specifies to use *gateway* as the next hop to the destination. The gateway must be a neighbor of the sending node.

-H Specifies to try reverse-lookup of IPv6 addresses. The **ping6** command does not try reverse-lookup unless the option is specified.

-h *hoplimit*
Set the IPv6 hoplimit.

-I *interface*
Source packets with the given interface address. This flag applies if the ping destination is a multicast address, or link-local/site-local unicast address.

−i *wait*
: Wait *wait* seconds *between sending each packet*. The default is to wait for one second between each packet. This option is incompatible with the **−f** option.

−l *preload*
: If *preload* is specified, **ping6** sends that many packets as fast as possible before falling into its normal mode of behavior. Only the super-user may use this option.

−m By default, **ping6** asks the kernel to fragment packets to fit into the minimum IPv6 MTU. **−m** will suppress the behavior in the following two levels: when the option is specified once, the behavior will be disabled for unicast packets. When the option is specified more than once, it will be disabled for both unicast and multicast packets.

−n Numeric output only. No attempt will be made to lookup symbolic names from addresses in the reply.

−N Probe node information multicast group (`ff02::2:xxxx:xxxx`). *host* must be string hostname of the target (must not be a numeric IPv6 address). Node information multicast group will be computed based on given *host*, and will be used as the final destination. Since node information multicast group is a link-local multicast group, outgoing interface needs to be specified by **−I** option.

−p *pattern*
: You may specify up to 16 "pad" bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, "`−p ff`" will cause the sent packet to be filled with all ones.

−P *policy*
: *policy* specifies IPsec policy to be used for the probe.

−q Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

−R Make the kernel believe that the target *host* (or the first *hop* if you specify *hops*) is reachable, by injecting upper-layer reachability confirmation hint. The option is meaningful only if the target *host* (or the first hop) is a neighbor.

−S *sourceaddr*
: Specifies the source address of request packets. The source address must be one of the unicast addresses of the sending node, and must be numeric.

−s *packetsize*
: Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data. You may need to specify **−b** as well to extend socket buffer size.

−t Generate ICMPv6 Node Information supported query types query, rather than echo-request. **−s** has no effect if **−t** is specified.

−v Verbose output. ICMP packets other than ECHO_RESPONSE that are received are listed.

−w Generate ICMPv6 Node Information DNS Name query, rather than echo-request. **−s** has no effect if **−w** is specified.

−W Same as **−w**, but with old packet format based on 03 draft. This option is present for backward compatibility. **−s** has no effect if **−w** is specified.

hops IPv6 addresses for intermediate nodes, which will be put into type 0 routing header.

host IPv6 address of the final destination node.

When using **ping6** for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be "pinged". Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the round-trip time statistics. When the specified number of packets have been sent (and received) or if the program is terminated with a SIGINT, a brief summary is displayed, showing the number of packets sent and received, and the minimum, maximum, mean, and standard deviation of the round-trip times.

This program is intended for use in network testing, measurement and management. Because of the load it can impose on the network, it is unwise to use **ping6** during normal operations or from automated scripts.

DUPLICATE AND DAMAGED PACKETS

ping6 will report duplicate and damaged packets. Duplicate packets should never occur when pinging a unicast address, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm. Duplicates are expected when pinging a multicast address, since they are not really duplicates but replies from different hosts to the same request.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the **ping6** packet's path (in the network or in the hosts).

TRYING DIFFERENT DATA PATTERNS

The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that does not have sufficient "transitions", such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It is not necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either cannot be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the **−p** option of **ping6**.

EXIT STATUS

ping6 exits with 0 on success (the host is alive), and non-zero if the arguments are incorrect or the host is not responding.

EXAMPLES

Normally, **ping6** works just like ping(8) would work; the following will send ICMPv6 echo request to dst.foo.com.

```
ping6 -n dst.foo.com
```

The following will probe hostnames for all nodes on the network link attached to wi0 interface. The address ff02::1 is named the link-local all-node multicast address, and the packet would reach every node on the network link.

```
ping6 -w ff02::1%wi0
```

The following will probe addresses assigned to the destination node, dst.foo.com.

```
      ping6 -a agl dst.foo.com
```

SEE ALSO

netstat(1), icmp6(4), inet6(4), ip6(4), ifconfig(8), ping(8), routed(8), traceroute(8), traceroute6(8)

A. Conta and S. Deering, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, RFC 2463, December 1998.

Matt Crawford, *IPv6 Node Information Queries*, draft-ietf-ipngwg-icmp-name-lookups-09.txt, May 2002, work in progress material.

HISTORY

The ping(8) command appeared in 4.3BSD. The **ping6** command with IPv6 support first appeared in the WIDE Hydrangea IPv6 protocol stack kit.

BUGS

ping6 is intentionally separate from ping(8).

NAME

pipe − Postfix delivery to external command

SYNOPSIS

pipe [generic Postfix daemon options] command_attributes...

DESCRIPTION

The **pipe**(8) daemon processes requests from the Postfix queue manager to deliver messages to external commands. This program expects to be run from the **master**(8) process manager.

Message attributes such as sender address, recipient address and next-hop host name can be specified as command-line macros that are expanded before the external command is executed.

The **pipe**(8) daemon updates queue files and marks recipients as finished, or it informs the queue manager that delivery should be tried again at a later time. Delivery status reports are sent to the **bounce**(8), **defer**(8) or **trace**(8) daemon as appropriate.

SINGLE-RECIPIENT DELIVERY

Some external commands cannot handle more than one recipient per delivery request. Examples of such transports are pagers or fax machines.

To prevent Postfix from sending multiple recipients per delivery request, specify

*transport*_**destination_recipient_limit = 1**

in the Postfix **main.cf** file, where *transport* is the name in the first column of the Postfix **master.cf** entry for the pipe-based delivery transport.

COMMAND ATTRIBUTE SYNTAX

The external command attributes are given in the **master.cf** file at the end of a service definition. The syntax is as follows:

chroot=*pathname* (optional)

Change the process root directory and working directory to the named directory. This happens before switching to the privileges specified with the **user** attribute, and before executing the optional **directory=***pathname* directive. Delivery is deferred in case of failure.

This feature is available as of Postfix 2.3.

directory=*pathname* (optional)

Change to the named directory before executing the external command. The directory must be accessible for the user specified with the **user** attribute (see below). The default working directory is **$queue_directory**. Delivery is deferred in case of failure.

This feature is available as of Postfix 2.2.

eol=*string* (optional, default: **\n**)

The output record delimiter. Typically one would use either **\r\n** or **\n**. The usual C-style backslash escape sequences are recognized: **\a \b \f \n \r \t \v** *ddd* (up to three octal digits) and \\.

flags=BDFORhqu.> (optional)

Optional message processing flags. By default, a message is copied unchanged.

B Append a blank line at the end of each message. This is required by some mail user agents that recognize "**From** " lines only when preceded by a blank line.

D Prepend a "**Delivered-To:** *recipient*" message header with the envelope recipient address. Note: for this to work, the *transport*_**destination_recipient_limit** must be 1.

This feature is available as of Postfix 2.0.

F Prepend a "**From** *sender time_stamp*" envelope header to the message content. This is expected by, for example, **UUCP** software.

O Prepend an "**X-Original-To:** *recipient*" message header with the recipient address as given to Postfix. Note: for this to work, the *transport*_**destination_recipient_limit** must be 1.

This feature is available as of Postfix 2.0.

R Prepend a **Return-Path:** message header with the envelope sender address.

h Fold the command-line **$recipient** domain name and **$nexthop** host name to lower case. This is recommended for delivery via **UUCP**.

q Quote white space and other special characters in the command-line **$sender** and **$recipient** address localparts (text to the left of the right-most **@** character), according to an 8-bit transparent version of RFC 822. This is recommended for delivery via **UUCP** or **BSMTP**.

The result is compatible with the address parsing of command-line recipients by the Postfix **sendmail**(1) mail submission command.

The **q** flag affects only entire addresses, not the partial address information from the **$user**, **$extension** or **$mailbox** command-line macros.

u Fold the command-line **$recipient** address localpart (text to the left of the right-most **@** character) to lower case. This is recommended for delivery via **UUCP**.

. Prepend "**.**" to lines starting with "**.**". This is needed by, for example, **BSMTP** software.

> Prepend ">" to lines starting with "**From** ". This is expected by, for example, **UUCP** software.

null_sender=*replacement* (default: MAILER-DAEMON)

Replace the null sender address (typically used for delivery status notifications) with the specified text when expanding the **$sender** command-line macro, and when generating a From_ or Return-Path: message header.

If the null sender replacement text is a non-empty string then it is affected by the **q** flag for address quoting in command-line arguments.

The null sender replacement text may be empty; this form is recommended for content filters that feed mail back into Postfix. The empty sender address is not affected by the **q** flag for address quoting in command-line arguments.

Caution: a null sender address is easily mis-parsed by naive software. For example, when the **pipe**(8) daemon executes a command such as:

```
command -f$sender -- $recipient (bad)
```

the command will mis-parse the -f option value when the sender address is a null string. For correct parsing, specify **$sender** as an argument by itself:

```
command -f $sender -- $recipient (good)
```

This feature is available with Postfix 2.3 and later.

size=*size_limit* (optional)

Messages greater in size than this limit (in bytes) will be returned to the sender as undeliverable.

user=*username* (required)

user=*username*:*groupname*

Execute the external command with the rights of the specified *username*. The software refuses to execute commands with root privileges, or with the privileges of the mail system owner. If *groupname* is specified, the corresponding group ID is used instead of the group ID of *username*.

argv=*command*... (required)

The command to be executed. This must be specified as the last command attribute. The command is executed directly, i.e. without interpretation of shell meta characters by a shell command interpreter.

In the command argument vector, the following macros are recognized and replaced with corresponding information from the Postfix queue manager delivery request.

In addition to the form ${*name*}, the forms $*name* and $(*name*) are also recognized. Specify **$$** where a single **$** is wanted.

${client_address}

This macro expands to the remote client network address.

This is available in Postfix 2.2 and later.

${client_helo}

This macro expands to the remote client HELO command parameter.

This is available in Postfix 2.2 and later.

${client_hostname}

This macro expands to the remote client hostname.

This is available in Postfix 2.2 and later.

${client_protocol}

This macro expands to the remote client protocol.

This is available in Postfix 2.2 and later.

${extension}

This macro expands to the extension part of a recipient address. For example, with an address *user+foo@domain* the extension is *foo*.

A command-line argument that contains **${extension**} expands into as many command-line arguments as there are recipients.

This information is modified by the **u** flag for case folding.

${mailbox}

This macro expands to the complete local part of a recipient address. For example, with an address *user+foo@domain* the mailbox is *user+foo*.

A command-line argument that contains **${mailbox**} expands to as many command-line arguments as there are recipients.

This information is modified by the **u** flag for case folding.

${nexthop}

This macro expands to the next-hop hostname.

This information is modified by the **h** flag for case folding.

${recipient}
This macro expands to the complete recipient address.

A command-line argument that contains **${recipient}** expands to as many command-line arguments as there are recipients.

This information is modified by the **hqu** flags for quoting and case folding.

${sasl_method}
This macro expands to the SASL authentication mechanism used during the reception of the message. An empty string is passed if the message has been received without SASL authentication.

This is available in Postfix 2.2 and later.

${sasl_sender}
This macro expands to the SASL sender name (i.e. the original submitter as per RFC 2554) used during the reception of the message.

This is available in Postfix 2.2 and later.

${sasl_username}
This macro expands to the SASL user name used during the reception of the message. An empty string is passed if the message has been received without SASL authentication.

This is available in Postfix 2.2 and later.

${sender}
This macro expands to the envelope sender address. By default, the null sender address expands to MAILER-DAEMON; this can be changed with the **null_sender** attribute, as described above.

This information is modified by the **q** flag for quoting.

${size} This macro expands to Postfix's idea of the message size, which is an approximation of the size of the message as delivered.

${user}
This macro expands to the username part of a recipient address. For example, with an address *user+foo@domain* the username part is *user*.

A command-line argument that contains **${user}** expands into as many command-line arguments as there are recipients.

This information is modified by the **u** flag for case folding.

STANDARDS

RFC 3463 (Enhanced status codes)

DIAGNOSTICS

Command exit status codes are expected to follow the conventions defined in <**sysexits.h**>. Exit status 0 means normal successful completion.

Postfix version 2.3 and later support RFC 3463-style enhanced status codes. If a command terminates with a non-zero exit status, and the command output begins with an enhanced status code, this status code takes precedence over the non-zero exit status.

Problems and transactions are logged to **syslogd**(8). Corrupted message files are marked so that the queue manager can move them to the **corrupt** queue for further inspection.

SECURITY

This program needs a dual personality 1) to access the private Postfix queue and IPC mechanisms, and 2) to execute external commands as the specified user. It is therefore security sensitive.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **pipe**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

RESOURCE AND RATE CONTROLS

In the text below, *transport* is the first field in a **master.cf** entry.

*transport*__destination_concurrency_limit ($default_destination_concurrency_limit)_**

Limit the number of parallel deliveries to the same destination, for delivery via the named *transport*. The limit is enforced by the Postfix queue manager.

*transport*__destination_recipient_limit ($default_destination_recipient_limit)_**

Limit the number of recipients per message delivery, for delivery via the named *transport*. The limit is enforced by the Postfix queue manager.

*transport*__time_limit ($command_time_limit)_**

Limit the time for delivery to external command, for delivery via the named *transport*. The limit is enforced by the pipe delivery agent.

Postfix 2.4 and later support a suffix that specifies the time unit: s (seconds), m (minutes), h (hours), d (days), w (weeks). The default time unit is seconds.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)

The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)

How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)

The maximal number of digits after the decimal point when logging sub-second delay values.

export_environment (see 'postconf -d' output)

The list of environment variables that a Postfix process will export to non-Postfix processes.

ipc_timeout (3600s)

The time limit for sending or receiving information over an internal communication channel.

mail_owner (postfix)

The UNIX system account that owns the Postfix queue and most Postfix daemon processes.

max_idle (100s)

The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)

The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)

The process ID of a Postfix command or daemon process.

process_name (read-only)

The process name of a Postfix command or daemon process.

queue_directory (see ’postconf -d’ output)
The location of the Postfix top-level queue directory.

recipient_delimiter (empty)
The separator between user names and address extensions (user+foo).

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

qmgr(8), queue manager
bounce(8), delivery status reports
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
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

plainrsa-gen — generator for Plain RSA keys

SYNOPSIS

plainrsa-gen [**-b** *bits*] [**-e** *pubexp*] [**-f** *outfile*] [**-h**]

DESCRIPTION

plainrsa-gen can be used to generate `Plain RSA keys` for authentication purposes. Using `Plain RSA keys` is optional. Other possibilities are `Pre-shared keys` or `X.509 certificates`.

-b *bits*
bit length of the key. Default is `1024`, recommended length is `2048` or even `4096` bits. Note that generating longer keys takes longer time.

-e *pubexp*
value of RSA public exponent. Default is `0x3`. Don't change this unless you really know what you are doing!

-f *outfile*
outfile instead of `stdout`. If the file already exists it won't be overwritten. You wouldn't like to lose your private key by accident, would you?

OUTPUT FILE FORMAT

This is the secret `private key` that should **never** leave your computer:

```
: RSA   {
        # RSA 1024 bits
        # pubkey=0sAQOrWlcwbAIdNSMhDt...
        Modulus: 0xab5a57306c021d3523...
        PublicExponent: 0x03
        PrivateExponent: 0x723c3a2048...
        Prime1: 0xd309b30e6adf9d85c01...
        Prime2: 0xcfdc2a8aa5b2b3c90e3...
        Exponent1: 0x8cb122099c9513ae...
        Exponent2: 0x8a92c7071921cd30...
        Coefficient: 0x722751305eafe9...
  }
```

The line `pubkey=0sAQOrW...` of the `private key` contains a `public key` that should be stored in the other peer's configuration in this format:

```
: PUB 0sAQOrWlcwbAIdNSMhDt...
```

You can also specify `from` and `to` addresses for which the key is valid:

```
0.0.0.0/0 10.20.30.0/24 : PUB 0sAQOrWlcwbAIdNSMhDt...
```

SEE ALSO

`racoon.conf`(5), `racoon`(8)

HISTORY

plainrsa-gen was written by Michal Ludvig ⟨michal@logix.cz⟩ and first appeared in **ipsec-tools 0.4**.

NAME

poffd — x68k shutdown daemon

SYNOPSIS

poffd [*shutdown-program*]

DESCRIPTION

poffd looks at the power switch of x68k. When the switch is turned off, **poffd** executes *shutdown-program* (using `/bin/sh -c`).

If the system is started by x68k's RTC timer, *shutdown-program* is executed when the system's shut down time (predefined using other utility, SX-Window Control Panel and `rtcalarm`(8) for example) arrives.

The argument *shutdown-program* may contain the character '%'. It is substituted with a number which indicates the way how the system was started:

0	Front power switch
1	External (I/O slot) power signal
2	RTC alarm

This is useful when choosing the shutdown message.

If the argument *shutdown-program* is omitted, '**`/sbin/shutdown -r +1`**' is executed instead.

SEE ALSO

`pow`(4), `rtcalarm`(8), `shutdown`(8)

AUTHORS

poffd was written by MINOURA Makoto ⟨minoura@flab.fujitsu.co.jp⟩.

SPECIAL THANKS

Liam Hahne Minn <hahne@sail.t.u-tokyo.ac.jp>.

NAME

popper — POP3 server

SYNOPSIS

popper [**-k**] [**-a** *plaintext*|*otp*|*sasl*] [**-t** *file*] [**-T** *seconds*] [**-d**] [**-i**] [**-p** *port*]
[**--address-log=**file]

DESCRIPTION

popper serves mail via the Post Office Protocol. Supported options include:

-a *plaintext*|*otp*|*sasl*
Tells **popper** which authentication mode is acceptable, *sasl* enables SASL (RFC2222), and *otp* enables OTP (RFC1938) authentication. Both disable plaintext passwords.

--address-log=file
Logs the addresses (along with a timestamp) of all clients to the specified file. This can be used to implement POP-before-SMTP authentication.

-d Enables more verbose log messages.

-i When not started by inetd, this flag tells **popper** that it has to create a socket by itself.

-k Tells **popper** to use Kerberos for authentication. This is the traditional way of doing Kerberos authentication, and is normally done on a separate port (as it doesn't follow RFC1939), and should be used instead of using SASL.

-p *port*
Port to listen to, in combination with **-i**.

-t *file*
Trace all commands to file.

-T *seconds*
Set timeout to something other than the default of 120 seconds.

SEE ALSO

push(8), movemail(8)

STANDARDS

RFC1939 (Post Office Protocol - Version 3)

AUTHORS

The server was initially developed at the University of California, Berkeley.

Many changes have been made as part of the KTH Kerberos distributions.

NAME

postinstall — check and fix installation after system upgrades

SYNOPSIS

postinstall [**-a** *arch*] [**-d** *destdir*] [**-m** *machine*]
[**-s** {*srcdir* | *tgzdir* | *tgzfile*}] *operation* [*item* [...]]

DESCRIPTION

The **postinstall** utility performs post-installation checks and/or fixes on a system's configuration files. It is especially useful after system upgrades, e.g. after updating from NetBSD 1.6.2 to NetBSD 2.0. The items to check or fix are divided in two groups: enabled by default and disabled by default. The latter are items that are dangerous for some reason, for example because they remove files which may be still in use. If no *items* are provided, the default checks or fixes are applied. Those which are disabled by default must be provided explicitly.

Supported options:

-a *arch* MACHINE_ARCH. Defaults to machine of the host operating system.

-d *destdir* Destination directory to check. Defaults to `/`.

-m *machine* MACHINE. Defaults to machine of the host operating system.

-s {*srcdir* | *tgzdir* | *tgzfile*}
The location of the reference files, or the NetBSD source files used to create the reference files. This may be specified in one of three ways:

-s *srcdir* The top level directory of the NetBSD source tree. By default this is `/usr/src`.

-s *tgzdir* A directory in which reference files have been extracted from one or more of the "`etc.tgz`" or "`xetc.tgz`" files from a binary distribution of NetBSD.

-s *tgzfile* The location of one or more set files (or "tgz files") from a binary distribution of NetBSD. Each set file is a compressed archive containing reference files. More than one set file may be specified, either by using a single **-s** option with a colon-separated list of *tgzfile* names, or by using multiple **-s** options; this is likely to be useful to specify the locations of both the `etc.tgz` and `xetc.tgz` set files. Note that file names may not contain embedded colon ('`:`') characters, because that would conflict with the use of the colon character to delimit multiple file names with a single argument.

The *operation* argument may be one of:

check Perform post-installation checks on items.

diff [`diff`(1) options]
Similar to **check**, but also show the differences between the files.

fix Apply fixes that **check** determines need to be applied. Not all items can be automatically fixed by **postinstall**, and in some cases an error will be reported, after which manual intervention will be required.

Conflicts between existing files in the target file system and new files from the NetBSD distribution are resolved by replacing the existing file with the new file; there is no attempt to merge the files. See `etcupdate`(8) for an alternative update method that is able to merge files.

`help` Display a short help.

`list` List available *`items`*, showing if they are enabled or disabled by default.

`usage` Same as **`help`**.

EXIT STATUS

The **`postinstall`** utility exits 0 on success, and >0 if an error occurs or a problem was found.

SEE ALSO

`etcupdate`(8)

HISTORY

The **`postinstall`** utility first appeared in NetBSD 1.6.

NAME

powerd — power management daemon for sysmon

SYNOPSIS

powerd [**-d**]

DESCRIPTION

powerd acts upon power management events posted by the kernel's power management facility. When events are posted, **powerd** translates the event into a script name and a list of arguments. **powerd** then runs the script in order to implement the power management policy defined by the system administrator.

powerd supports the following option:

-d Enable debugging mode. Verbose messages will be sent to stderr and **powerd** will stay in the foreground of the controlling terminal.

CONFIGURATION SCRIPTS

All of **powerd** configuration is encapsulated into scripts that are run when power management events occur. **powerd** will look for these scripts in two locations. The first location is `/etc/powerd/scripts/<power_type>`, where ⟨`power_type`⟩ is defined by the power management mechanism supported by the system, e.g., "apm" or "acpi". If the script is not found in the first location, **powerd** looks in `/etc/powerd/scripts`.

Configuration scripts are run synchronously, i.e., **powerd** will start the script and wait for its completion before it handles the next event.

Configuration scripts are called with different arguments, depending on the script class. These classes are described in the following sections.

POWER SWITCH SCRIPTS

Power switch scripts are called when a state change event occurs on a power switch device. Power switch scripts are called with two arguments: the device with which the device is associated, and the event type.

The following power switch script names are defined:

power_button	This script is called when an event occurs on a power button device.
reset_button	This script is called when an event occurs on a reset button device.
sleep_button	This script is called when an event occurs on a sleep button device.
lid_switch	This script is called when an event occurs on a lid switch device.
acadapter	This script is called when an online or offline event occurs on an AC adapter device.
hotkey_button	This script is called when an event occurs on a hotkey button device.

The following events are defined for power switch devices:

pressed	The button was pressed, the lid closed, or the AC adapter connected.
released	The button was released, the lid opened, or the AC adapter disconnected. Note that power and sleep button devices usually do not post this type of event.

The following is an example of how a power button script might be invoked when a power button is pressed by the operator:

```
/etc/powerd/scripts/power_button acpi0 pressed
```

ENVSYS SCRIPTS

`envsys`(4) scripts are called when a condition was triggered in a sensor. These scripts are called with three arguments: the device associated, the event type and sensor's name. The **sensor_drive** and the **sensor_battery** scripts uses a fourth argument: state description.

The following envsys script names are defined:

sensor_battery	This script is called when an event occurs on a battery sensor (Wh/Ah/Battery state).
sensor_drive	This script is called when an event occurs on a drive sensor.
sensor_fan	This script is called when an event occurs on a fan sensor.
sensor_indicator	This script is called when an event ocurrs on a indicator/integer sensor.
sensor_power	This script is called when an event occurs on a power sensor (W/Ampere).
sensor_resistance	This script is called when an event occurs on a resistance sensor (Ohm).
sensor_temperature	This script is called when an event occurs on a temperature sensor.
sensor_voltage	This script is called when an event occurs on a voltage sensor.

The following event is defined for all scripts but is only sent if a critical/warning or any other event was previously sent:

normal	A normal state/capacity/condition was triggered.

The following events are defined for fan, indicator, power, resistance, temperature and voltage sensors:

critical	A critical condition was triggered.
critical-under	A critical under condition was triggered.
critical-over	A critical over condition was triggered.
warning-under	A warning under condition was triggered.
warning-over	A warning over condition was triggered.

The following events are defined only for battery sensors:

user-capacity	Capacity dropped below the limit set by the user.
low-power	System is running in low power, that means that there is not any AC Adapter connected and all batteries are in critical or low capacity. When this event is received there's no much time so you should only suspend or shutdown the system. The script shutdowns the system gracefully by default.

The following events are defined for drive and battery sensors:

state-changed	The state on the sensor has been changed and it's not in normal state.

The following is an example of how a temperature sensor script might be invoked when a critical over condition is triggered:

```
/etc/powerd/scripts/sensor_temperature lm0 critical-over "CPU Temp"
```

SEE ALSO

`acpi`(4), `acpiacad`(4), `acpibut`(4), `acpilid`(4), `envsys`(4), `i386/apm`(4)

HISTORY

powerd first appeared in NetBSD 2.0. Support to handle envsys(4) events appeared in NetBSD 5.0.

AUTHORS

powerd was written by Jason R. Thorpe ⟨thorpej@wasabisystems.com⟩ and contributed by Wasabi Systems, Inc.
Juan Romero Pardines added support to handle envsys(4) events.

BUGS

Due to its synchronous nature **powerd** cannot be trusted to handle events within a certain time.

NAME

pppd − Point-to-Point Protocol Daemon

SYNOPSIS

pppd [*options*]

DESCRIPTION

PPP is the protocol used for establishing internet links over dial-up modems, DSL connections, and many other types of point-to-point links. The *pppd* daemon works together with the kernel PPP driver to establish and maintain a PPP link with another system (called the *peer*) and to negotiate Internet Protocol (IP) addresses for each end of the link. Pppd can also authenticate the peer and/or supply authentication information to the peer. PPP can be used with other network protocols besides IP, but such use is becoming increasingly rare.

FREQUENTLY USED OPTIONS

ttyname
: Use the serial port called *ttyname* to communicate with the peer. If *ttyname* does not begin with a slash (/), the string "/dev/" is prepended to *ttyname* to form the name of the device to open. If no device name is given, or if the name of the terminal connected to the standard input is given, pppd will use that terminal, and will not fork to put itself in the background. A value for this option from a privileged source cannot be overridden by a non-privileged user.

speed
: An option that is a decimal number is taken as the desired baud rate for the serial device. On systems such as 4.4BSD and NetBSD, any speed can be specified. Other systems (e.g. Linux, SunOS) only support the commonly-used baud rates.

asyncmap *map*
: This option sets the Async-Control-Character-Map (ACCM) for this end of the link. The ACCM is a set of 32 bits, one for each of the ASCII control characters with values from 0 to 31, where a 1 bit indicates that the corresponding control character should not be used in PPP packets sent to this system. The map is encoded as a hexadecimal number (without a leading 0x) where the least significant bit (00000001) represents character 0 and the most significant bit (80000000) represents character 31. Pppd will ask the peer to send these characters as a 2-byte escape sequence. If multiple *asyncmap* options are given, the values are ORed together. If no *asyncmap* option is given, the default is zero, so pppd will ask the peer not to escape any control characters. To escape transmitted characters, use the *escape* option.

auth
: Require the peer to authenticate itself before allowing network packets to be sent or received. This option is the default if the system has a default route. If neither this option nor the *noauth* option is specified, pppd will only allow the peer to use IP addresses to which the system does not already have a route.

call *name*
: Read additional options from the file /etc/ppp/peers/*name*. This file may contain privileged options, such as *noauth*, even if pppd is not being run by root. The *name* string may not begin with / or include .. as a pathname component. The format of the options file is described below.

connect *script*
: Usually there is something which needs to be done to prepare the link before the PPP protocol can be started; for instance, with a dial-up modem, commands need to be sent to the modem to dial the appropriate phone number. This option specifies an command for pppd to execute (by passing it to a shell) before attempting to start PPP negotiation. The chat (8) program is often useful here, as it provides a way to send arbitrary strings to a modem and respond to received characters. A value for this option from a privileged source cannot be overridden by a non-privileged user.

crtscts
: Specifies that pppd should set the serial port to use hardware flow control using the RTS and CTS signals in the RS-232 interface. If neither the *crtscts*, the *nocrtscts*, the *cdtrcts* nor the *nocdtrcts* option is given, the hardware flow control setting for the serial port is left unchanged. Some serial ports (such as Macintosh serial ports) lack a true RTS output. Such serial ports use this mode to implement unidirectional flow control. The serial port will suspend transmission when requested

by the modem (via CTS) but will be unable to request the modem to stop sending to the computer. This mode retains the ability to use DTR as a modem control line.

defaultroute
Add a default route to the system routing tables, using the peer as the gateway, when IPCP negotiation is successfully completed. This entry is removed when the PPP connection is broken. This option is privileged if the *nodefaultroute* option has been specified.

disconnect *script*
Execute the command specified by *script*, by passing it to a shell, after pppd has terminated the link. This command could, for example, issue commands to the modem to cause it to hang up if hardware modem control signals were not available. The disconnect script is not run if the modem has already hung up. A value for this option from a privileged source cannot be overridden by a non-privileged user.

escape *xx,yy,...*
Specifies that certain characters should be escaped on transmission (regardless of whether the peer requests them to be escaped with its async control character map). The characters to be escaped are specified as a list of hex numbers separated by commas. Note that almost any character can be specified for the *escape* option, unlike the *asyncmap* option which only allows control characters to be specified. The characters which may not be escaped are those with hex values 0x20 - 0x3f or 0x5e.

file *name*
Read options from file *name* (the format is described below). The file must be readable by the user who has invoked pppd.

init *script*
Execute the command specified by *script*, by passing it to a shell, to initialize the serial line. This script would typically use the chat(8) program to configure the modem to enable auto answer. A value for this option from a privileged source cannot be overridden by a non-privileged user.

lock Specifies that pppd should create a UUCP-style lock file for the serial device to ensure exclusive access to the device. By default, pppd will not create a lock file.

mru *n* Set the MRU [Maximum Receive Unit] value to *n*. Pppd will ask the peer to send packets of no more than *n* bytes. The value of *n* must be between 128 and 16384; the default is 1500. A value of 296 works well on very slow links (40 bytes for TCP/IP header + 256 bytes of data). Note that for the IPv6 protocol, the MRU must be at least 1280.

mtu *n* Set the MTU [Maximum Transmit Unit] value to *n*. Unless the peer requests a smaller value via MRU negotiation, pppd will request that the kernel networking code send data packets of no more than *n* bytes through the PPP network interface. Note that for the IPv6 protocol, the MTU must be at least 1280.

passive Enables the "passive" option in the LCP. With this option, pppd will attempt to initiate a connection; if no reply is received from the peer, pppd will then just wait passively for a valid LCP packet from the peer, instead of exiting, as it would without this option.

OPTIONS

*<local_IP_address>***:***<remote_IP_address>*
Set the local and/or remote interface IP addresses. Either one may be omitted. The IP addresses can be specified with a host name or in decimal dot notation (e.g. 150.234.56.78). The default local address is the (first) IP address of the hostname of the system (unless the *noipdefault* option is given). The remote address will be obtained from the peer if not specified in any option. Thus, in simple cases, this option is not required. If a local and/or remote IP address is specified with this option, pppd will not accept a different value from the peer in the IPCP negotiation, unless the *ipcp−accept−local* and/or *ipcp−accept−remote* options are given, respectively.

ipv6 *<local_interface_identifier>,<remote_interface_identifier>*
Set the local and/or remote 64-bit interface identifier. Either one may be omitted. The identifier must be specified in standard ascii notation of IPv6 addresses (e.g. ::dead:beef). If the *ipv6cp−use−ipaddr* option is given, the local identifier is the local IPv4 address (see above). On systems which supports a unique persistent id, such as EUI−48 derived from the Ethernet MAC address, *ipv6cp−use−persistent* option can be used to replace the *ipv6 <local>,<remote>* option. Otherwise the identifier is randomized.

active−filter−in *filter−expression*

active−filter−out *filter−expression*
Specifies an incoming and outgoing packet filter to be applied to data packets to determine which packets are to be regarded as link activity, and therefore reset the idle timer, or cause the link to be brought up in demand-dialing mode. This option is useful in conjunction with the **idle** option if there are packets being sent or received regularly over the link (for example, routing information packets) which would otherwise prevent the link from ever appearing to be idle. The *filter−expression* syntax is as described for tcpdump(8), except that qualifiers which are inappropriate for a PPP link, such as **ether** and **arp**, are not permitted. Generally the filter expression should be enclosed in single-quotes to prevent whitespace in the expression from being interpreted by the shell. This option is currently only available under NetBSD, and then only if both the kernel and pppd were compiled with PPP_FILTER defined.

allow−ip *address(es)*
Allow peers to use the given IP address or subnet without authenticating themselves. The parameter is parsed as for each element of the list of allowed IP addresses in the secrets files (see the AUTHENTICATION section below).

allow−number *number*
Allow peers to connect from the given telephone number. A trailing ‘*’ character will match all numbers beginning with the leading part.

bsdcomp *nr,nt*
Request that the peer compress packets that it sends, using the BSD-Compress scheme, with a maximum code size of *nr* bits, and agree to compress packets sent to the peer with a maximum code size of *nt* bits. If *nt* is not specified, it defaults to the value given for *nr*. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values give better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use *nobsdcomp* or *bsdcomp 0* to disable BSD-Compress compression entirely.

callback *phone_number*
Request a call-back to the *phone-number*. This only works if the peer is speaking the Call Back Configuration Protocol. Don’t put this into the main options file if you sometimes connect to servers that don’t support it.

cdtrcts Use a non-standard hardware flow control (i.e. DTR/CTS) to control the flow of data on the serial port. If neither the *crtscts*, the *nocrtscts*, the *cdtrcts* nor the *nocdtrcts* option is given, the hardware flow control setting for the serial port is left unchanged. Some serial ports (such as Macintosh serial ports) lack a true RTS output. Such serial ports use this mode to implement true bi-directional flow control. The sacrifice is that this flow control mode does not permit using DTR as a modem control line.

chap−interval *n*
If this option is given, pppd will rechallenge the peer every *n* seconds.

chap−max−challenge *n*
Set the maximum number of CHAP challenge transmissions to *n* (default 10).

chap−restart *n*
Set the CHAP restart interval (retransmission timeout for challenges) to *n* seconds (default 3).

child−timeout *n*
When exiting, wait for up to *n* seconds for any child processes (such as the command specified with the **pty** command) to exit before exiting. At the end of the timeout, pppd will send a SIGTERM signal to any remaining child processes and exit. A value of 0 means no timeout, that is, pppd will wait until all child processes have exited.

connect−delay *n*
Wait for up to *n* milliseconds after the connect script finishes for a valid PPP packet from the peer. At the end of this time, or when a valid PPP packet is received from the peer, pppd will commence negotiation by sending its first LCP packet. The default value is 1000 (1 second). This wait period only applies if the **connect** or **pty** option is used.

debug Enables connection debugging facilities. If this option is given, pppd will log the contents of all control packets sent or received in a readable form. The packets are logged through syslog with facility *daemon* and level *debug*. This information can be directed to a file by setting up /etc/syslog.conf appropriately (see syslog.conf(5)).

default−asyncmap
Disable asyncmap negotiation, forcing all control characters to be escaped for both the transmit and the receive direction.

default−mru
Disable MRU [Maximum Receive Unit] negotiation. With this option, pppd will use the default MRU value of 1500 bytes for both the transmit and receive direction.

deflate *nr,nt*
Request that the peer compress packets that it sends, using the Deflate scheme, with a maximum window size of *2**nr* bytes, and agree to compress packets sent to the peer with a maximum window size of *2**nt* bytes. If *nt* is not specified, it defaults to the value given for *nr*. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values give better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use *nodeflate* or *deflate 0* to disable Deflate compression entirely. (Note: pppd requests Deflate compression in preference to BSD-Compress if the peer can do either.)

demand
Initiate the link only on demand, i.e. when data traffic is present. With this option, the remote IP address must be specified by the user on the command line or in an options file. Pppd will initially configure the interface and enable it for IP traffic without connecting to the peer. When traffic is available, pppd will connect to the peer and perform negotiation, authentication, etc. When this is completed, pppd will commence passing data packets (i.e., IP packets) across the link.

The *demand* option implies the *persist* option. If this behavior is not desired, use the *nopersist* option after the *demand* option. The *idle* and *holdoff* options are also useful in conjunction with the *demand* option.

domain *d*
Append the domain name *d* to the local host name for authentication purposes. For example, if gethostname() returns the name porsche, but the fully qualified domain name is porsche.Quotron.COM, you could specify *domain Quotron.COM*. Pppd would then use the name *porsche.Quotron.COM* for looking up secrets in the secrets file, and as the default name to send to the peer when authenticating itself to the peer. This option is privileged.

dryrun With the **dryrun** option, pppd will print out all the option values which have been set and then exit, after parsing the command line and options files and checking the option values, but before initiating the link. The option values are logged at level info, and also printed to standard output unless the device on standard output is the device that pppd would be using to communicate with the peer.

dump With the **dump** option, pppd will print out all the option values which have been set. This option is like the **dryrun** option except that pppd proceeds as normal rather than exiting.

endpoint <*epdisc*>
Sets the endpoint discriminator sent by the local machine to the peer during multilink negotiation to <*epdisc*>. The default is to use the MAC address of the first ethernet interface on the system, if any, otherwise the IPv4 address corresponding to the hostname, if any, provided it is not in the multicast or locally-assigned IP address ranges, or the localhost address. The endpoint discriminator can be the string **null** or of the form *type*:*value*, where type is a decimal number or one of the strings **local**, **IP**, **MAC**, **magic**, or **phone**. The value is an IP address in dotted-decimal notation for the **IP** type, or a string of bytes in hexadecimal, separated by periods or colons for the other types. For the MAC type, the value may also be the name of an ethernet or similar network interface. This option is currently only available under Linux.

eap−interval *n*
If this option is given and pppd authenticates the peer with EAP (i.e., is the server), pppd will restart EAP authentication every *n* seconds. For EAP SRP−SHA1, see also the **srp−interval** option, which enables lightweight rechallenge.

eap−max−rreq *n*
Set the maximum number of EAP Requests to which pppd will respond (as a client) without hearing EAP Success or Failure. (Default is 20.)

eap−max−sreq *n*
Set the maximum number of EAP Requests that pppd will issue (as a server) while attempting authentication. (Default is 10.)

eap−restart *n*
Set the retransmit timeout for EAP Requests when acting as a server (authenticator). (Default is 3 seconds.)

eap−timeout *n*
Set the maximum time to wait for the peer to send an EAP Request when acting as a client (authenticatee). (Default is 20 seconds.)

hide−password
When logging the contents of PAP packets, this option causes pppd to exclude the password string from the log. This is the default.

holdoff *n*
Specifies how many seconds to wait before re-initiating the link after it terminates. This option only has any effect if the *persist* or *demand* option is used. The holdoff period is not applied if the link was terminated because it was idle.

idle *n* Specifies that pppd should disconnect if the link is idle for *n* seconds. The link is idle when no data packets (i.e. IP packets) are being sent or received. Note: it is not advisable to use this option with the *persist* option without the *demand* option. If the **active−filter−in** and/or **active−filter−out** options are given, data packets which are rejected by the specified activity filter also count as the link being idle.

ipcp−accept−local
With this option, pppd will accept the peer's idea of our local IP address, even if the local IP address was specified in an option.

ipcp−accept−remote
With this option, pppd will accept the peer's idea of its (remote) IP address, even if the remote IP address was specified in an option.

ipcp−max−configure *n*
Set the maximum number of IPCP configure-request transmissions to *n* (default 10).

ipcp−max−failure *n*
Set the maximum number of IPCP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

ipcp−max−terminate *n*
Set the maximum number of IPCP terminate-request transmissions to *n* (default 3).

ipcp−restart *n*
Set the IPCP restart interval (retransmission timeout) to *n* seconds (default 3).

ipparam *string*
Provides an extra parameter to the ip−up, ip−pre−up and ip−down scripts. If this option is given, the *string* supplied is given as the 6th parameter to those scripts.

+ipv6 Enable IPv6CP negotiation and IPv6 communication. It needs to be explicitly specified if you want IPv6CP.

−ipv6 Disable IPv6CP negotiation and IPv6 communication.

ipv6cp−accept−local
With this option, pppd will accept the peer's idea of our local IPv6 address, even if the local IPv6 address was specified in an option.

ipv6cp−use−ipaddr
Use the local IPv4 address as the local interface address.

ipv6cp−use−persistent
Use uniquely-available persistent value for link local address (Solaris 2 only).

ipv6cp−max−configure *n*
Set the maximum number of IPv6CP configure-request transmissions to *n* (default 10).

ipv6cp−max−failure *n*
Set the maximum number of IPv6CP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

ipv6cp−max−terminate *n*
Set the maximum number of IPv6CP terminate-request transmissions to *n* (default 3).

ipv6cp−restart *n*
Set the IPv6CP restart interval (retransmission timeout) to *n* seconds (default 3).

ipx Enable the IPXCP and IPX protocols. This option is presently only supported under Linux, and only if your kernel has been configured to include IPX support.

ipx−network *n*
Set the IPX network number in the IPXCP configure request frame to *n*, a hexadecimal number (without a leading 0x). There is no valid default. If this option is not specified, the network number is obtained from the peer. If the peer does not have the network number, the IPX protocol will not be started.

ipx−node *n***:***m*
Set the IPX node numbers. The two node numbers are separated from each other with a colon character. The first number *n* is the local node number. The second number *m* is the peer's node number. Each node number is a hexadecimal number, at most 10 digits long. The node numbers on the ipx−network must be unique. There is no valid default. If this option is not specified then the node numbers are obtained from the peer.

ipx−router−name *<string>*
Set the name of the router. This is a string and is sent to the peer as information data.

ipx−routing *n*
: Set the routing protocol to be received by this option. More than one instance of *ipx−routing* may be specified. The ’*none*’ option (0) may be specified as the only instance of ipx−routing. The values may be *0* for *NONE*, *2* for *RIP/SAP*, and *4* for *NLSP*.

ipxcp−accept−local
: Accept the peer’s NAK for the node number specified in the ipx−node option. If a node number was specified, and non-zero, the default is to insist that the value be used. If you include this option then you will permit the peer to override the entry of the node number.

ipxcp−accept−network
: Accept the peer’s NAK for the network number specified in the ipx−network option. If a network number was specified, and non-zero, the default is to insist that the value be used. If you include this option then you will permit the peer to override the entry of the node number.

ipxcp−accept−remote
: Use the peer’s network number specified in the configure request frame. If a node number was specified for the peer and this option was not specified, the peer will be forced to use the value which you have specified.

ipxcp−max−configure *n*
: Set the maximum number of IPXCP configure request frames which the system will send to *n*. The default is 10.

ipxcp−max−failure *n*
: Set the maximum number of IPXCP NAK frames which the local system will send before it rejects the options. The default value is 3.

ipxcp−max−terminate *n*
: Set the maximum number of IPXCP terminate request frames before the local system considers that the peer is not listening to them. The default value is 3.

kdebug *n*
: Enable debugging code in the kernel-level PPP driver. The argument values depend on the specific kernel driver, but in general a value of 1 will enable general kernel debug messages. (Note that these messages are usually only useful for debugging the kernel driver itself.) For the Linux 2.2.x kernel driver, the value is a sum of bits: 1 to enable general debug messages, 2 to request that the contents of received packets be printed, and 4 to request that the contents of transmitted packets be printed. On most systems, messages printed by the kernel are logged by syslogd(8) to a file as directed in the /etc/syslog.conf configuration file.

ktune
: Enables pppd to alter kernel settings as appropriate. Under Linux, pppd will enable IP forwarding (i.e. set /proc/sys/net/ipv4/ip_forward to 1) if the *proxyarp* option is used, and will enable the dynamic IP address option (i.e. set /proc/sys/net/ipv4/ip_dynaddr to 1) in demand mode if the local address changes.

lcp−echo−failure *n*
: If this option is given, pppd will presume the peer to be dead if *n* LCP echo−requests are sent without receiving a valid LCP echo−reply. If this happens, pppd will terminate the connection. Use of this option requires a non-zero value for the *lcp−echo−interval* parameter. This option can be used to enable pppd to terminate after the physical connection has been broken (e.g., the modem has hung up) in situations where no hardware modem control lines are available.

lcp−echo−interval *n*
: If this option is given, pppd will send an LCP echo−request frame to the peer every *n* seconds. Normally the peer should respond to the echo−request by sending an echo−reply. This option can be used with the *lcp−echo−failure* option to detect that the peer is no longer connected.

lcp−max−configure *n*
: Set the maximum number of LCP configure-request transmissions to *n* (default 10).

lcp−max−failure *n*
Set the maximum number of LCP configure-NAKs returned before starting to send configure-Rejects instead to *n* (default 10).

lcp−max−terminate *n*
Set the maximum number of LCP terminate-request transmissions to *n* (default 3).

lcp−restart *n*
Set the LCP restart interval (retransmission timeout) to *n* seconds (default 3).

linkname *name*
Sets the logical name of the link to *name*. Pppd will create a file named **ppp−***name***.pid** in /var/run (or /etc/ppp on some systems) containing its process ID. This can be useful in determining which instance of pppd is responsible for the link to a given peer system. This is a privileged option.

local Don't use the modem control lines. With this option, pppd will ignore the state of the CD (Carrier Detect) signal from the modem and will not change the state of the DTR (Data Terminal Ready) signal. This is the opposite of the **modem** option.

logfd *n* Send log messages to file descriptor *n*. Pppd will send log messages to at most one file or file descriptor (as well as sending the log messages to syslog), so this option and the **logfile** option are mutually exclusive. The default is for pppd to send log messages to stdout (file descriptor 1), unless the serial port is already open on stdout.

logfile *filename*
Append log messages to the file *filename* (as well as sending the log messages to syslog). The file is opened with the privileges of the user who invoked pppd, in append mode.

login Use the system password database for authenticating the peer using PAP, and record the user in the system wtmp file. Note that the peer must have an entry in the /etc/ppp/pap−secrets file as well as the system password database to be allowed access.

maxconnect *n*
Terminate the connection when it has been available for network traffic for *n* seconds (i.e. *n* seconds after the first network control protocol comes up).

maxfail *n*
Terminate after *n* consecutive failed connection attempts. A value of 0 means no limit. The default value is 10.

modem
Use the modem control lines. This option is the default. With this option, pppd will wait for the CD (Carrier Detect) signal from the modem to be asserted when opening the serial device (unless a connect script is specified), and it will drop the DTR (Data Terminal Ready) signal briefly when the connection is terminated and before executing the connect script. On Ultrix, this option implies hardware flow control, as for the *crtscts* option. This is the opposite of the **local** option.

mp Enables the use of PPP multilink; this is an alias for the 'multilink' option. This option is currently only available under Linux.

mppe−stateful
Allow MPPE to use stateful mode. Stateless mode is still attempted first. The default is to disallow stateful mode.

mpshortseq
Enables the use of short (12-bit) sequence numbers in multilink headers, as opposed to 24-bit sequence numbers. This option is only available under Linux, and only has any effect if multilink is enabled (see the multilink option).

mrru *n* Sets the Maximum Reconstructed Receive Unit to *n*. The MRRU is the maximum size for a received packet on a multilink bundle, and is analogous to the MRU for the individual links. This option is currently only available under Linux, and only has any effect if multilink is enabled (see the multilink option).

ms−dns *<addr>*
If pppd is acting as a server for Microsoft Windows clients, this option allows pppd to supply one or two DNS (Domain Name Server) addresses to the clients. The first instance of this option specifies the primary DNS address; the second instance (if given) specifies the secondary DNS address. (This option was present in some older versions of pppd under the name **dns−addr**.)

ms−wins *<addr>*
If pppd is acting as a server for Microsoft Windows or "Samba" clients, this option allows pppd to supply one or two WINS (Windows Internet Name Services) server addresses to the clients. The first instance of this option specifies the primary WINS address; the second instance (if given) specifies the secondary WINS address.

multilink
Enables the use of the PPP multilink protocol. If the peer also supports multilink, then this link can become part of a bundle between the local system and the peer. If there is an existing bundle to the peer, pppd will join this link to that bundle, otherwise pppd will create a new bundle. See the MULTILINK section below. This option is currently only available under Linux.

name *name*
Set the name of the local system for authentication purposes to *name*. This is a privileged option. With this option, pppd will use lines in the secrets files which have *name* as the second field when looking for a secret to use in authenticating the peer. In addition, unless overridden with the *user* option, *name* will be used as the name to send to the peer when authenticating the local system to the peer. (Note that pppd does not append the domain name to *name*.)

noaccomp
Disable Address/Control compression in both directions (send and receive).

noauth Do not require the peer to authenticate itself. This option is privileged.

nobsdcomp
Disables BSD-Compress compression; **pppd** will not request or agree to compress packets using the BSD-Compress scheme.

noccp Disable CCP (Compression Control Protocol) negotiation. This option should only be required if the peer is buggy and gets confused by requests from pppd for CCP negotiation.

nocrtscts
Disable hardware flow control (i.e. RTS/CTS) on the serial port. If neither the *crtscts* nor the *nocrtscts* nor the *cdtrcts* nor the *nocdtrcts* option is given, the hardware flow control setting for the serial port is left unchanged.

nocdtrcts
This option is a synonym for *nocrtscts*. Either of these options will disable both forms of hardware flow control.

nodefaultroute
Disable the *defaultroute* option. The system administrator who wishes to prevent users from creating default routes with pppd can do so by placing this option in the /etc/ppp/options file.

nodeflate
Disables Deflate compression; pppd will not request or agree to compress packets using the Deflate scheme.

nodetach
Don't detach from the controlling terminal. Without this option, if a serial device other than the terminal on the standard input is specified, pppd will fork to become a background process.

noendpoint
Disables pppd from sending an endpoint discriminator to the peer or accepting one from the peer (see the MULTILINK section below). This option should only be required if the peer is buggy.

noip Disable IPCP negotiation and IP communication. This option should only be required if the peer is buggy and gets confused by requests from pppd for IPCP negotiation.

noipv6 An alias for **-ipv6.**

noipdefault
Disables the default behavior when no local IP address is specified, which is to determine (if possible) the local IP address from the hostname. With this option, the peer will have to supply the local IP address during IPCP negotiation (unless it specified explicitly on the command line or in an options file).

noipx Disable the IPXCP and IPX protocols. This option should only be required if the peer is buggy and gets confused by requests from pppd for IPXCP negotiation.

noktune
Opposite of the *ktune* option; disables pppd from changing system settings.

nolock Opposite of the *lock* option; specifies that pppd should not create a UUCP-style lock file for the serial device. This option is privileged.

nolog Do not send log messages to a file or file descriptor. This option cancels the **logfd** and **logfile** options.

nomagic
Disable magic number negotiation. With this option, pppd cannot detect a looped-back line. This option should only be needed if the peer is buggy.

nomp Disables the use of PPP multilink. This option is currently only available under Linux.

nomppe
Disables MPPE (Microsoft Point to Point Encryption). This is the default.

nomppe−40
Disable 40-bit encryption with MPPE.

nomppe−128
Disable 128-bit encryption with MPPE.

nomppe−stateful
Disable MPPE stateful mode. This is the default.

nompshortseq
Disables the use of short (12-bit) sequence numbers in the PPP multilink protocol, forcing the use of 24-bit sequence numbers. This option is currently only available under Linux, and only has any effect if multilink is enabled.

nomultilink
Disables the use of PPP multilink. This option is currently only available under Linux.

nopcomp
Disable protocol field compression negotiation in both the receive and the transmit direction.

nopersist
Exit once a connection has been made and terminated. This is the default unless the *persist* or *demand* option has been specified.

nopredictor1
Do not accept or agree to Predictor−1 compression.

noproxyarp
Disable the *proxyarp* option. The system administrator who wishes to prevent users from creating proxy ARP entries with pppd can do so by placing this option in the /etc/ppp/options file.

notty Normally, pppd requires a terminal device. With this option, pppd will allocate itself a pseudo-tty master/slave pair and use the slave as its terminal device. Pppd will create a child process to act as a ‘character shunt’ to transfer characters between the pseudo-tty master and its standard input and output. Thus pppd will transmit characters on its standard output and receive characters on its standard input even if they are not terminal devices. This option increases the latency and CPU overhead of transferring data over the ppp interface as all of the characters sent and received must flow through the character shunt process. An explicit device name may not be given if this option is used.

novj Disable Van Jacobson style TCP/IP header compression in both the transmit and the receive direction.

novjccomp
Disable the connection-ID compression option in Van Jacobson style TCP/IP header compression. With this option, pppd will not omit the connection-ID byte from Van Jacobson compressed TCP/IP headers, nor ask the peer to do so.

papcrypt
Indicates that all secrets in the /etc/ppp/pap−secrets file which are used for checking the identity of the peer are encrypted, and thus pppd should not accept a password which, before encryption, is identical to the secret from the /etc/ppp/pap−secrets file.

pap−max−authreq *n*
Set the maximum number of PAP authenticate-request transmissions to *n* (default 10).

pap−restart *n*
Set the PAP restart interval (retransmission timeout) to *n* seconds (default 3).

pap−timeout *n*
Set the maximum time that pppd will wait for the peer to authenticate itself with PAP to *n* seconds (0 means no limit).

pass−filter−in *filter−expression*

pass−filter−out *filter−expression*
Specifies an incoming and outgoing packet filter to applied to data packets being sent or received to determine which packets should be allowed to pass. Packets which are rejected by the filter are silently discarded. This option can be used to prevent specific network daemons (such as routed) using up link bandwidth, or to provide a basic firewall capability. The *filter−expression* syntax is as described for tcpdump(8), except that qualifiers which are inappropriate for a PPP link, such as **ether** and **arp**, are not permitted. Generally the filter expression should be enclosed in single-quotes to prevent whitespace in the expression from being interpreted by the shell. This option is currently only available under NetBSD, and then only if both the kernel and pppd were compiled with PPP_FILTER defined.

password *password−string*
Specifies the password to use for authenticating to the peer. Use of this option is discouraged, as the password is likely to be visible to other users on the system (for example, by using ps(1)).

persist Do not exit after a connection is terminated; instead try to reopen the connection. The **maxfail** option still has an effect on persistent connections.

plugin *filename*
Load the shared library object file *filename* as a plugin. This is a privileged option. If *filename* does not contain a slash (/), pppd will look in the **/usr/lib/pppd/***version* directory for the plugin, where *version* is the version number of pppd (for example, 2.4.2).

predictor1
Request that the peer compress frames that it sends using Predictor-1 compression, and agree to compress transmitted frames with Predictor-1 if requested. This option has no effect unless the kernel driver supports Predictor-1 compression.

privgroup *group−name*
Allows members of group *group−name* to use privileged options. This is a privileged option. Use of this option requires care as there is no guarantee that members of *group−name* cannot use pppd

to become root themselves. Consider it equivalent to putting the members of *group−name* in the kmem or disk group.

proxyarp
Add an entry to this system's ARP [Address Resolution Protocol] table with the IP address of the peer and the Ethernet address of this system. This will have the effect of making the peer appear to other systems to be on the local ethernet.

pty *script*
Specifies that the command *script* is to be used to communicate rather than a specific terminal device. Pppd will allocate itself a pseudo-tty master/slave pair and use the slave as its terminal device. The *script* will be run in a child process with the pseudo-tty master as its standard input and output. An explicit device name may not be given if this option is used. (Note: if the *record* option is used in conjunction with the *pty* option, the child process will have pipes on its standard input and output.)

receive−all
With this option, pppd will accept all control characters from the peer, including those marked in the receive asyncmap. Without this option, pppd will discard those characters as specified in RFC1662. This option should only be needed if the peer is buggy.

record *filename*
Specifies that pppd should record all characters sent and received to a file named *filename*. This file is opened in append mode, using the user's user-ID and permissions. This option is implemented using a pseudo-tty and a process to transfer characters between the pseudo-tty and the real serial device, so it will increase the latency and CPU overhead of transferring data over the ppp interface. The characters are stored in a tagged format with timestamps, which can be displayed in readable form using the pppdump(8) program.

remotename *name*
Set the assumed name of the remote system for authentication purposes to *name*.

remotenumber *number*
Set the assumed telephone number of the remote system for authentication purposes to *number*.

refuse−chap
With this option, pppd will not agree to authenticate itself to the peer using CHAP.

refuse−mschap
With this option, pppd will not agree to authenticate itself to the peer using MS−CHAP.

refuse−mschap−v2
With this option, pppd will not agree to authenticate itself to the peer using MS−CHAPv2.

refuse−eap
With this option, pppd will not agree to authenticate itself to the peer using EAP.

refuse−pap
With this option, pppd will not agree to authenticate itself to the peer using PAP.

require−chap
Require the peer to authenticate itself using CHAP [Challenge Handshake Authentication Protocol] authentication.

require−mppe
Require the use of MPPE (Microsoft Point to Point Encryption). This option disables all other compression types. This option enables both 40-bit and 128-bit encryption. In order for MPPE to successfully come up, you must have authenticated with either MS−CHAP or MS−CHAPv2. This option is presently only supported under Linux, and only if your kernel has been configured to include MPPE support.

require−mppe−40
Require the use of MPPE, with 40-bit encryption.

require−mppe−128
Require the use of MPPE, with 128-bit encryption.

require−mschap
Require the peer to authenticate itself using MS−CHAP [Microsoft Challenge Handshake Authentication Protocol] authentication.

require−mschap−v2
Require the peer to authenticate itself using MS−CHAPv2 [Microsoft Challenge Handshake Authentication Protocol, Version 2] authentication.

require−eap
Require the peer to authenticate itself using EAP [Extensible Authentication Protocol] authentication.

require−pap
Require the peer to authenticate itself using PAP [Password Authentication Protocol] authentication.

show−password
When logging the contents of PAP packets, this option causes pppd to show the password string in the log message.

silent With this option, pppd will not transmit LCP packets to initiate a connection until a valid LCP packet is received from the peer (as for the ‘passive’ option with ancient versions of pppd).

srp−interval *n*
If this parameter is given and pppd uses EAP SRP−SHA1 to authenticate the peer (i.e., is the server), then pppd will use the optional lightweight SRP rechallenge mechanism at intervals of *n* seconds. This option is faster than **eap−interval** reauthentication because it uses a hash−based mechanism and does not derive a new session key.

srp−pn−secret *string*
Set the long-term pseudonym-generating secret for the server. This value is optional and if set, needs to be known at the server (authenticator) side only, and should be different for each server (or poll of identical servers). It is used along with the current date to generate a key to encrypt and decrypt the client’s identity contained in the pseudonym.

srp−use−pseudonym
When operating as an EAP SRP−SHA1 client, attempt to use the pseudonym stored in ˜/.ppp_psuedonym first as the identity, and save in this file any pseudonym offered by the peer during authentication.

sync Use synchronous HDLC serial encoding instead of asynchronous. The device used by pppd with this option must have sync support. Currently supports Microgate SyncLink adapters under Linux and FreeBSD 2.2.8 and later.

unit *num*
Sets the ppp unit number (for a ppp0 or ppp1 etc interface name) for outbound connections.

updetach
With this option, pppd will detach from its controlling terminal once it has successfully established the ppp connection (to the point where the first network control protocol, usually the IP control protocol, has come up).

usehostname
Enforce the use of the hostname (with domain name appended, if given) as the name of the local system for authentication purposes (overrides the *name* option). This option is not normally needed since the *name* option is privileged.

usepeerdns
Ask the peer for up to 2 DNS server addresses. The addresses supplied by the peer (if any) are passed to the /etc/ppp/ip−up script in the environment variables DNS1 and DNS2, and the environment variable USEPEERDNS will be set to 1. In addition, pppd will create an /etc/ppp/resolv.conf file containing one or two nameserver lines with the address(es) supplied by the peer.

user *name*
Sets the name used for authenticating the local system to the peer to *name*.

vj−max−slots *n*
Sets the number of connection slots to be used by the Van Jacobson TCP/IP header compression and decompression code to *n*, which must be between 2 and 16 (inclusive).

welcome *script*
Run the executable or shell command specified by *script* before initiating PPP negotiation, after the connect script (if any) has completed. A value for this option from a privileged source cannot be overridden by a non-privileged user.

xonxoff
Use software flow control (i.e. XON/XOFF) to control the flow of data on the serial port.

OPTIONS FILES

Options can be taken from files as well as the command line. Pppd reads options from the files /etc/ppp/options, ˜/.ppprc and /etc/ppp/options.*ttyname* (in that order) before processing the options on the command line. (In fact, the command-line options are scanned to find the terminal name before the options.*ttyname* file is read.) In forming the name of the options.*ttyname* file, the initial /dev/ is removed from the terminal name, and any remaining / characters are replaced with dots.

An options file is parsed into a series of words, delimited by whitespace. Whitespace can be included in a word by enclosing the word in double-quotes ("). A backslash (\) quotes the following character. A hash (#) starts a comment, which continues until the end of the line. There is no restriction on using the *file* or *call* options within an options file.

SECURITY

pppd provides system administrators with sufficient access control that PPP access to a server machine can be provided to legitimate users without fear of compromising the security of the server or the network it's on. This control is provided through restrictions on which IP addresses the peer may use, based on its authenticated identity (if any), and through restrictions on which options a non-privileged user may use. Several of pppd's options are privileged, in particular those which permit potentially insecure configurations; these options are only accepted in files which are under the control of the system administrator, or if pppd is being run by root.

The default behaviour of pppd is to allow an unauthenticated peer to use a given IP address only if the system does not already have a route to that IP address. For example, a system with a permanent connection to the wider internet will normally have a default route, and thus all peers will have to authenticate themselves in order to set up a connection. On such a system, the *auth* option is the default. On the other hand, a system where the PPP link is the only connection to the internet will not normally have a default route, so the peer will be able to use almost any IP address without authenticating itself.

As indicated above, some security-sensitive options are privileged, which means that they may not be used by an ordinary non-privileged user running a setuid-root pppd, either on the command line, in the user's ˜/.ppprc file, or in an options file read using the *file* option. Privileged options may be used in /etc/ppp/options file or in an options file read using the *call* option. If pppd is being run by the root user, privileged options can be used without restriction.

When opening the device, pppd uses either the invoking user's user ID or the root UID (that is, 0), depending on whether the device name was specified by the user or the system administrator. If the device name comes from a privileged source, that is, /etc/ppp/options or an options file read using the *call* option, pppd uses full root privileges when opening the device. Thus, by creating an appropriate file under /etc/ppp/peers, the system administrator can allow users to establish a ppp connection via a device which they would not normally have permission to access. Otherwise pppd uses the invoking user's real UID when opening the device.

AUTHENTICATION

Authentication is the process whereby one peer convinces the other of its identity. This involves the first peer sending its name to the other, together with some kind of secret information which could only come from the genuine authorized user of that name. In such an exchange, we will call the first peer the "client" and the other the "server". The client has a name by which it identifies itself to the server, and the server also has a name by which it identifies itself to the client. Generally the genuine client shares some secret (or password) with the server, and authenticates itself by proving that it knows that secret. Very often, the names used for authentication correspond to the internet hostnames of the peers, but this is not essential.

At present, pppd supports three authentication protocols: the Password Authentication Protocol (PAP), Challenge Handshake Authentication Protocol (CHAP), and Extensible Authentication Protocol (EAP). PAP involves the client sending its name and a cleartext password to the server to authenticate itself. In contrast, the server initiates the CHAP authentication exchange by sending a challenge to the client (the challenge packet includes the server's name). The client must respond with a response which includes its name plus a hash value derived from the shared secret and the challenge, in order to prove that it knows the secret. EAP supports CHAP-style authentication, and also includes the SRP−SHA1 mechanism, which is resistant to dictionary-based attacks and does not require a cleartext password on the server side.

The PPP protocol, being symmetrical, allows both peers to require the other to authenticate itself. In that case, two separate and independent authentication exchanges will occur. The two exchanges could use different authentication protocols, and in principle, different names could be used in the two exchanges.

The default behaviour of pppd is to agree to authenticate if requested, and to not require authentication from the peer. However, pppd will not agree to authenticate itself with a particular protocol if it has no secrets which could be used to do so.

Pppd stores secrets for use in authentication in secrets files (/etc/ppp/pap−secrets for PAP, /etc/ppp/chap−secrets for CHAP, MS−CHAP, MS−CHAPv2, and EAP MD5-Challenge, and /etc/ppp/srp−secrets for EAP SRP−SHA1). All secrets files have the same format. The secrets files can contain secrets for pppd to use in authenticating itself to other systems, as well as secrets for pppd to use when authenticating other systems to itself.

Each line in a secrets file contains one secret. A given secret is specific to a particular combination of client and server - it can only be used by that client to authenticate itself to that server. Thus each line in a secrets file has at least 3 fields: the name of the client, the name of the server, and the secret. These fields may be followed by a list of the IP addresses that the specified client may use when connecting to the specified server.

A secrets file is parsed into words as for a options file, so the client name, server name and secrets fields must each be one word, with any embedded spaces or other special characters quoted or escaped. Note that case is significant in the client and server names and in the secret.

If the secret starts with an '@', what follows is assumed to be the name of a file from which to read the secret. A "*" as the client or server name matches any name. When selecting a secret, pppd takes the best match, i.e. the match with the fewest wildcards.

Any following words on the same line are taken to be a list of acceptable IP addresses for that client. If there are only 3 words on the line, or if the first word is "−", then all IP addresses are disallowed. To allow any address, use "*". A word starting with "!" indicates that the specified address is *not* acceptable. An address may be followed by "/" and a number *n*, to indicate a whole subnet, i.e. all addresses which have the same value in the most significant *n* bits. In this form, the address may be followed by a plus sign ("+") to indicate that one address from the subnet is authorized, based on the ppp network interface unit number in use. In this case, the host part of the address will be set to the unit number plus one.

Thus a secrets file contains both secrets for use in authenticating other hosts, plus secrets which we use for authenticating ourselves to others. When pppd is authenticating the peer (checking the peer's identity), it chooses a secret with the peer's name in the first field and the name of the local system in the second field.

The name of the local system defaults to the hostname, with the domain name appended if the *domain* option is used. This default can be overridden with the *name* option, except when the *usehostname* option is used. (For EAP SRP−SHA1, see the srp−entry(8) utility for generating proper validator entries to be used in the "secret" field.)

When pppd is choosing a secret to use in authenticating itself to the peer, it first determines what name it is going to use to identify itself to the peer. This name can be specified by the user with the *user* option. If this option is not used, the name defaults to the name of the local system, determined as described in the previous paragraph. Then pppd looks for a secret with this name in the first field and the peer's name in the second field. Pppd will know the name of the peer if CHAP or EAP authentication is being used, because the peer will have sent it in the challenge packet. However, if PAP is being used, pppd will have to determine the peer's name from the options specified by the user. The user can specify the peer's name directly with the *remotename* option. Otherwise, if the remote IP address was specified by a name (rather than in numeric form), that name will be used as the peer's name. Failing that, pppd will use the null string as the peer's name.

When authenticating the peer with PAP, the supplied password is first compared with the secret from the secrets file. If the password doesn't match the secret, the password is encrypted using crypt() and checked against the secret again. Thus secrets for authenticating the peer can be stored in encrypted form if desired. If the *papcrypt* option is given, the first (unencrypted) comparison is omitted, for better security.

Furthermore, if the *login* option was specified, the username and password are also checked against the system password database. Thus, the system administrator can set up the pap−secrets file to allow PPP access only to certain users, and to restrict the set of IP addresses that each user can use. Typically, when using the *login* option, the secret in /etc/ppp/pap−secrets would be "", which will match any password supplied by the peer. This avoids the need to have the same secret in two places.

Authentication must be satisfactorily completed before IPCP (or any other Network Control Protocol) can be started. If the peer is required to authenticate itself, and fails to do so, pppd will terminated the link (by closing LCP). If IPCP negotiates an unacceptable IP address for the remote host, IPCP will be closed. IP packets can only be sent or received when IPCP is open.

In some cases it is desirable to allow some hosts which can't authenticate themselves to connect and use one of a restricted set of IP addresses, even when the local host generally requires authentication. If the peer refuses to authenticate itself when requested, pppd takes that as equivalent to authenticating with PAP using the empty string for the username and password. Thus, by adding a line to the pap−secrets file which specifies the empty string for the client and password, it is possible to allow restricted access to hosts which refuse to authenticate themselves.

ROUTING

When IPCP negotiation is completed successfully, pppd will inform the kernel of the local and remote IP addresses for the ppp interface. This is sufficient to create a host route to the remote end of the link, which will enable the peers to exchange IP packets. Communication with other machines generally requires further modification to routing tables and/or ARP (Address Resolution Protocol) tables. In most cases the *defaultroute* and/or *proxyarp* options are sufficient for this, but in some cases further intervention is required. The /etc/ppp/ip−up script can be used for this.

Sometimes it is desirable to add a default route through the remote host, as in the case of a machine whose only connection to the Internet is through the ppp interface. The *defaultroute* option causes pppd to create such a default route when IPCP comes up, and delete it when the link is terminated.

In some cases it is desirable to use proxy ARP, for example on a server machine connected to a LAN, in order to allow other hosts to communicate with the remote host. The *proxyarp* option causes pppd to look for a network interface on the same subnet as the remote host (an interface supporting broadcast and ARP, which is up and not a point-to-point or loopback interface). If found, pppd creates a permanent, published ARP entry with the IP address of the remote host and the hardware address of the network interface found.

When the *demand* option is used, the interface IP addresses have already been set at the point when IPCP comes up. If pppd has not been able to negotiate the same addresses that it used to configure the interface (for example when the peer is an ISP that uses dynamic IP address assignment), pppd has to change the interface IP addresses to the negotiated addresses. This may disrupt existing connections, and the use of demand dialing with peers that do dynamic IP address assignment is not recommended.

MULTILINK

Multilink PPP provides the capability to combine two or more PPP links between a pair of machines into a single ‘bundle’, which appears as a single virtual PPP link which has the combined bandwidth of the individual links. Currently, multilink PPP is only supported under Linux.

Pppd detects that the link it is controlling is connected to the same peer as another link using the peer’s endpoint discriminator and the authenticated identity of the peer (if it authenticates itself). The endpoint discriminator is a block of data which is hopefully unique for each peer. Several types of data can be used, including locally-assigned strings of bytes, IP addresses, MAC addresses, randomly strings of bytes, or E−164 phone numbers. The endpoint discriminator sent to the peer by pppd can be set using the endpoint option.

In some circumstances the peer may send no endpoint discriminator or a non-unique value. The bundle option adds an extra string which is added to the peer’s endpoint discriminator and authenticated identity when matching up links to be joined together in a bundle. The bundle option can also be used to allow the establishment of multiple bundles between the local system and the peer. Pppd uses a TDB database in /var/run/pppd2.tdb to match up links.

Assuming that multilink is enabled and the peer is willing to negotiate multilink, then when pppd is invoked to bring up the first link to the peer, it will detect that no other link is connected to the peer and create a new bundle, that is, another ppp network interface unit. When another pppd is invoked to bring up another link to the peer, it will detect the existing bundle and join its link to it.

If the first link terminates (for example, because of a hangup or a received LCP terminate-request) the bundle is not destroyed unless there are no other links remaining in the bundle. Rather than exiting, the first pppd keeps running after its link terminates, until all the links in the bundle have terminated. If the first pppd receives a SIGTERM or SIGINT signal, it will destroy the bundle and send a SIGHUP to the pppd processes for each of the links in the bundle. If the first pppd receives a SIGHUP signal, it will terminate its link but not the bundle.

Note: demand mode is not currently supported with multilink.

EXAMPLES

The following examples assume that the /etc/ppp/options file contains the *auth* option (as in the default /etc/ppp/options file in the ppp distribution).

Probably the most common use of pppd is to dial out to an ISP. This can be done with a command such as

```
pppd call isp
```

where the /etc/ppp/peers/isp file is set up by the system administrator to contain something like this:

```
ttyS0 19200 crtscts
connect ’/usr/sbin/chat −v −f /etc/ppp/chat−isp’
noauth
```

In this example, we are using chat to dial the ISP’s modem and go through any log on sequence required. The /etc/ppp/chat−isp file contains the script used by chat; it could for example contain something like this:

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
"" "at"
OK "at&d0&c1"
OK "atdt2468135"
"name:" "^Umyuserid"
```

```
"word:" "\qmypassword"
"ispts" "\q^Uppp"
"~-^Uppp-~"
```

See the chat(8) man page for details of chat scripts.

Pppd can also be used to provide a dial-in ppp service for users. If the users already have login accounts, the simplest way to set up the ppp service is to let the users log in to their accounts and run pppd (installed setuid-root) with a command such as

```
pppd proxyarp
```

To allow a user to use the PPP facilities, you need to allocate an IP address for that user's machine and create an entry in /etc/ppp/pap−secrets, /etc/ppp/chap−secrets, or /etc/ppp/srp−secrets (depending on which authentication method the PPP implementation on the user's machine supports), so that the user's machine can authenticate itself. For example, if Joe has a machine called "joespc" that is to be allowed to dial in to the machine called "server" and use the IP address joespc.my.net, you would add an entry like this to /etc/ppp/pap−secrets or /etc/ppp/chap−secrets:

```
joespc    server    "joe's secret"    joespc.my.net
```

(See srp−entry(8) for a means to generate the server's entry when SRP−SHA1 is in use.) Alternatively, you can create a username called (for example) "ppp", whose login shell is pppd and whose home directory is /etc/ppp. Options to be used when pppd is run this way can be put in /etc/ppp/.ppprc.

If your serial connection is any more complicated than a piece of wire, you may need to arrange for some control characters to be escaped. In particular, it is often useful to escape XON (^Q) and XOFF (^S), using *asyncmap a0000*. If the path includes a telnet, you probably should escape ^] as well (*asyncmap 200a0000*). If the path includes an rlogin, you will need to use the *escape ff* option on the end which is running the rlogin client, since many rlogin implementations are not transparent; they will remove the sequence [0xff, 0xff, 0x73, 0x73, followed by any 8 bytes] from the stream.

DIAGNOSTICS

Messages are sent to the syslog daemon using facility LOG_DAEMON. (This can be overridden by recompiling pppd with the macro LOG_PPP defined as the desired facility.) See the syslog(8) documentation for details of where the syslog daemon will write the messages. On most systems, the syslog daemon uses the /etc/syslog.conf file to specify the destination(s) for syslog messages. You may need to edit that file to suit.

The *debug* option causes the contents of all control packets sent or received to be logged, that is, all LCP, PAP, CHAP, EAP, or IPCP packets. This can be useful if the PPP negotiation does not succeed or if authentication fails. If debugging is enabled at compile time, the *debug* option also causes other debugging messages to be logged.

Debugging can also be enabled or disabled by sending a SIGUSR1 signal to the pppd process. This signal acts as a toggle.

EXIT STATUS

The exit status of pppd is set to indicate whether any error was detected, or the reason for the link being terminated. The values used are:

0 Pppd has detached, or otherwise the connection was successfully established and terminated at the peer's request.

1 An immediately fatal error of some kind occurred, such as an essential system call failing, or running out of virtual memory.

2 An error was detected in processing the options given, such as two mutually exclusive options being used.

3 Pppd is not setuid-root and the invoking user is not root.

4 The kernel does not support PPP, for example, the PPP kernel driver is not included or cannot be loaded.

5 Pppd terminated because it was sent a SIGINT, SIGTERM or SIGHUP signal.

6 The serial port could not be locked.

7 The serial port could not be opened.

8 The connect script failed (returned a non-zero exit status).

9 The command specified as the argument to the *pty* option could not be run.

10 The PPP negotiation failed, that is, it didn't reach the point where at least one network protocol (e.g. IP) was running.

11 The peer system failed (or refused) to authenticate itself.

12 The link was established successfully and terminated because it was idle.

13 The link was established successfully and terminated because the connect time limit was reached.

14 Callback was negotiated and an incoming call should arrive shortly.

15 The link was terminated because the peer is not responding to echo requests.

16 The link was terminated by the modem hanging up.

17 The PPP negotiation failed because serial loopback was detected.

18 The init script failed (returned a non-zero exit status).

19 We failed to authenticate ourselves to the peer.

SCRIPTS

Pppd invokes scripts at various stages in its processing which can be used to perform site-specific ancillary processing. These scripts are usually shell scripts, but could be executable code files instead. Pppd does not wait for the scripts to finish (except for the ip-pre-up script). The scripts are executed as root (with the real and effective user-id set to 0), so that they can do things such as update routing tables or run privileged daemons. Be careful that the contents of these scripts do not compromise your system's security. Pppd runs the scripts with standard input, output and error redirected to /dev/null, and with an environment that is empty except for some environment variables that give information about the link. The environment variables that pppd sets are:

DEVICE
The name of the serial tty device being used.

IFNAME
The name of the network interface being used.

IPLOCAL
The IP address for the local end of the link. This is only set when IPCP has come up.

IPREMOTE
The IP address for the remote end of the link. This is only set when IPCP has come up.

PEERNAME
The authenticated name of the peer. This is only set if the peer authenticates itself.

SPEED
The baud rate of the tty device.

ORIG_UID
The real user-id of the user who invoked pppd.

PPPLOGNAME
The username of the real user-id that invoked pppd. This is always set.

For the ip-down and auth-down scripts, pppd also sets the following variables giving statistics for the connection:

CONNECT_TIME
The number of seconds from when the PPP negotiation started until the connection was terminated.

BYTES_SENT
The number of bytes sent (at the level of the serial port) during the connection.

BYTES_RCVD
The number of bytes received (at the level of the serial port) during the connection.

LINKNAME
The logical name of the link, set with the *linkname* option.

DNS1 If the peer supplies DNS server addresses, this variable is set to the first DNS server address supplied.

DNS2 If the peer supplies DNS server addresses, this variable is set to the second DNS server address supplied.

Pppd invokes the following scripts, if they exist. It is not an error if they don't exist.

/etc/ppp/auth−up
A program or script which is executed after the remote system successfully authenticates itself. It is executed with the parameters

interface−name peer−name user−name tty−device speed

Note that this script is not executed if the peer doesn't authenticate itself, for example when the *noauth* option is used.

/etc/ppp/auth−down
A program or script which is executed when the link goes down, if /etc/ppp/auth−up was previously executed. It is executed in the same manner with the same parameters as /etc/ppp/auth−up.

/etc/ppp/ip−pre−up
A program or script which is executed just before the ppp network interface is brought up. It is executed with the same parameters as the ip−up script (below). At this point the interface exists and has IP addresses assigned but is still down. This can be used to add firewall rules before any IP traffic can pass through the interface. Pppd will wait for this script to finish before bringing the interface up, so this script should run quickly.

/etc/ppp/ip−up
A program or script which is executed when the link is available for sending and receiving IP packets (that is, IPCP has come up). It is executed with the parameters

interface−name tty−device speed local−IP−address remote−IP−address ipparam

/etc/ppp/ip−down
A program or script which is executed when the link is no longer available for sending and receiving IP packets. This script can be used for undoing the effects of the /etc/ppp/ip−up and /etc/ppp/ip−pre−up scripts. It is invoked in the same manner and with the same parameters as the ip−up script.

/etc/ppp/ipv6−up
Like /etc/ppp/ip−up, except that it is executed when the link is available for sending and receiving IPv6 packets. It is executed with the parameters

interface−name tty−device speed local−link−local−address remote−link−local−address ipparam

/etc/ppp/ipv6−down
Similar to /etc/ppp/ip−down, but it is executed when IPv6 packets can no longer be transmitted on the link. It is executed with the same parameters as the ipv6−up script.

/etc/ppp/ipx−up

A program or script which is executed when the link is available for sending and receiving IPX packets (that is, IPXCP has come up). It is executed with the parameters

interface−name tty−device speed network−number local−IPX−node−address remote−IPX−node−address local−IPX−routing−protocol remote−IPX−routing−protocol local−IPX−router−name remote−IPX−router−name ipparam pppd−pid

The local−IPX−routing−protocol and remote−IPX−routing−protocol field may be one of the following:

NONE	to indicate that there is no routing protocol
RIP	to indicate that RIP/SAP should be used
NLSP	to indicate that Novell NLSP should be used
RIP NLSP	to indicate that both RIP/SAP and NLSP should be used

/etc/ppp/ipx−down

A program or script which is executed when the link is no longer available for sending and receiving IPX packets. This script can be used for undoing the effects of the /etc/ppp/ipx−up script. It is invoked in the same manner and with the same parameters as the ipx−up script.

FILES

/var/run/ppp*n*.pid (BSD or Linux), **/etc/ppp/ppp*n*.pid** (others)

Process-ID for pppd process on ppp interface unit *n*.

/var/run/ppp−*name*.pid (BSD or Linux),

/etc/ppp/ppp−*name*.pid (others) Process-ID for pppd process for logical link *name* (see the *linkname* option).

/var/run/pppd2.tdb

Database containing information about pppd processes, interfaces and links, used for matching links to bundles in multilink operation. May be examined by external programs to obtain information about running pppd instances, the interfaces and devices they are using, IP address assignments, etc. **/etc/ppp/pap−secrets** Usernames, passwords and IP addresses for PAP authentication. This file should be owned by root and not readable or writable by any other user. Pppd will log a warning if this is not the case.

/etc/ppp/chap−secrets

Names, secrets and IP addresses for CHAP/MS−CHAP/MS−CHAPv2 authentication. As for /etc/ppp/pap−secrets, this file should be owned by root and not readable or writable by any other user. Pppd will log a warning if this is not the case.

/etc/ppp/srp−secrets

Names, secrets, and IP addresses for EAP authentication. As for /etc/ppp/pap−secrets, this file should be owned by root and not readable or writable by any other user. Pppd will log a warning if this is not the case.

˜/.ppp_pseudonym

Saved client-side SRP−SHA1 pseudonym. See the *srp−use−pseudonym* option for details.

/etc/ppp/options

System default options for pppd, read before user default options or command-line options.

˜/.ppprc

User default options, read before /etc/ppp/options.*ttyname*.

/etc/ppp/options.*ttyname*

System default options for the serial port being used, read after ˜/.ppprc. In forming the *ttyname* part of this filename, an initial /dev/ is stripped from the port name (if present), and any slashes in the remaining part are converted to dots.

/etc/ppp/peers
A directory containing options files which may contain privileged options, even if pppd was invoked by a user other than root. The system administrator can create options files in this directory to permit non-privileged users to dial out without requiring the peer to authenticate, but only to certain trusted peers.

SEE ALSO

chat(8), **pppstats**(8)

RFC1144
Jacobson, V. *Compressing TCP/IP headers for low-speed serial links.* February 1990.

RFC1321
Rivest, R. *The MD5 Message-Digest Algorithm.* April 1992.

RFC1332
McGregor, G. *PPP Internet Protocol Control Protocol (IPCP).* May 1992.

RFC1334
Lloyd, B.; Simpson, W.A. *PPP authentication protocols.* October 1992.

RFC1661
Simpson, W.A. *The Point-to-Point Protocol (PPP).* July 1994.

RFC1662
Simpson, W.A. *PPP in HDLC-like Framing.* July 1994.

RFC2284
Blunk, L.; Vollbrecht, J., *PPP Extensible Authentication Protocol (EAP).* March 1998.

RFC2472
Haskin, D. *IP Version 6 over PPP* December 1998.

RFC2945
Wu, T., *The SRP Authentication and Key Exchange System* September 2000.

draft−ietf−pppext−eap−srp−03.txt
Carlson, J.; et al., *EAP SRP−SHA1 Authentication Protocol.* July 2001.

NOTES

Some limited degree of control can be exercised over a running pppd process by sending it a signal from the list below.

SIGINT, SIGTERM
These signals cause pppd to terminate the link (by closing LCP), restore the serial device settings, and exit. If a connector or disconnector process is currently running, pppd will send the same signal to its process group, so as to terminate the connector or disconnector process.

SIGHUP
This signal causes pppd to terminate the link, restore the serial device settings, and close the serial device. If the *persist* or *demand* option has been specified, pppd will try to reopen the serial device and start another connection (after the holdoff period). Otherwise pppd will exit. If this signal is received during the holdoff period, it causes pppd to end the holdoff period immediately. If a connector or disconnector process is running, pppd will send the same signal to its process group.

SIGUSR1
This signal toggles the state of the *debug* option.

SIGUSR2
This signal causes pppd to renegotiate compression. This can be useful to re-enable compression after it has been disabled as a result of a fatal decompression error. (Fatal decompression errors generally indicate a bug in one or other implementation.)

AUTHORS

Paul Mackerras (paulus@samba.org), based on earlier work by Drew Perkins, Brad Clements, Karl Fox, Greg Christy, and Brad Parker.

COPYRIGHT

NAME

pppdump − convert PPP record file to readable format

SYNOPSIS

pppdump [**−h** | **−p** [**−d**]] [**−r**] [**−m** *mru*] [*file* ...]

DESCRIPTION

The **pppdump** utility converts the files written using the *record* option of **pppd** into a human-readable format. If one or more filenames are specified, **pppdump** will read each in turn; otherwise it will read its standard input. In each case the result is written to standard output.

The options are as follows:

−h Prints the bytes sent and received in hexadecimal. If neither this option nor the **−p** option is specified, the bytes are printed as the characters themselves, with non-printing and non-ASCII characters printed as escape sequences.

−p Collects the bytes sent and received into PPP packets, interpreting the async HDLC framing and escape characters and checking the FCS (frame check sequence) of each packet. The packets are printed as hex values and as characters (non-printable characters are printed as '.').

−d With the **−p** option, this option causes **pppdump** to decompress packets which have been compressed with the BSD-Compress or Deflate methods.

−r Reverses the direction indicators, so that 'sent' is printed for bytes or packets received, and 'rcvd' is printed for bytes or packets sent.

−m *mru*

Use *mru* as the MRU (maximum receive unit) for both directions of the link when checking for over-length PPP packets (with the **−p** option).

SEE ALSO

pppd(8)

NAME

pppoectl, **ipppctl** — display or set parameters for an pppoe or isdn ppp (ippp) interface

SYNOPSIS

pppoectl [**−v**] *ifname* [*parameter*[=*value*]] [*...*]

ipppctl [**−v**] *ifname* [*parameter*[=*value*]] [*...*]

pppoectl **−e** *ethernet-ifname* [**−s** *service-name*]
[**−a** *access-concentrator-name*] [**−d**] [**−n** *1* **|** *2*] *ifname*

pppoectl **−f** *config-file* *ifname* [*...*]

DESCRIPTION

There are two basic modes of operation: configuring security related parameters and attaching a PPPoE interface to its ethernet interface, optionally passing in additional parameters for the PPPoE encapsulation.

The later usage is indicated by the presence of the **−e** option, which takes the name of the ethernet interface as its argument.

−e specifies the ethernet interface used to communicate with the access concentrator (typically via a DSL modem).

−a specifies the name of the access concentrator.

−s specifies the name of the service connected to.

−d dump the current connection state information (this parameter is typically used alone, for informational purposes, not during interface configuration).

−n *1* **|** *2*
print the IP address of the primary or secondary DNS name server for this PPP connection. This is only available if DNS query is enabled, see *query-dns*.

−f parse *config-file* for *parameter*[=*value*] pairs, one per line, as if they had been specified on the command line. This allows the password to be not passed as a command line argument. Unless escaped by \, comments starting with # to the end of the current line are ignored.

Typically, not both the access concentrator name and the service name are specified.

The ippp(4) or the pppoe(4) drivers require a number of additional arguments or optional parameters besides the settings that can be adjusted with ifconfig(8). These are things like authentication protocol parameters, but also other tunable configuration variables. The **pppoectl** utility can be used to display the current settings, or adjust these parameters as required.

For whatever intent **pppoectl** is being called, at least the parameter *ifname* needs to be specified, naming the interface for which the settings are to be performed or displayed. Use ifconfig(8) or netstat(1) to see which interfaces are available.

If no other parameter is given, **pppoectl** will just list the current settings for *ifname* and exit. The reported settings include the current PPP phase the interface is in, which can be one of the names *dead*, *establish*, *authenticate*, *network*, or *terminate*. If an authentication protocol is configured for the interface, the name of the protocol to be used, as well as the system name to be used or expected will be displayed, plus any possible options to the authentication protocol if applicable. Note that the authentication secrets (sometimes also called *keys*) are not being returned by the underlying system call, and are thus not displayed.

If any additional parameter is supplied, superuser privileges are required, and the command works in set mode. This is normally done quietly, unless the option **−v** is also enabled, which will cause a final printout of the settings as described above once all other actions have been taken. Use of this mode will be rejected if the interface is currently in any other phase than *dead*. Note that you can force an interface into *dead* phase by calling `ifconfig`(8) with the parameter `down`.

The currently supported parameters include:

`authproto`=*protoname*	Set both his and my authentication protocol to *protoname*. The protocol name can be one of `chap`, `pap`, or `none`. In the latter case, the use of an authentication protocol will be turned off for the named interface. This has the side-effect of clearing the other authentication-related parameters for this interface as well (i. e., system name and authentication secret will be forgotten).
`myauthproto`=*protoname*	Same as above, but only for my end of the link. I.e., this is the protocol when remote is authenticator, and I am the peer required to authenticate.
`hisauthproto`=*protoname*	Same as above, but only for his end of the link.
`myauthname`=*name*	Set my system name for the authentication protocol.
`hisauthname`=*name*	Set his system name for the authentication protocol. For CHAP, this will only be used as a hint, causing a warning message if remote did supply a different name. For PAP, it's the name remote must use to authenticate himself (in connection with his secret).
`myauthsecret`=*secret*	Set my secret (key, password) for use in the authentication phase. For CHAP, this will be used to compute the response hash value, based on remote's challenge. For PAP, it will be transmitted as plaintext together with the system name. Don't forget to quote the secrets from the shell if they contain shell metacharacters (or whitespace).
`myauthkey`=*secret*	Same as above.
`hisauthsecret`=*secret*	Same as above, to be used if we are authenticator and the remote peer needs to authenticate.
`hisauthkey`=*secret*	Same as above.
`callin`	Require remote to authenticate himself only when he's calling in, but not when we are caller. This is required for some peers that do not implement the authentication protocols symmetrically (like Ascend routers, for example).
`always`	The opposite of *`callin`*. Require remote to always authenticate, regardless of which side is placing the call. This is the default, and will not be explicitly displayed in `list` mode.
`norechallenge`	Only meaningful with CHAP. Do not re-challenge peer once the initial CHAP handshake was successful. Used to work around broken peer implementations that can't grok being re-challenged once the connection is up.
`rechallenge`	With CHAP, send re-challenges at random intervals while the connection is in network phase. (The intervals are currently in the range of 300 through approximately 800 seconds.) This is the default, and will not be explicitly displayed in `list` mode.

idle-timeout=idle-seconds
: For services that are charged by connection time the interface can optionally disconnect after a configured idle time. If set to 0, this feature is disabled. Note: for ISDN devices, it is preferable to use the `isdnd`(8) based timeout mechanism, as isdnd can predict the next charging unit for ISDN connections and optimize the timeout with this information.

lcp-timeout=timeout-value
: Allows to change the value of the LCP timeout. The default value of the LCP timeout is currently set to 1 second. The timeout-value must be specified in milliseconds.

max-noreceive=sec
: Sets the number of seconds after last reception of data from the peer before the line state is probed by sending LCP echo requests. The *sec* interval is not used verbatim, the first echo request might be delayed upto 10 seconds after the configured interval.

max-alive-missed=count
: Sets the number of unanswered LCP echo requests that we will tolerate before considering a connection to be dead. LCP echo requests are sent in 10 seconds interval after the configured *max-noreceive* interval has passed with no data received from the peer.

max-auth-failure=count
: Since some ISPs disable accounts after too many unsuccessful authentication attempts, there is a maximum number of authentication failures before we will stop retrying without manual intervention. Manual intervention is either changing the authentication data (name, password) or setting the maximum retry count. If *count* is set to *0* this feature is disabled.

clear-auth-failure
: If an authentication failure has been caused by remote problems and you want to retry connecting using unchanged local settings, this command can be used to reset the failure count to zero.

query-dns=flags
: During PPP protocol negotiation we can query the peer for addresses of two name servers. If *flags* is *1* only the first server address will be requested, if *flags* is *2* the second will be requested. Setting *flags* to *3* queries both.

  The result of the negotiation can be retrieved with the **-n** option.

EXAMPLES

```
# ipppctl ippp0
ippp0:  phase=dead
        myauthproto=chap myauthname="uriah"
        hisauthproto=chap hisauthname="ifb-gw" norechallenge
        lcp timeout: 3.000 s
```

Display the settings for ippp0. The interface is currently in *dead* phase, i.e. the LCP layer is down, and no traffic is possible. Both ends of the connection use the CHAP protocol, my end tells remote the system name `uriah`, and remote is expected to authenticate by the name `ifb-gw`. Once the initial CHAP handshake was successful, no further CHAP challenges will be transmitted. There are supposedly some known CHAP secrets for both ends of the link which are not being shown.

```
# ipppctl ippp0 \
        authproto=chap \
        myauthname=uriah myauthsecret='some secret' \
        hisauthname=ifb-gw hisauthsecret='another' \
        norechallenge
```

A possible call to **pppoectl** that could have been used to bring the interface into the state shown by the previous example.

The following example is the complete sequence of commands to bring a PPPoE connection up:

```
# Need ethernet interface UP (or it won't send any packets)
ifconfig ne0 up

# Let pppoe0 use ne0 as its ethernet interface
pppoectl -e ne0 pppoe0

# Configure authentication
pppoectl pppoe0 \
  myauthproto=pap \
  myauthname=XXXXX \
  myauthsecret=YYYYY \
  hisauthproto=none

# Configure the pppoe0 interface itself.  These addresses are magic,
# meaning we don't care about either address and let the remote
# ppp choose them.
ifconfig pppoe0 0.0.0.0 0.0.0.1 netmask 0xffffffff up
```

SEE ALSO

netstat(1), ippp(4), pppoe(4), ifconfig(8), ifwatchd(8)

B. Lloyd and W. Simpson, *PPP Authentication Protocols*, RFC 1334.

W. Simpson, Editor, *The Point-to-Point Protocol (PPP)*, RFC 1661.

W. Simpson, *PPP Challenge Handshake Authentication Protocol (CHAP)*, RFC 1994.

L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, and R. Wheeler, *A Method for Transmitting PPP Over Ethernet (PPPoE)*, RFC 2516.

HISTORY

The **pppoectl** utility is based on the **spppcontrol** utility which appeared in FreeBSD 3.0.

AUTHORS

The program was written by Jörg Wunsch, Dresden, and modified for PPPoE support by Martin Husemann.

NAME

pppstats, slstats − print PPP/SLIP statistics

SYNOPSIS

pppstats, slstats [**−a**] [**−v**] [**−r**] [**−z**] [**−c** <*count*>] [**−w** <*secs*>] [*interface*]

DESCRIPTION

The **pppstats** or **slstats** utility reports PPP or SLIP related statistics at regular intervals for the specified PPP/SLIP interface. If the interface is unspecified, **pppstats** will default to ppp0 and **slstats** will default to sl0. The display is split horizontally into input and output sections containing columns of statistics describing the properties and volume of packets received and transmitted by the interface.

The options are as follows:

−a Display absolute values rather than deltas. With this option, all reports show statistics for the time since the link was initiated. Without this option, the second and subsequent reports show statistics for the time since the last report.

−c *count*
Repeat the display *count* times. If this option is not specified, the default repeat count is 1 if the **−w** option is not specified, otherwise infinity.

−r Display additional statistics summarizing the compression ratio achieved by the packet compression algorithm in use.

−v Display additional statistics relating to the performance of the Van Jacobson TCP header compression algorithm.

−w *wait*
Pause *wait* seconds between each display. If this option is not specified, the default interval is 5 seconds.

−z Instead of the standard display, show statistics indicating the performance of the packet compression algorithm in use.

The following fields are printed on the input side when the **−z** option is not used:

IN The total number of bytes received by this interface.

PACK The total number of packets received by this interface.

VJCOMP
The number of header-compressed TCP packets received by this interface.

VJUNC
The number of header-uncompressed TCP packets received by this interface. Not reported when the **−r** option is specified.

VJERR
The number of corrupted or bogus header-compressed TCP packets received by this interface. Not reported when the **−r** option is specified.

VJTOSS
The number of VJ header-compressed TCP packets dropped on reception by this interface because of preceding errors. Only reported when the **−v** option is specified.

NON-VJ
The total number of non-TCP packets received by this interface. Only reported when the **−v** option is specified.

RATIO
The compression ratio achieved for received packets by the packet compression scheme in use, defined as the uncompressed size divided by the compressed size. Only reported when the **−r** option is specified.

UBYTE
The total number of bytes received, after decompression of compressed packets. Only reported when the **−r** option is specified.

The following fields are printed on the output side:

OUT The total number of bytes transmitted from this interface.

PACK The total number of packets transmitted from this interface.

VJCOMP
The number of TCP packets transmitted from this interface with VJ-compressed TCP headers.

VJUNC
The number of TCP packets transmitted from this interface with VJ-uncompressed TCP headers. Not reported when the **−r** option is specified.

NON-VJ
The total number of non-TCP packets transmitted from this interface. Not reported when the **−r** option is specified.

VJSRCH
The number of searches for the cached header entry for a VJ header compressed TCP packet. Only reported when the **−v** option is specified.

VJMISS
The number of failed searches for the cached header entry for a VJ header compressed TCP packet. Only reported when the **−v** option is specified.

RATIO
The compression ratio achieved for transmitted packets by the packet compression scheme in use, defined as the size before compression divided by the compressed size. Only reported when the **−r** option is specified.

UBYTE
The total number of bytes to be transmitted, before packet compression is applied. Only reported when the **−r** option is specified.

When the **−z** option is specified, **pppstats** instead displays the following fields, relating to the packet compression algorithm currently in use. This option is not supported by **slstats** and it always displays zeros. If packet compression is not in use, these fields will all display zeroes. The fields displayed on the input side are:

COMPRESSED BYTE
The number of bytes of compressed packets received.

COMPRESSED PACK
The number of compressed packets received.

INCOMPRESSIBLE BYTE
The number of bytes of incompressible packets (that is, those which were transmitted in uncompressed form) received.

INCOMPRESSIBLE PACK
The number of incompressible packets received.

COMP RATIO
The recent compression ratio for incoming packets, defined as the uncompressed size divided by the compressed size (including both compressible and incompressible packets).

The fields displayed on the output side are:

COMPRESSED BYTE
The number of bytes of compressed packets transmitted.

COMPRESSED PACK
The number of compressed packets transmitted.

INCOMPRESSIBLE BYTE
The number of bytes of incompressible packets transmitted (that is, those which were transmitted in uncompressed form).

INCOMPRESSIBLE PACK
The number of incompressible packets transmitted.

COMP RATIO
The recent compression ratio for outgoing packets.

SEE ALSO

pppd(8), slattach(8)

NAME

proxymap − Postfix lookup table proxy server

SYNOPSIS

proxymap [generic Postfix daemon options]

DESCRIPTION

The **proxymap**(8) server provides read-only table lookup service to Postfix processes. The purpose of the service is:

- To overcome chroot restrictions. For example, a chrooted SMTP server needs access to the system passwd file in order to reject mail for non-existent local addresses, but it is not practical to maintain a copy of the passwd file in the chroot jail. The solution:

  ```
  local_recipient_maps =
     proxy:unix:passwd.byname $alias_maps
  ```

- To consolidate the number of open lookup tables by sharing one open table among multiple processes. For example, making mysql connections from every Postfix daemon process results in "too many connections" errors. The solution:

  ```
  virtual_alias_maps =
     proxy:mysql:/etc/postfix/virtual_alias.cf
  ```

  The total number of connections is limited by the number of proxymap server processes.

The **proxymap**(8) server implements the following requests:

open *maptype:mapname flags*
: Open the table with type *maptype* and name *mapname*, as controlled by *flags*. The reply includes the *maptype* dependent flags (to distinguish a fixed string table from a regular expression table).

lookup *maptype:mapname flags key*
: Look up the data stored under the requested key. The reply is the request completion status code (below) and the lookup result value. The *maptype:mapname* and *flags* are the same as with the **open** request.

There is no **close** command, nor are tables implicitly closed when a client disconnects. The purpose is to share tables among multiple client processes.

SERVER PROCESS MANAGEMENT

proxymap(8) servers run under control by the Postfix **master**(8) server. Each server can handle multiple simultaneous connections. When all servers are busy while a client connects, the **master**(8) creates a new **proxymap**(8) server process, provided that the process limit is not exceeded. Each server terminates after serving at least **$max_use** clients or after **$max_idle** seconds of idle time.

SECURITY

The **proxymap**(8) server opens only tables that are approved via the **proxy_read_maps** configuration parameter, does not talk to users, and can run at fixed low privilege, chrooted or not. However, running the proxymap server chrooted severely limits usability, because it can open only chrooted tables.

The **proxymap**(8) server is not a trusted daemon process, and must not be used to look up sensitive information such as user or group IDs, mailbox file/directory names or external commands.

In Postfix version 2.2 and later, the proxymap client recognizes requests to access a table for security-sensitive purposes, and opens the table directly. This allows the same main.cf setting to be used by sensitive and non-sensitive processes.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

The **proxymap**(8) server provides service to multiple clients, and must therefore not be used for tables that have high-latency lookups.

CONFIGURATION PARAMETERS

On busy mail systems a long time may pass before **proxymap**(8) relevant changes to **main.cf** are picked up. Use the command "**postfix reload**" to speed up a change.

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.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

proxy_read_maps (see 'postconf -d' output)
The lookup tables that the **proxymap**(8) server is allowed to access.

SEE ALSO

postconf(5), configuration parameters
master(5), generic daemon options

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.

HISTORY

The proxymap service was introduced with Postfix 2.0.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

pstat — display system data structures

SYNOPSIS

pstat [**−T** | **−f** | **−s** | **−t** | **−v**] [**−ghkmn**] [**−M** *core*] [**−N** *system*]

DESCRIPTION

pstat displays open file entry, swap space utilization, terminal state, and vnode data structures. If *corefile* is given, the information is sought there, otherwise in `/dev/kmem`. The required namelist is taken from `/netbsd` unless *system* is specified.

The following options are available:

−T Prints the number of used and free slots for open files, used vnodes, and swap space. This option is useful for checking to see how large system tables become if the system is under heavy load.

−f Print the open file table with these headings:

LOC	The core location of this table entry.
TYPE	The type of object the file table entry points to.
FLG	Miscellaneous state variables encoded thus:

R	open for reading
W	open for writing
A	open for appending
S	shared lock present
X	exclusive lock present
I	signal pgrp when data ready

CNT	Number of processes that know this open file.
MSG	Number of messages outstanding for this file.
DATA	The location of the vnode table entry or socket structure for this file.
USE	Number of active users of this open file.
IFLG	Value of internal flags.
OFFSET	The file offset (see `lseek`(2)).

−g The **−g** option uses (1024 ∗ 1024 ∗ 1024) byte blocks instead of the default 512 byte.

−h Use `humanize_number`(3) to display (swap) sizes.

−k Use 1K-byte blocks.

−m The **−m** option uses (1024 ∗ 1024) byte blocks instead of the default 512 byte.

−n Print devices by major/minor number rather than by name.

−s Print information about swap space usage on all the swap areas compiled into the kernel. 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 *Available* column indicates how much space is remaining on each partition. The *Capacity* reports the percentage of space used.

If more than one partition is configured into the system, totals for all of the statistics will be reported in the final line of the report.

-t Print table for terminals with these headings:

LINE	Physical device name.
RAW	Number of characters in raw input queue.
CAN	Number of characters in canonicalized input queue.
OUT	Number of characters in output queue.
HWT	High water mark for output.
LWT	Low water mark for output.
COL	Calculated column position of terminal.
STATE	Miscellaneous state variables encoded thus:

T	delay timeout in progress
O	open
F	outq has been flushed during DMA
C	carrier is on
B	busy doing output
A	process is awaiting output
X	open for exclusive use
S	output stopped
K	further input blocked
Y	tty in async I/O mode
D	state for lowercase '`\`' work
E	within a `\.../` for PRTRUB
L	next character is literal
P	retyping suspended input (PENDIN)
N	counting tab width, ignore FLUSHO
>	tty used for dialout

SESS	Session for which this is controlling terminal.
PGID	Current foreground process group associated with this terminal.
DISC	Line discipline; `term` for TTYDISC (see `termios`(4)), `tab` for TABLDISC (see `tb`(4)), `slip` for SLIPDISC (see `sl`(4)), `ppp` for PPPDISC (see `ppp`(4)), `strip` for STRIPDISC (see `strip`(4)), `hdlc` for HDLCDISC.

-v Print the active vnodes. Each group of vnodes corresponding to a particular filesystem is preceded by a two line header. The first line consists of the following:

∗∗∗ MOUNT *fstype from* on *on fsflags*

where *fstype* is one of *adosfs*, *afs*, *cd9660*, *fdesc*, *ffs*, *ext2fs*, *kernfs*, *lfs*, *lofs*, *mfs*, *msdos*, *nfs*, *null*, *portal*, *procfs*, *umap*, *union*; *from* is the filesystem mounted from; *on* is the directory the filesystem is mounted on; and *fsflags* is a list of optional flags applied to the mount (see `mount`(8)). The second line is a header for the individual fields, the first part of which are fixed, and the second part are filesystem type specific. The headers common to all vnodes are:

ADDR Location of this vnode.

TYP File type.

VFLAG A list of letters representing vnode flags:

R	VROOT root of its file system.
T	VTEXT pure text prototype.
S	VSYSTEM vnode being used by kernel.
I	VISTTY vnode is a tty.
E	VEXECMAP vnode has PROT_EXEC mappings.
L	VXLOCK locked to change underlying type.
W	VXWANT process is waiting for vnode.
B	VBWAIT waiting for output to complete.
A	VALIASED vnode has an alias.
D	VDIROP lfs vnode involved in directory op.
Y	VLAYER vnode is on layer filesystem.
O	VONWORKLST vnode is on syncer work-list.

USE The number of references to this vnode.

HOLD The number of I/O buffers held by this vnode.

TAG The type of underlying data.

NPAGE The number of pages in this vnode.

FILEID The vnode fileid. In the case of *ffs* or *ext2fs* this is the inode number.

IFLAG Miscellaneous filesystem specific state variables encoded thus:

For ffs, lfs or ext2fs:

A	access time must be corrected
C	changed time must be corrected
U	update time (`fs`(5)) must be corrected
M	contains modifications
a	has been accessed
R	has a rename in progress
S	shared lock applied
E	exclusive lock applied
c	is being cleaned (LFS)
D	directory operation in progress (LFS)
s	blocks to be freed in free count

For nfs:

W	waiting for I/O buffer flush to complete
P	I/O buffers being flushed
M	locally modified data exists
E	an earlier write failed
A	special file accessed
U	special file updated
C	special file times changed

SIZ/RDEV
Number of bytes in an ordinary file, or major and minor device of special file.

ENVIRONMENT

BLOCKSIZE	If the environment variable BLOCKSIZE is set, and the **-k** option is not specified, the block counts will be displayed in units of that size block.

FILES

/netbsd	namelist
/dev/kmem	default source of tables

SEE ALSO

ps(1), systat(1), vmstat(1), stat(2), fs(5), iostat(8)

HISTORY

The **pstat** command appeared in 4.0BSD.

BUGS

Swap statistics are reported for all swap partitions compiled into the kernel, regardless of whether those partitions are being used.

Does not understand NFS swap servers.

NAME

push — fetch mail via POP

SYNOPSIS

push [**−4** | **−-krb4**] [**−5** | **−-krb5**] [**−v** | **−-verbose**] [**−f** | **−-fork**] [**−l** | **−-leave**] [**−-from**] [**−c** | **−-count**] [**−-headers**=*headers*] [**−p** *port-spec* | **−-port**=*port-spec*] *po-box* `filename`

DESCRIPTION

push retrieves mail from the post office box *po-box*, and stores the mail in mbox format in `filename`. The *po-box* can have any of the following formats:

```
hostname:username
po:hostname:username
username@hostname
po:username@hostname
hostname
po:username
```

If no username is specified, **push** assumes that it's the same as on the local machine; *hostname* defaults to the value of the `MAILHOST` environment variable.

Supported options:

−4, **−-krb4**
use Kerberos 4 (if compiled with support for Kerberos 4)

−5, **−-krb5**
use Kerberos 5 (if compiled with support for Kerberos 5)

−f, **−-fork**
fork before starting to delete messages

−l, **−-leave**
don't delete fetched mail

−-from
behave like from.

−c, **−-count**
first print how many messages and bytes there are.

−-headers=*headers*
a list of comma-separated headers that should get printed.

−p *port-spec*, **−-port**=*port-spec*
use this port instead of the default `kpop` or `1109`.

The default is to first try Kerberos 5 authentication and then, if that fails, Kerberos 4.

ENVIRONMENT

`MAILHOST`
points to the post office, if no other hostname is specified.

EXAMPLES

```
$ push cornfield:roosta ~/.emacs-mail-crash-box
```

tries to fetch mail for the user *roosta* from the post office at “cornfield”, and stores the mail in `~/.emacs-mail-crash-box` (you are using Gnus, aren't you?)

```
      $ push --from -5 havregryn
```

tries to fetch **From:** lines for current user at post office “havregryn” using Kerberos 5.

SEE ALSO

`from`(1), `pfrom`(1), `movemail`(8), `popper`(8)

HISTORY

push was written while waiting for **movemail** to finish getting the mail.

NAME

pvcsif — configure ATM PVC sub interfaces

SYNOPSIS

pvcsif *interface* [**-s**]
pvcsif -a

DESCRIPTION

pvcsif creates a sub interface for an ATM PVC. A sub interface pvc(4) is dynamically created. The created interface is bound to *interface* but at this point no VC is assigned. To assign a VC, pvctxctl(8) should be used later.

A PVC sub interface is intended to use an ATM PVC as an alternative serial connection, and to be allocated per PVC basis. A PVC sub interface looks as a point-to-point interface and is multicast capable, as opposed to the NBMA (NonBroadcast Multiple Access) model that requires a MARS server. A point-to-point interface is useful to run MBone or protocols requiring multicast, such as RSVP and IPv6, over a PVC WAN connection.

Note that a sub interface is not a full-fledged interface but just an indirect reference to the real interface.

The options are as follows:

-s	For use with a shell, it prints the created interface name.
-a	Lists the existing sub interfaces.

SEE ALSO

en(4), pvc(4), ifconfig(8), pvctxctl(8)

BUGS

Currently, there is no way to remove a sub interface.

NAME

pvctxctl — display or control ATM PVC transmitter parameters

SYNOPSIS

pvctxctl *interface* [[vpi:]*vci*]
pvctxctl *interface* [vpi:]*vci* [**-n**] [**-b** *max-bandwidth*] [**-j** [vpi:]*vci*] [**-p** *pcr*]

DESCRIPTION

pvctxctl displays or controls the shaper parameters of an ATM VC. When a shaper value is specified, **pvctxctl** sets a shaper to an ATM VC. **pvctxctl** works for a PVC sub interface pvc(4) as well as a real ATM interface en(4).

For a real ATM interface, the specified VC should be assigned beforehand by route(8).

For a sub interface, **pvctxctl** assigns the specified VC to the sub interface. If another VC is already assigned to the sub interface, the old VC is invalidated.

Availability of shapers, the number of hardware shaper channels, and accuracy of shaping are all device dependent. For example, ENI Midway chip has 8 shaper channels but the driver reserves one for non-shaping VCs.

The options are as follows:

interface
The *interface* parameter is a string of the form "name unit", for example, "en0".

[vpi:]*vci* The VC number to which the shaper is assigned. When the VPI number is omitted, VPI number 0 is assumed. For example, to assign a shaper to VPI=0 and VCI=201, the following forms can be used: "201", "0xc9", "0:201", "0:0xc9".

-b *max-bandwidth*
The PCR parameter can be specified also in "bits per second". The rate is the rate of AAL5 frame and the PCR is calculated by the following form:

```
PCR = max-bandwidth / 8 / 48
```

"K" and "M" can be used as a short hand of "000" and "000000" respectively. For example, "45M" means "45Mbps" or PCR value "117187".

-j [vpi:]*vci*
The join parameter is intended for VP shaping. The VC shaper channel is shared with the existing VC, which means the sum of the cell rates never exceeds the maximum PCR among the shared VCs. On the other hand, when the shaper channel is not shared, the sum of the cell rates could be the sum of the PCRs.

For example, if two VCs (say 201 and 202) share a 45Mbps VP, use:

```
# pvctxctl en0 201 -b 45M
# pvctxctl en0 202 -b 45M -j 201
```

-n This parameter is only for a sub interface. Use NULL encapsulation instead of LLC/SNAP.

-p *pcr* The PCR (Peak Cell Rate) parameter specifies the peak cell rate in "cells per second". If PCR value "0" is specified, no shaper is assigned, which means cells are sent at full-speed of the link. If PCR value "-1" is specified, the corresponding VC is invalidated.

SEE ALSO

`en`(4), `ifconfig`(8), `pvcsif`(8), `route`(8)

BUGS

A real ATM interface and a sub interface require different sequences to set a shaper. For example, to assign a 45Mbps shaper to VC 201 (0xc9) of en0:

```
# ifconfig en0 10.0.0.1
# route add -iface 10.0.0.2 -link en0:3.0.0.c9
# pvctxctl en0 0xc9 -b 45M
```

For a shadow interface,

```
# pvcsif en0                          # creates pvc0
# ifconfig pvc0 10.0.0.1 10.0.0.2
# pvctxctl pvc0 201 -b 45M
```

NAME

pwd_mkdb — generate the password databases

SYNOPSIS

pwd_mkdb [**−BLps**] [**−c** *cachesize*] [**−d** *directory*] [**−u** *username*] *file*

DESCRIPTION

pwd_mkdb creates `db`(3) style secure and insecure databases for the specified file. These databases are then installed into "`/etc/spwd.db`" and "`/etc/pwd.db`" respectively. The file is installed into "`/etc/master.passwd`". The file must be in the correct format (see `passwd`(5)). It is important to note that the format used in this system is different from the historic Version 7 style format.

The options are as follows:

−B Store data in big-endian format (see also **−L**).

−c *cachesize*
Specify the size of the memory cache in megabytes used by the hashing library. On systems with a large user base, a small cache size can lead to prohibitively long database file rebuild times. As a rough guide, the memory usage of **pwd_mkdb** in megabytes will be a little bit more than twice the figure specified here. If unspecified, this value will be calculated based on the size of the input file up to a maximum of 8 megabytes.

−d *directory*
Change the root directory of the generated files from "`/`" to *directory*.

−L Store data in little-endian format (see also **−B**).

−p Create a Version 7 style password file and install it into "`/etc/passwd`".

−s Update the secure database only. This is useful when only encrypted passwords have changed. This option negates the effect of any **−p** option.

−u *name*
Don't re-build the database files, but instead modify or add entries for the specified user only. This option may only be used when the line number and user name in the password file have not changed, or when adding a new user from the last line in the password file.

The two databases differ in that the secure version contains the user's encrypted password and the insecure version has an asterisk ("*").

The databases are used by the C library password routines (see `getpwent`(3)).

EXIT STATUS

pwd_mkdb exits zero on success, non-zero on failure.

FILES

`/etc/master.passwd`	The current password file.
`/etc/passwd`	A Version 7 format password file.
`/etc/pwd.db`	The insecure password database file.
`/etc/pwd.db.tmp`	A temporary file.
`/etc/spwd.db`	The secure password database file.
`/etc/spwd.db.tmp`	A temporary file.

BUGS

Because of the necessity for atomic update of the password files, **pwd_mkdb** uses `rename`(2) to install them. This, however, requires that the file specified on the command line live on the same file system as the "`/etc`" directory.

There are the obvious races with multiple people running **pwd_mkdb** on different password files at the same time. The front-ends to `chpass`(1), `passwd`(1), `useradd`(8), `userdel`(8), `usermod`(8), and `vipw`(8) handle the locking necessary to avoid this problem.

The database files are copied when the **-u** option is used. Real locking would make this unnecessary.

Although the DB format is endian-transparent, the data stored in the DB is not. Also, the format doesn't lend itself to insertion or removal of records from arbitrary locations in the password file. This is difficult to fix without breaking compatibility.

Using the **-u** option on a system where multiple users share the same UID can have unexpected results.

COMPATIBILITY

Previous versions of the system had a program similar to **pwd_mkdb** which built *dbm* style databases for the password file but depended on the calling programs to install them. The program was renamed in order that previous users of the program not be surprised by the changes in functionality.

SEE ALSO

`chpass`(1), `passwd`(1), `pwhash`(1), `db`(3), `getpwent`(3), `pw_mkdb`(3), `passwd`(5), `useradd`(8), `userdel`(8), `usermod`(8), `vipw`(8)

NAME

pxeboot — network boot NetBSD/i386 through a PXE BIOS extension

DESCRIPTION

pxeboot is a NetBSD boot program running on top of a PXE BIOS extension which is provided by the motherboard or a plug-in network adapter, in accordance with the Intel Preboot eXecution Environement (PXE) specification.

Network booting a system through PXE is a two-stage process:

1. The PXE BIOS issues a DHCP request and fetches the NetBSD **pxeboot** program using TFTP.
2. The NetBSD **pxeboot** program takes control. An interactive mode is entered if the user presses a key within five seconds. After this time or after the user's **boot** command, another DHCP request is issued and the kernel is loaded as specified in the DHCP reply. To read the kernel file, the NFS (version 2) or TFTP protocols can be used.

The DHCP request issued by the NetBSD **pxeboot** program has the following special parameters:

Bootfile name
: is set to the *filename* argument on the **boot** command line typed in by the user (can be empty), or to "netbsd" in the non-interactive case.

DHCP Vendor class identifier tag
: is set to "NetBSD:i386:libsa".

The DHCP server can use these fields to distinguish between the various originators of requests (first and second PXE stage, NetBSD kernel), and to control conditional behaviour depending on the user's command line input to the **pxeboot** program, e.g. to support alternative NetBSD installations on one machine.

In addition to the standard network interface configuration, the following fields in the DHCP reply are interpreted:

Bootfile name
: specifies the protocol to be used, and the filename of the NetBSD kernel to be booted, separated by a colon. Available protocols are "nfs" and "tftp". The kernel filename part is interpreted relatively to the NFS root directory (see the *Root path* reply field below) or the TFTP server's root directory (which might be a subdirectory within the TFTP server's filesystem, depending on the implementation), respectively. If the *Bootfile name* field replied by the DHCP server does not contain a colon, it is ignored, and the *filename* typed in at the **pxeboot** command line prompt (or the "netbsd" default, see the section about the *Bootfile name* field in the DHCP request above) is used. If no protocol was specified, "nfs" is assumed.

Swap server
: can be used to override the "server IP address" if NFS is used to access the kernel. This matches the behaviour of the NetBSD kernel to access its root file system on NFS. This way, different TFTP and NFS servers can be communicated to the DHCP client (it is actually a deficiency of the DHCP protocol to provide a "root path" field but no corresponding IP address) .

Root path
: is used as path to be mounted in the NFS case to access the kernel file, matching the NetBSD kernel's behaviour.

The commands accepted in interactive mode are:

boot [*device*:][*filename*] [**-acdqsvxz**]
: Boot NetBSD. See **boot** in boot(8) for full details.

help
Print an overview about commands and arguments.

quit
Leave the **pxeboot** program.

By default the output from **pxeboot** and from the booted kernel will go to the system's BIOS console. This can be changed to be one of the serial ports by using **installboot** to modify the boot options contained in the `pxeboot_ia32.bin` file.

FILES

`/usr/mdec/pxeboot_ia32.bin`

EXAMPLES

The first `/etc/dhcpd.conf` example shows a simple configuration which just loads a file "netbsd" from the client's NFS root directory, using the defaults for protocol and kernel filename. Similar setups should be possible with any BOOTP/DHCP server.

```
host myhost {
    hardware ethernet 00:00:00:00:00:00;
    fixed-address myhost;
    option host-name "myhost";
    filename "pxeboot_ia32.bin";
    option swap-server mynfsserver;
    option root-path "/export/myhost";
}
```

The following `/etc/dhcpd.conf` entry shows how different system installations can be booted depending on the user's input on the **pxeboot** command line.

```
host myhost {
    hardware ethernet 00:00:00:00:00:00;
    fixed-address myhost;
    option host-name "myhost";
    if substring (option vendor-class-identifier, 0, 9) = "PXEClient" {
        filename "pxeboot_ia32.bin";
    } elsif filename = "tftp" {
        filename "tftp:netbsd.myhost";
    } else {
        option swap-server mynfsserver;
        option root-path "/export/myhost";
        if filename = "generic" {
            filename "nfs:gennetbsd";
        } else {
            filename "nfs:netbsd";
        }
    }
}
```

It is assumed that the TFTP server is the same as the DHCP server unless a *next-server* directive is specified somewhere else in `dhcpd.conf`, and that the NFS server for the root file system is *mynfsserver*. The *swap-server:root-path* is only used in the NFS case and by the NetBSD kernel to mount the root file system.

SEE ALSO

`boot`(8), `dhcpd`(8), `diskless`(8), `installboot`(8)

Intel Corporation, *Preboot Execution Environment (PXE) Specification*, Version 2.1, September 20, 1999.

HISTORY

The NetBSD/i386 **`pxeboot`** command first appeared in NetBSD 1.6.

BUGS

If an error is encountered while reading the NetBSD kernel file or if its file format wasn't recognized, it is impossible to retry the operation because the PXE network stack is already removed from the system RAM.

You need the **`pxeboot`** from an i386 build to boot an i386 kernel, and that from an amd64 build to boot an amd64 kernel.

NAME

qmgr − Postfix queue manager

SYNOPSIS

qmgr [generic Postfix daemon options]

DESCRIPTION

The **qmgr**(8) daemon awaits the arrival of incoming mail and arranges for its delivery via Postfix delivery processes. The actual mail routing strategy is delegated to the **trivial-rewrite**(8) daemon. This program expects to be run from the **master**(8) process manager.

Mail addressed to the local **double-bounce** address is logged and discarded. This stops potential loops caused by undeliverable bounce notifications.

MAIL QUEUES

The **qmgr**(8) daemon maintains the following queues:

incoming
: Inbound mail from the network, or mail picked up by the local **pickup**(8) daemon from the **maildrop** directory.

active
: Messages that the queue manager has opened for delivery. Only a limited number of messages is allowed to enter the **active** queue (leaky bucket strategy, for a fixed delivery rate).

deferred
: Mail that could not be delivered upon the first attempt. The queue manager implements exponential backoff by doubling the time between delivery attempts.

corrupt
: Unreadable or damaged queue files are moved here for inspection.

hold
: Messages that are kept "on hold" are kept here until someone sets them free.

DELIVERY STATUS REPORTS

The **qmgr**(8) daemon keeps an eye on per-message delivery status reports in the following directories. Each status report file has the same name as the corresponding message file:

bounce
: Per-recipient status information about why mail is bounced. These files are maintained by the **bounce**(8) daemon.

defer
: Per-recipient status information about why mail is delayed. These files are maintained by the **defer**(8) daemon.

trace
: Per-recipient status information as requested with the Postfix "**sendmail -v**" or "**sendmail -bv**" command. These files are maintained by the **trace**(8) daemon.

The **qmgr**(8) daemon is responsible for asking the **bounce**(8), **defer**(8) or **trace**(8) daemons to send delivery reports.

STRATEGIES

The queue manager implements a variety of strategies for either opening queue files (input) or for message delivery (output).

leaky bucket
: This strategy limits the number of messages in the **active** queue and prevents the queue manager from running out of memory under heavy load.

fairness
: When the **active** queue has room, the queue manager takes one message from the **incoming** queue and one from the **deferred** queue. This prevents a large mail backlog from blocking the delivery of new mail.

slow start
: This strategy eliminates "thundering herd" problems by slowly adjusting the number of parallel deliveries to the same destination.

round robin
The queue manager sorts delivery requests by destination. Round-robin selection prevents one destination from dominating deliveries to other destinations.

exponential backoff
Mail that cannot be delivered upon the first attempt is deferred. The time interval between delivery attempts is doubled after each attempt.

destination status cache
The queue manager avoids unnecessary delivery attempts by maintaining a short-term, in-memory list of unreachable destinations.

preemptive message scheduling
The queue manager attempts to minimize the average per-recipient delay while still preserving the correct per-message delays, using a sophisticated preemptive message scheduling.

TRIGGERS

On an idle system, the queue manager waits for the arrival of trigger events, or it waits for a timer to go off. A trigger is a one-byte message. Depending on the message received, the queue manager performs one of the following actions (the message is followed by the symbolic constant used internally by the software):

D (QMGR_REQ_SCAN_DEFERRED)
Start a deferred queue scan. If a deferred queue scan is already in progress, that scan will be restarted as soon as it finishes.

I (QMGR_REQ_SCAN_INCOMING)
Start an incoming queue scan. If an incoming queue scan is already in progress, that scan will be restarted as soon as it finishes.

A (QMGR_REQ_SCAN_ALL)
Ignore deferred queue file time stamps. The request affects the next deferred queue scan.

F (QMGR_REQ_FLUSH_DEAD)
Purge all information about dead transports and destinations.

W (TRIGGER_REQ_WAKEUP)
Wakeup call, This is used by the master server to instantiate servers that should not go away forever. The action is to start an incoming queue scan.

The **qmgr**(8) daemon reads an entire buffer worth of triggers. Multiple identical trigger requests are collapsed into one, and trigger requests are sorted so that **A** and **F** precede **D** and **I**. Thus, in order to force a deferred queue run, one would request **A F D**; in order to notify the queue manager of the arrival of new mail one would request **I**.

STANDARDS

RFC 3463 (Enhanced status codes)
RFC 3464 (Delivery status notifications)

SECURITY

The **qmgr**(8) daemon is not security sensitive. It reads single-character messages from untrusted local users, and thus may be susceptible to denial of service attacks. The **qmgr**(8) daemon does not talk to the outside world, and it can be run at fixed low privilege in a chrooted environment.

DIAGNOSTICS

Problems and transactions are logged to the syslog daemon. Corrupted message files are saved to the **corrupt** queue for further inspection.

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces and of other trouble.

BUGS

A single queue manager process has to compete for disk access with multiple front-end processes such as **cleanup**(8). A sudden burst of inbound mail can negatively impact outbound delivery rates.

CONFIGURATION PARAMETERS

Changes to **main.cf** are not picked up automatically as **qmgr**(8) is a persistent process. Use the "**postfix reload**" command after a configuration change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

In the text below, *transport* is the first field in a **master.cf** entry.

COMPATIBILITY CONTROLS

allow_min_user (no)
Allow a recipient address to have '-' as the first character.

ACTIVE QUEUE CONTROLS

qmgr_clog_warn_time (300s)
The minimal delay between warnings that a specific destination is clogging up the Postfix active queue.

qmgr_message_active_limit (20000)
The maximal number of messages in the active queue.

qmgr_message_recipient_limit (20000)
The maximal number of recipients held in memory by the Postfix queue manager, and the maximal size of the size of the short-term, in-memory "dead" destination status cache.

qmgr_message_recipient_minimum (10)
The minimal number of in-memory recipients for any message.

default_recipient_limit (20000)
The default per-transport upper limit on the number of in-memory recipients.

*transport***_recipient_limit ($default_recipient_limit)**
Idem, for delivery via the named message *transport*.

default_extra_recipient_limit (1000)
The default value for the extra per-transport limit imposed on the number of in-memory recipients.

*transport***_extra_recipient_limit ($default_extra_recipient_limit)**
Idem, for delivery via the named message *transport*.

Available in Postfix version 2.4 and later:

default_recipient_refill_limit (100)
The default per-transport limit on the number of recipients refilled at once.

*transport***_recipient_refill_limit ($default_recipient_refill_limit)**
Idem, for delivery via the named message *transport*.

default_recipient_refill_delay (5s)
The default per-transport maximum delay between recipients refills.

*transport***_recipient_refill_delay ($default_recipient_refill_delay)**
Idem, for delivery via the named message *transport*.

DELIVERY CONCURRENCY CONTROLS

initial_destination_concurrency (5)
The initial per-destination concurrency level for parallel delivery to the same destination.

default_destination_concurrency_limit (20)
The default maximal number of parallel deliveries to the same destination.

*transport***_destination_concurrency_limit ($default_destination_concurrency_limit)**
Idem, for delivery via the named message *transport*.

RECIPIENT SCHEDULING CONTROLS

default_destination_recipient_limit (50)
The default maximal number of recipients per message delivery.

*transport*_**destination_recipient_limit ($default_destination_recipient_limit)**
Idem, for delivery via the named message *transport*.

MESSAGE SCHEDULING CONTROLS

default_delivery_slot_cost (5)
How often the Postfix queue manager's scheduler is allowed to preempt delivery of one message with another.

*transport*_**delivery_slot_cost ($default_delivery_slot_cost)**
Idem, for delivery via the named message *transport*.

default_minimum_delivery_slots (3)
How many recipients a message must have in order to invoke the Postfix queue manager's scheduling algorithm at all.

*transport*_**minimum_delivery_slots ($default_minimum_delivery_slots)**
Idem, for delivery via the named message *transport*.

default_delivery_slot_discount (50)
The default value for transport-specific _delivery_slot_discount settings.

*transport*_**delivery_slot_discount ($default_delivery_slot_discount)**
Idem, for delivery via the named message *transport*.

default_delivery_slot_loan (3)
The default value for transport-specific _delivery_slot_loan settings.

*transport*_**delivery_slot_loan ($default_delivery_slot_loan)**
Idem, for delivery via the named message *transport*.

OTHER RESOURCE AND RATE CONTROLS

minimal_backoff_time (version dependent)
The minimal time between attempts to deliver a deferred message.

maximal_backoff_time (4000s)
The maximal time between attempts to deliver a deferred message.

maximal_queue_lifetime (5d)
The maximal time a message is queued before it is sent back as undeliverable.

queue_run_delay (version dependent)
The time between deferred queue scans by the queue manager.

transport_retry_time (60s)
The time between attempts by the Postfix queue manager to contact a malfunctioning message delivery transport.

Available in Postfix version 2.1 and later:

bounce_queue_lifetime (5d)
The maximal time a bounce message is queued before it is considered undeliverable.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

defer_transports (empty)
The names of message delivery transports that should not deliver mail unless someone issues "**sendmail -q**" or equivalent.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

helpful_warnings (yes)
Log warnings about problematic configuration settings, and provide helpful suggestions.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

FILES

/var/spool/postfix/incoming, incoming queue
/var/spool/postfix/active, active queue
/var/spool/postfix/deferred, deferred queue
/var/spool/postfix/bounce, non-delivery status
/var/spool/postfix/defer, non-delivery status
/var/spool/postfix/trace, delivery status

SEE ALSO

trivial-rewrite(8), address routing
bounce(8), delivery status reports
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
SCHEDULER_README, scheduling algorithm
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

Scheduler enhancements:
Patrik Rak
Modra 6

155 00, Prague, Czech Republic

NAME

qmqpd − Postfix QMQP server

SYNOPSIS

qmqpd [generic Postfix daemon options]

DESCRIPTION

The Postfix QMQP server receives one message per connection. Each message is piped through the **cleanup**(8) daemon, and is placed into the **incoming** queue as one single queue file. The program expects to be run from the **master**(8) process manager.

The QMQP server implements one access policy: only explicitly authorized client hosts are allowed to use the service.

SECURITY

The QMQP server is moderately security-sensitive. It talks to QMQP clients and to DNS servers on the network. The QMQP server can be run chrooted at fixed low privilege.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

The QMQP protocol provides only one server reply per message delivery. It is therefore not possible to reject individual recipients.

The QMQP protocol requires the server to receive the entire message before replying. If a message is malformed, or if any netstring component is longer than acceptable, Postfix replies immediately and closes the connection. It is left up to the client to handle the situation.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically, as **qmqpd**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

CONTENT INSPECTION CONTROLS

content_filter (empty)
The name of a mail delivery transport that filters mail after it is queued.

receive_override_options (empty)
Enable or disable recipient validation, built-in content filtering, or address mapping.

RESOURCE AND RATE CONTROLS

line_length_limit (2048)
Upon input, long lines are chopped up into pieces of at most this length; upon delivery, long lines are reconstructed.

hopcount_limit (50)
The maximal number of Received: message headers that is allowed in the primary message headers.

message_size_limit (10240000)
The maximal size in bytes of a message, including envelope information.

qmqpd_timeout (300s)
The time limit for sending or receiving information over the network.

TROUBLE SHOOTING CONTROLS

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.

soft_bounce (no)
Safety net to keep mail queued that would otherwise be returned to the sender.

TARPIT CONTROLS

qmqpd_error_delay (1s)
How long the QMQP server will pause before sending a negative reply to the client.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

qmqpd_authorized_clients (empty)
What clients are allowed to connect to the QMQP server port.

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".

verp_delimiter_filter (-=+)
The characters Postfix accepts as VERP delimiter characters on the Postfix **sendmail**(1) command line and in SMTP commands.

SEE ALSO

http://cr.yp.to/proto/qmqp.html, QMQP protocol
cleanup(8), message canonicalization
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
QMQP_README, Postfix ezmlm-idx howto.

LICENSE

The Secure Mailer license must be distributed with this software.

HISTORY

The qmqpd service 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

quot — display disk space occupied by each user

SYNOPSIS

quot [**−acfhknv**] [*filesystem ...*]

DESCRIPTION

quot is used to gather statistics about the disk usage for each local user.

The following options are available:

−a Include statistics for all mounted filesystems.

−c Display three columns containing number of blocks per file, number of files in this category, and aggregate total of blocks in files with this or lower size.

−f For each user, display count of files and space occupied.

−h Estimate the number of blocks in each file based on its size. Despite that this doesn't give the correct results (it doesn't account for the holes in files), this option isn't any faster and thus is discouraged.

−k By default, all sizes are reported in 512-byte block counts. The **−k** options causes the numbers to be reported in kilobyte counts.

−n Given a list of inodes (plus some optional data on each line) in the standard input, for each file print out the owner (plus the remainder of the input line). This is traditionally used in the pipe:

```
ncheck filesystem | sort +0n | quot -n filesystem
```

to get a report of files and their owners.

−v In addition to the default output, display the number of files not accessed within 30, 60 and 90 days.

ENVIRONMENT

`BLOCKSIZE` If the environment variable `BLOCKSIZE` is set, and the option is not specified, the block counts will be displayed in units of that size block.

SEE ALSO

df(1), quota(1), getbsize(3), getmntinfo(3), fstab(5), mount(8)

NAME

quotacheck — filesystem quota consistency checker

SYNOPSIS

quotacheck [**−gquv**] *filesystem ...*
quotacheck [**−gquv**] [**−l** *maxparallel*] **−a**

DESCRIPTION

quotacheck examines each filesystem, builds a table of current disk usage, and compares this table against that recorded in the disk quota file for the filesystem. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated (the latter only occurs if an active filesystem is checked). By default both user and group quotas are checked.

Available options:

−a If the **−a** flag is supplied in place of any filesystem names, **quotacheck** will check all the filesystems indicated in `/etc/fstab` to be read-write with disk quotas. By default only the types of quotas listed in `/etc/fstab` are checked. See also **−l**.

−g Only group quotas listed in `/etc/fstab` are to be checked. See also **−u**.

−l *maxparallel*
Limit the number of parallel checks to the number specified in the following argument. By default, the limit is the number of disks, running one process per disk. If a smaller limit is given, the disks are checked round-robin, one file system at a time. This option is only valid with **−a**.

−q **quotacheck** runs more quickly, particularly on systems with sparse user id usage, but fails to correct quotas for users [groups] not in the system user [group] database, and owning no files on the filesystem, if the quota file incorrectly believes that they do.

−u Only user quotas listed in `/etc/fstab` are to be checked. See also **−g**.

−v **quotacheck** is more verbose, and reports corrected discrepancies between the calculated and recorded disk quotas.

Specifying both **−g** and **−u** is equivalent to the default. Parallel passes are run on the filesystems required, using the pass numbers in `/etc/fstab` in an identical fashion to `fsck`(8).

Normally **quotacheck** operates silently.

quotacheck expects each filesystem to be checked to have a quota files named `quota.user` and `quota.group` which are located at the root of the associated file system. These defaults may be overridden in `/etc/fstab`. If a file is not present, **quotacheck** will create it.

quotacheck is normally run at boot time from the `/etc/rc` file, see `rc`(8), before enabling disk quotas with `quotaon`(8).

quotacheck accesses the raw device in calculating the actual disk usage for each user. Thus, the filesystems checked should be quiescent while **quotacheck** is running.

If **quotacheck** receives a `SIGINFO` signal (see the **status** argument for `stty`(1)), a line will be written to the standard error output indicating the name of the device currently being checked and progress information.

FILES

`quota.user`	at the filesystem root with user quotas
`quota.group`	at the filesystem root with group quotas

`/etc/fstab` default filesystems

SEE ALSO

`quota`(1), `quotactl`(2), `fstab`(5), `edquota`(8), `fsck`(8), `quotaon`(8), `repquota`(8)

HISTORY

The **`quotacheck`** command appeared in 4.2BSD.

NAME

quotaon, **quotaoff** — turn filesystem quotas on and off

SYNOPSIS

quotaon [**−g**] [**−u**] [**−v**] *filesystem ...*
quotaon [**−g**] [**−u**] [**−v**] **−a**
quotaoff [**−g**] [**−u**] [**−v**] *filesystem ...*
quotaoff [**−g**] [**−u**] [**−v**] **−a**

DESCRIPTION

quotaon announces to the system that disk quotas should be enabled on one or more filesystems. **quotaoff** announces to the system that the specified filesystems should have any disk quotas turned off. The filesystems specified must have entries in `/etc/fstab` and be mounted. **quotaon** expects each filesystem to have quota files named `quota.user` and `quota.group` which are located at the root of the associated file system. These defaults may be overridden in `/etc/fstab`. By default both user and group quotas are enabled.

Available options:

−a If the **−a** flag is supplied in place of any filesystem names, **quotaon**/**quotaoff** will enable/disable all the filesystems indicated in `/etc/fstab` to be read-write with disk quotas. By default only the types of quotas listed in `/etc/fstab` are enabled.

−g Only group quotas listed in `/etc/fstab` should be enabled/disabled.

−u Only user quotas listed in `/etc/fstab` should be enabled/disabled.

−v Causes **quotaon** and **quotaoff** to print a message for each filesystem where quotas are turned on or off.

Specifying both **−g** and **−u** is equivalent to the default.

FILES

`quota.user`	at the filesystem root with user quotas
`quota.group`	at the filesystem root with group quotas
`/etc/fstab`	filesystem table

SEE ALSO

`quota`(1), `quotactl`(2), `fstab`(5), `edquota`(8), `quotacheck`(8), `repquota`(8)

HISTORY

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

NAME

racoon — IKE (ISAKMP/Oakley) key management daemon

SYNOPSIS

racoon [**-46BdFLv**] [**-f** *configfile*] [**-l** *logfile*] [**-P** *isakmp-natt-port*]
[**-p** *isakmp-port*]

DESCRIPTION

racoon speaks the IKE (ISAKMP/Oakley) key management protocol, to establish security associations with other hosts. The SPD (Security Policy Database) in the kernel usually triggers **racoon**. **racoon** usually sends all informational messages, warnings and error messages to `syslogd`(8) with the facility `LOG_DAEMON` and the priority `LOG_INFO`. Debugging messages are sent with the priority `LOG_DEBUG`. You should configure `syslog.conf`(5) appropriately to see these messages.

-4

-6 Specify the default address family for the sockets.

-B Install SA(s) from the file which is specified in `racoon.conf`(5).

-d Increase the debug level. Multiple **-d** arguments will increase the debug level even more.

-F Run **racoon** in the foreground.

-f *configfile*
Use *configfile* as the configuration file instead of the default.

-L Include *file_name:line_number:function_name* in all messages.

-l *logfile*
Use *logfile* as the logging file instead of `syslogd`(8).

-P *isakmp-natt-port*
Use *isakmp-natt-port* for NAT-Traversal port-floating. The default is 4500.

-p *isakmp-port*
Listen to the ISAKMP key exchange on port *isakmp-port* instead of the default port number, 500.

-v This flag causes the packet dump be more verbose, with higher debugging level.

racoon assumes the presence of the kernel random number device `rnd`(4) at `/dev/urandom`.

RETURN VALUES

The command exits with 0 on success, and non-zero on errors.

FILES

`/etc/racoon.conf` default configuration file.

SEE ALSO

`ipsec`(4), `racoon.conf`(5), `syslog.conf`(5), `setkey`(8), `syslogd`(8)

HISTORY

The **racoon** command first appeared in the "YIPS" Yokogawa IPsec implementation.

SECURITY CONSIDERATIONS

The use of IKE phase 1 aggressive mode is not recommended, as described in `http://www.kb.cert.org/vuls/id/886601`.

NAME

racoonctl — racoon administrative control tool

SYNOPSIS

racoonctl reload-config
racoonctl show-schedule
racoonctl [**-l** [**-l**]] show-sa [isakmp|esp|ah|ipsec]
racoonctl flush-sa [isakmp|esp|ah|ipsec]
racoonctl delete-sa *saopts*
racoonctl establish-sa [**-u** *identity*] [**-w**] *saopts*
racoonctl vpn-connect [**-u** *identity*] *vpn_gateway*
racoonctl vpn-disconnect *vpn_gateway*
racoonctl show-event
racoonctl logout-user *login*

DESCRIPTION

racoonctl is used to control racoon(8) operation, if ipsec-tools was configured with adminport support. Communication between **racoonctl** and racoon(8) is done through a UNIX socket. By changing the default mode and ownership of the socket, you can allow non-root users to alter racoon(8) behavior, so do that with caution.

The following commands are available:

reload-config
This should cause racoon(8) to reload its configuration file.

show-schedule
Unknown command.

show-sa [isakmp|esp|ah|ipsec]
Dump the SA: All the SAs if no SA class is provided, or either ISAKMP SAs, IPsec ESP SAs, IPsec AH SAs, or all IPsec SAs. Use **-l** to increase verbosity.

flush-sa [isakmp|esp|ah|ipsec]
is used to flush all SAs if no SA class is provided, or a class of SAs, either ISAKMP SAs, IPsec ESP SAs, IPsec AH SAs, or all IPsec SAs.

establish-sa [**-u** *username*] [**-w**] *saopts*
Establish an SA, either an ISAKMP SA, IPsec ESP SA, or IPsec AH SA. The optional **-u** *username* can be used when establishing an ISAKMP SA while hybrid auth is in use. **racoonctl** will prompt you for the password associated with *username* and these credentials will be used in the Xauth exchange.

Specifying **-w** will make racoonctl wait until the SA is actually established or an error occurs.

saopts has the following format:

isakmp {inet|inet6} *src dst*

{esp|ah} {inet|inet6} *src/prefixlen/port dst/prefixlen/port*
{icmp|tcp|udp|gre|any}

vpn-connect [**-u** *username*] *vpn_gateway*
This is a particular case of the previous command. It will establish an ISAKMP SA with *vpn_gateway*.

delete-sa *saopts*
: Delete an SA, either an ISAKMP SA, IPsec ESP SA, or IPsec AH SA.

vpn-disconnect *vpn_gateway*
: This is a particular case of the previous command. It will kill all SAs associated with *vpn_gateway*.

show-event
: Listen for all events reported by `racoon`(8).

logout-user *login*
: Delete all SA established on behalf of the Xauth user *login*.

Command shortcuts are available:

rc	reload-config
ss	show-sa
sc	show-schedule
fs	flush-sa
ds	delete-sa
es	establish-sa
vc	vpn-connect
vd	vpn-disconnect
se	show-event
lu	logout-user

RETURN VALUES

The command should exit with 0 on success, and non-zero on errors.

FILES

`/var/racoon/racoon.sock` or
`/var/run/racoon.sock` `racoon`(8) control socket.

SEE ALSO

`ipsec`(4), `racoon`(8)

HISTORY

Once was **kmpstat** in the KAME project. It turned into **racoonctl** but remained undocumented for a while. Emmanuel Dreyfus ⟨manu@NetBSD.org⟩ wrote this man page.

NAME

raidctl — configuration utility for the RAIDframe disk driver

SYNOPSIS

raidctl [**−v**] **−a** *component dev*
raidctl [**−v**] **−A** [yes | no | root] *dev*
raidctl [**−v**] **−B** *dev*
raidctl [**−v**] **−c** *config_file dev*
raidctl [**−v**] **−C** *config_file dev*
raidctl [**−v**] **−f** *component dev*
raidctl [**−v**] **−F** *component dev*
raidctl [**−v**] **−g** *component dev*
raidctl [**−v**] **−G** *dev*
raidctl [**−v**] **−i** *dev*
raidctl [**−v**] **−I** *serial_number dev*
raidctl [**−v**] **−p** *dev*
raidctl [**−v**] **−P** *dev*
raidctl [**−v**] **−r** *component dev*
raidctl [**−v**] **−R** *component dev*
raidctl [**−v**] **−s** *dev*
raidctl [**−v**] **−S** *dev*
raidctl [**−v**] **−u** *dev*

DESCRIPTION

raidctl is the user-land control program for raid(4), the RAIDframe disk device. **raidctl** is primarily used to dynamically configure and unconfigure RAIDframe disk devices. For more information about the RAIDframe disk device, see raid(4).

This document assumes the reader has at least rudimentary knowledge of RAID and RAID concepts.

The command-line options for **raidctl** are as follows:

−a *component dev*
Add *component* as a hot spare for the device *dev*. Component labels (which identify the location of a given component within a particular RAID set) are automatically added to the hot spare after it has been used and are not required for *component* before it is used.

−A yes *dev*
Make the RAID set auto-configurable. The RAID set will be automatically configured at boot *before* the root file system is mounted. Note that all components of the set must be of type RAID in the disklabel.

−A no *dev*
Turn off auto-configuration for the RAID set.

−A root *dev*
Make the RAID set auto-configurable, and also mark the set as being eligible to be the root partition. A RAID set configured this way will *override* the use of the boot disk as the root device. All components of the set must be of type RAID in the disklabel. Note that only certain architectures (currently alpha, i386, pmax, sparc, sparc64, and vax) support booting a kernel directly from a RAID set.

−B *dev*
Initiate a copyback of reconstructed data from a spare disk to its original disk. This is performed after a component has failed, and the failed drive has been reconstructed onto a spare drive.

−c *config_file dev*
: Configure the RAIDframe device *dev* according to the configuration given in *config_file*. A description of the contents of *config_file* is given later.

−C *config_file dev*
: As for **−c**, but forces the configuration to take place. This is required the first time a RAID set is configured.

−f *component dev*
: This marks the specified *component* as having failed, but does not initiate a reconstruction of that component.

−F *component dev*
: Fails the specified *component* of the device, and immediately begin a reconstruction of the failed disk onto an available hot spare. This is one of the mechanisms used to start the reconstruction process if a component does have a hardware failure.

−g *component dev*
: Get the component label for the specified component.

−G *dev*
: Generate the configuration of the RAIDframe device in a format suitable for use with the **−c** or **−C** options.

−i *dev*
: Initialize the RAID device. In particular, (re-)write the parity on the selected device. This *MUST* be done for *all* RAID sets before the RAID device is labeled and before file systems are created on the RAID device.

−I *serial_number dev*
: Initialize the component labels on each component of the device. *serial_number* is used as one of the keys in determining whether a particular set of components belong to the same RAID set. While not strictly enforced, different serial numbers should be used for different RAID sets. This step *MUST* be performed when a new RAID set is created.

−p *dev*
: Check the status of the parity on the RAID set. Displays a status message, and returns successfully if the parity is up-to-date.

−P *dev*
: Check the status of the parity on the RAID set, and initialize (re-write) the parity if the parity is not known to be up-to-date. This is normally used after a system crash (and before a fsck(8)) to ensure the integrity of the parity.

−r *component dev*
: Remove the spare disk specified by *component* from the set of available spare components.

−R *component dev*
: Fails the specified *component*, if necessary, and immediately begins a reconstruction back to *component*. This is useful for reconstructing back onto a component after it has been replaced following a failure.

−s *dev*
: Display the status of the RAIDframe device for each of the components and spares.

−S *dev*
: Check the status of parity re-writing, component reconstruction, and component copyback. The output indicates the amount of progress achieved in each of these areas.

-u *dev*
Unconfigure the RAIDframe device.

-v Be more verbose. For operations such as reconstructions, parity re-writing, and copybacks, provide a progress indicator.

The device used by **raidctl** is specified by *dev*. *dev* may be either the full name of the device, e.g., `/dev/rraid0d`, for the i386 architecture, or `/dev/rraid0c` for many others, or just simply `raid0` (for `/dev/rraid0[cd]`). It is recommended that the partitions used to represent the RAID device are not used for file systems.

Configuration file

The format of the configuration file is complex, and only an abbreviated treatment is given here. In the configuration files, a '#' indicates the beginning of a comment.

There are 4 required sections of a configuration file, and 2 optional sections. Each section begins with a 'START', followed by the section name, and the configuration parameters associated with that section. The first section is the 'array' section, and it specifies the number of rows, columns, and spare disks in the RAID set. For example:

```
START array
1 3 0
```

indicates an array with 1 row, 3 columns, and 0 spare disks. Note that although multi-dimensional arrays may be specified, they are *NOT* supported in the driver.

The second section, the 'disks' section, specifies the actual components of the device. For example:

```
START disks
/dev/sd0e
/dev/sd1e
/dev/sd2e
```

specifies the three component disks to be used in the RAID device. If any of the specified drives cannot be found when the RAID device is configured, then they will be marked as 'failed', and the system will operate in degraded mode. Note that it is *imperative* that the order of the components in the configuration file does not change between configurations of a RAID device. Changing the order of the components will result in data loss if the set is configured with the **-C** option. In normal circumstances, the RAID set will not configure if only **-c** is specified, and the components are out-of-order.

The next section, which is the 'spare' section, is optional, and, if present, specifies the devices to be used as 'hot spares' — devices which are on-line, but are not actively used by the RAID driver unless one of the main components fail. A simple 'spare' section might be:

```
START spare
/dev/sd3e
```

for a configuration with a single spare component. If no spare drives are to be used in the configuration, then the 'spare' section may be omitted.

The next section is the 'layout' section. This section describes the general layout parameters for the RAID device, and provides such information as sectors per stripe unit, stripe units per parity unit, stripe units per reconstruction unit, and the parity configuration to use. This section might look like:

```
START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level
32 1 1 5
```

The sectors per stripe unit specifies, in blocks, the interleave factor; i.e., the number of contiguous sectors to be written to each component for a single stripe. Appropriate selection of this value (32 in this example) is the subject of much research in RAID architectures. The stripe units per parity unit and stripe units per reconstruction unit are normally each set to 1. While certain values above 1 are permitted, a discussion of valid values and the consequences of using anything other than 1 are outside the scope of this document. The last value in this section (5 in this example) indicates the parity configuration desired. Valid entries include:

0	RAID level 0. No parity, only simple striping.
1	RAID level 1. Mirroring. The parity is the mirror.
4	RAID level 4. Striping across components, with parity stored on the last component.
5	RAID level 5. Striping across components, parity distributed across all components.

There are other valid entries here, including those for Even-Odd parity, RAID level 5 with rotated sparing, Chained declustering, and Interleaved declustering, but as of this writing the code for those parity operations has not been tested with NetBSD.

The next required section is the 'queue' section. This is most often specified as:

```
START queue
fifo 100
```

where the queuing method is specified as fifo (first-in, first-out), and the size of the per-component queue is limited to 100 requests. Other queuing methods may also be specified, but a discussion of them is beyond the scope of this document.

The final section, the 'debug' section, is optional. For more details on this the reader is referred to the RAIDframe documentation discussed in the **HISTORY** section.

See **EXAMPLES** for a more complete configuration file example.

FILES

`/dev/{,r}raid*` **`raid`** device special files.

EXAMPLES

It is highly recommended that before using the RAID driver for real file systems that the system administrator(s) become quite familiar with the use of **`raidctl`**, and that they understand how the component reconstruction process works. The examples in this section will focus on configuring a number of different RAID sets of varying degrees of redundancy. By working through these examples, administrators should be able to develop a good feel for how to configure a RAID set, and how to initiate reconstruction of failed components.

In the following examples 'raid0' will be used to denote the RAID device. Depending on the architecture, `/dev/rraid0c` or `/dev/rraid0d` may be used in place of `raid0`.

Initialization and Configuration

The initial step in configuring a RAID set is to identify the components that will be used in the RAID set. All components should be the same size. Each component should have a disklabel type of `FS_RAID`, and a typical disklabel entry for a RAID component might look like:

```
f:  1800000  200495     RAID              # (Cyl.  405*- 4041*)
```

While `FS_BSDFFS` will also work as the component type, the type `FS_RAID` is preferred for RAIDframe use, as it is required for features such as auto-configuration. As part of the initial configuration of each RAID set, each component will be given a 'component label'. A 'component label' contains important information about the component, including a user-specified serial number, the row and column of that component in the RAID set, the redundancy level of the RAID set, a 'modification counter', and whether the parity information (if any) on that component is known to be correct. Component labels are an integral part of the RAID set, since they are used to ensure that components are configured in the correct order, and used to keep track of other vital information about the RAID set. Component labels are also required for the auto-detection and auto-configuration of RAID sets at boot time. For a component label to be considered valid, that particular component label must be in agreement with the other component labels in the set. For example, the serial number, 'modification counter', number of rows and number of columns must all be in agreement. If any of these are different, then the component is not considered to be part of the set. See `raid`(4) for more information about component labels.

Once the components have been identified, and the disks have appropriate labels, **`raidctl`** is then used to configure the `raid`(4) device. To configure the device, a configuration file which looks something like:

```
START array
# numRow numCol numSpare
1 3 1

START disks
/dev/sd1e
/dev/sd2e
/dev/sd3e

START spare
/dev/sd4e

START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_5
32 1 1 5

START queue
fifo 100
```

is created in a file. The above configuration file specifies a RAID 5 set consisting of the components `/dev/sd1e`, `/dev/sd2e`, and `/dev/sd3e`, with `/dev/sd4e` available as a 'hot spare' in case one of the three main drives should fail. A RAID 0 set would be specified in a similar way:

```
START array
# numRow numCol numSpare
1 4 0

START disks
/dev/sd10e
/dev/sd11e
/dev/sd12e
/dev/sd13e

START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_0
64 1 1 0

START queue
fifo 100
```

In this case, devices `/dev/sd10e`, `/dev/sd11e`, `/dev/sd12e`, and `/dev/sd13e` are the components that make up this RAID set. Note that there are no hot spares for a RAID 0 set, since there is no way to recover data if any of the components fail.

For a RAID 1 (mirror) set, the following configuration might be used:

```
START array
# numRow numCol numSpare
1 2 0

START disks
/dev/sd20e
/dev/sd21e

START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_1
128 1 1 1

START queue
fifo 100
```

In this case, `/dev/sd20e` and `/dev/sd21e` are the two components of the mirror set. While no hot spares have been specified in this configuration, they easily could be, just as they were specified in the RAID 5 case above. Note as well that RAID 1 sets are currently limited to only 2 components. At present, n-way mirroring is not possible.

The first time a RAID set is configured, the **−C** option must be used:

```
raidctl -C raid0.conf raid0
```

where `raid0.conf` is the name of the RAID configuration file. The **−C** forces the configuration to succeed, even if any of the component labels are incorrect. The **−C** option should not be used lightly in situations other than initial configurations, as if the system is refusing to configure a RAID set, there is probably a very good reason for it. After the initial configuration is done (and appropriate component labels are added with the **−I** option) then raid0 can be configured normally with:

```
raidctl -c raid0.conf raid0
```

When the RAID set is configured for the first time, it is necessary to initialize the component labels, and to initialize the parity on the RAID set. Initializing the component labels is done with:

```
raidctl -I 112341 raid0
```

where '112341' is a user-specified serial number for the RAID set. This initialization step is *required* for all RAID sets. As well, using different serial numbers between RAID sets is *strongly encouraged*, as using the same serial number for all RAID sets will only serve to decrease the usefulness of the component label checking.

Initializing the RAID set is done via the **−i** option. This initialization *MUST* be done for *all* RAID sets, since among other things it verifies that the parity (if any) on the RAID set is correct. Since this initialization may be quite time-consuming, the **−v** option may be also used in conjunction with **−i**:

```
raidctl -iv raid0
```

This will give more verbose output on the status of the initialization:

```
Initiating re-write of parity
Parity Re-write status:
 10% |****                                  | ETA:    06:03 /
```

The output provides a 'Percent Complete' in both a numeric and graphical format, as well as an estimated time to completion of the operation.

Since it is the parity that provides the 'redundancy' part of RAID, it is critical that the parity is correct as much as possible. If the parity is not correct, then there is no guarantee that data will not be lost if a component fails.

Once the parity is known to be correct, it is then safe to perform `disklabel`(8), `newfs`(8), or `fsck`(8) on the device or its file systems, and then to mount the file systems for use.

Under certain circumstances (e.g., the additional component has not arrived, or data is being migrated off of a disk destined to become a component) it may be desirable to configure a RAID 1 set with only a single component. This can be achieved by using the word "absent" to indicate that a particular component is not present. In the following:

```
START array
# numRow numCol numSpare
1 2 0

START disks
absent
/dev/sd0e

START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_1
128 1 1 1

START queue
fifo 100
```

`/dev/sd0e` is the real component, and will be the second disk of a RAID 1 set. The first component is simply marked as being absent. Configuration (using **-C** and **-I** *12345* as above) proceeds normally, but initialization of the RAID set will have to wait until all physical components are present. After configuration, this set can be used normally, but will be operating in degraded mode. Once a second physical component is obtained, it can be hot-added, the existing data mirrored, and normal operation resumed.

The size of the resulting RAID set will depend on the number of data components in the set. Space is automatically reserved for the component labels, and the actual amount of space used for data on a component will be rounded down to the largest possible multiple of the sectors per stripe unit (sectPerSU) value. Thus, the amount of space provided by the RAID set will be less than the sum of the size of the components.

Maintenance of the RAID set

After the parity has been initialized for the first time, the command:

```
raidctl -p raid0
```

can be used to check the current status of the parity. To check the parity and rebuild it necessary (for example, after an unclean shutdown) the command:

```
raidctl -P raid0
```

is used. Note that re-writing the parity can be done while other operations on the RAID set are taking place (e.g., while doing a `fsck`(8) on a file system on the RAID set). However: for maximum effectiveness of the RAID set, the parity should be known to be correct before any data on the set is modified.

To see how the RAID set is doing, the following command can be used to show the RAID set's status:

```
raidctl -s raid0
```

The output will look something like:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: optimal
           /dev/sd3e: optimal
Spares:
           /dev/sd4e: spare
Component label for /dev/sd1e:
   Row: 0 Column: 0 Num Rows: 1 Num Columns: 3
   Version: 2 Serial Number: 13432 Mod Counter: 65
   Clean: No Status: 0
   sectPerSU: 32 SUsPerPU: 1 SUsPerRU: 1
   RAID Level: 5  blocksize: 512 numBlocks: 1799936
   Autoconfig: No
   Last configured as: raid0
Component label for /dev/sd2e:
   Row: 0 Column: 1 Num Rows: 1 Num Columns: 3
   Version: 2 Serial Number: 13432 Mod Counter: 65
   Clean: No Status: 0
   sectPerSU: 32 SUsPerPU: 1 SUsPerRU: 1
   RAID Level: 5  blocksize: 512 numBlocks: 1799936
   Autoconfig: No
   Last configured as: raid0
Component label for /dev/sd3e:
   Row: 0 Column: 2 Num Rows: 1 Num Columns: 3
   Version: 2 Serial Number: 13432 Mod Counter: 65
   Clean: No Status: 0
   sectPerSU: 32 SUsPerPU: 1 SUsPerRU: 1
   RAID Level: 5  blocksize: 512 numBlocks: 1799936
   Autoconfig: No
   Last configured as: raid0
Parity status: clean
Reconstruction is 100% complete.
Parity Re-write is 100% complete.
Copyback is 100% complete.
```

This indicates that all is well with the RAID set. Of importance here are the component lines which read ‘optimal’, and the ‘Parity status’ line. ‘Parity status: clean’ indicates that the parity is up-to-date for this RAID set, whether or not the RAID set is in redundant or degraded mode. ‘Parity status: DIRTY’ indicates that it is not known if the parity information is consistent with the data, and that the parity information needs to be checked. Note that if there are file systems open on the RAID set, the individual components will not be ‘clean’ but the set as a whole can still be clean.

To check the component label of `/dev/sd1e`, the following is used:

```
raidctl -g /dev/sd1e raid0
```

The output of this command will look something like:

```
Component label for /dev/sd1e:
   Row: 0 Column: 0 Num Rows: 1 Num Columns: 3
   Version: 2 Serial Number: 13432 Mod Counter: 65
```

```
            Clean: No Status: 0
            sectPerSU: 32 SUsPerPU: 1 SUsPerRU: 1
            RAID Level: 5  blocksize: 512 numBlocks: 1799936
            Autoconfig: No
            Last configured as: raid0
```

Dealing with Component Failures

If for some reason (perhaps to test reconstruction) it is necessary to pretend a drive has failed, the following will perform that function:

```
raidctl -f /dev/sd2e raid0
```

The system will then be performing all operations in degraded mode, where missing data is re-computed from existing data and the parity. In this case, obtaining the status of raid0 will return (in part):

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: failed
           /dev/sd3e: optimal
Spares:
           /dev/sd4e: spare
```

Note that with the use of **−f** a reconstruction has not been started. To both fail the disk and start a reconstruction, the **−F** option must be used:

```
raidctl -F /dev/sd2e raid0
```

The **−f** option may be used first, and then the **−F** option used later, on the same disk, if desired. Immediately after the reconstruction is started, the status will report:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: reconstructing
           /dev/sd3e: optimal
Spares:
           /dev/sd4e: used_spare
[...]
Parity status: clean
Reconstruction is 10% complete.
Parity Re-write is 100% complete.
Copyback is 100% complete.
```

This indicates that a reconstruction is in progress. To find out how the reconstruction is progressing the **−S** option may be used. This will indicate the progress in terms of the percentage of the reconstruction that is completed. When the reconstruction is finished the **−s** option will show:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: spared
           /dev/sd3e: optimal
Spares:
           /dev/sd4e: used_spare
[...]
Parity status: clean
Reconstruction is 100% complete.
Parity Re-write is 100% complete.
```

```
Copyback is 100% complete.
```

At this point there are at least two options. First, if `/dev/sd2e` is known to be good (i.e., the failure was either caused by **−f** or **−F**, or the failed disk was replaced), then a copyback of the data can be initiated with the **−B** option. In this example, this would copy the entire contents of `/dev/sd4e` to `/dev/sd2e`. Once the copyback procedure is complete, the status of the device would be (in part):

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: optimal
           /dev/sd3e: optimal
Spares:
           /dev/sd4e: spare
```

and the system is back to normal operation.

The second option after the reconstruction is to simply use `/dev/sd4e` in place of `/dev/sd2e` in the configuration file. For example, the configuration file (in part) might now look like:

```
START array
1 3 0

START drives
/dev/sd1e
/dev/sd4e
/dev/sd3e
```

This can be done as `/dev/sd4e` is completely interchangeable with `/dev/sd2e` at this point. Note that extreme care must be taken when changing the order of the drives in a configuration. This is one of the few instances where the devices and/or their orderings can be changed without loss of data! In general, the ordering of components in a configuration file should *never* be changed.

If a component fails and there are no hot spares available on-line, the status of the RAID set might (in part) look like:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: failed
           /dev/sd3e: optimal
No spares.
```

In this case there are a number of options. The first option is to add a hot spare using:

```
raidctl -a /dev/sd4e raid0
```

After the hot add, the status would then be:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: failed
           /dev/sd3e: optimal
Spares:
           /dev/sd4e: spare
```

Reconstruction could then take place using **−F** as describe above.

A second option is to rebuild directly onto `/dev/sd2e`. Once the disk containing `/dev/sd2e` has been replaced, one can simply use:

```
raidctl -R /dev/sd2e raid0
```

to rebuild the `/dev/sd2e` component. As the rebuilding is in progress, the status will be:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: reconstructing
           /dev/sd3e: optimal
No spares.
```

and when completed, will be:

```
Components:
           /dev/sd1e: optimal
           /dev/sd2e: optimal
           /dev/sd3e: optimal
No spares.
```

In circumstances where a particular component is completely unavailable after a reboot, a special component name will be used to indicate the missing component. For example:

```
Components:
           /dev/sd2e: optimal
          component1: failed
No spares.
```

indicates that the second component of this RAID set was not detected at all by the auto-configuration code. The name 'component1' can be used anywhere a normal component name would be used. For example, to add a hot spare to the above set, and rebuild to that hot spare, the following could be done:

```
raidctl -a /dev/sd3e raid0
raidctl -F component1 raid0
```

at which point the data missing from 'component1' would be reconstructed onto `/dev/sd3e`.

When more than one component is marked as 'failed' due to a non-component hardware failure (e.g., loss of power to two components, adapter problems, termination problems, or cabling issues) it is quite possible to recover the data on the RAID set. The first thing to be aware of is that the first disk to fail will almost certainly be out-of-sync with the remainder of the array. If any IO was performed between the time the first component is considered 'failed' and when the second component is considered 'failed', then the first component to fail will *not* contain correct data, and should be ignored. When the second component is marked as failed, however, the RAID device will (currently) panic the system. At this point the data on the RAID set (not including the first failed component) is still self consistent, and will be in no worse state of repair than had the power gone out in the middle of a write to a file system on a non-RAID device. The problem, however, is that the component labels may now have 3 different 'modification counters' (one value on the first component that failed, one value on the second component that failed, and a third value on the remaining components). In such a situation, the RAID set will not autoconfigure, and can only be forcibly re-configured with the **−C** option. To recover the RAID set, one must first remedy whatever physical problem caused the multiple-component failure. After that is done, the RAID set can be restored by forcibly configuring the raid set *without* the component that failed first. For example, if `/dev/sd1e` and `/dev/sd2e` fail (in that order) in a RAID set of the following configuration:

```
START array
1 4 0

START drives
/dev/sd1e
```

```
/dev/sd2e
/dev/sd3e
/dev/sd4e

START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_5
64 1 1 5

START queue
fifo 100
```

then the following configuration (say "recover_raid0.conf")

```
START array
1 4 0

START drives
/dev/sd6e
/dev/sd2e
/dev/sd3e
/dev/sd4e

START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_5
64 1 1 5

START queue
fifo 100
```

(where `/dev/sd6e` has no physical device) can be used with

```
raidctl -C recover_raid0.conf raid0
```

to force the configuration of raid0. A

```
raidctl -I 12345 raid0
```

will be required in order to synchronize the component labels. At this point the file systems on the RAID set can then be checked and corrected. To complete the re-construction of the RAID set, `/dev/sd1e` is simply hot-added back into the array, and reconstructed as described earlier.

RAID on RAID

RAID sets can be layered to create more complex and much larger RAID sets. A RAID 0 set, for example, could be constructed from four RAID 5 sets. The following configuration file shows such a setup:

```
START array
# numRow numCol numSpare
1 4 0

START disks
/dev/raid1e
/dev/raid2e
/dev/raid3e
/dev/raid4e
```

```
START layout
# sectPerSU SUsPerParityUnit SUsPerReconUnit RAID_level_0
128 1 1 0

START queue
fifo 100
```

A similar configuration file might be used for a RAID 0 set constructed from components on RAID 1 sets. In such a configuration, the mirroring provides a high degree of redundancy, while the striping provides additional speed benefits.

Auto-configuration and Root on RAID

RAID sets can also be auto-configured at boot. To make a set auto-configurable, simply prepare the RAID set as above, and then do a:

```
raidctl -A yes raid0
```

to turn on auto-configuration for that set. To turn off auto-configuration, use:

```
raidctl -A no raid0
```

RAID sets which are auto-configurable will be configured before the root file system is mounted. These RAID sets are thus available for use as a root file system, or for any other file system. A primary advantage of using the auto-configuration is that RAID components become more independent of the disks they reside on. For example, SCSI ID's can change, but auto-configured sets will always be configured correctly, even if the SCSI ID's of the component disks have become scrambled.

Having a system's root file system (`/`) on a RAID set is also allowed, with the 'a' partition of such a RAID set being used for `/`. To use raid0a as the root file system, simply use:

```
raidctl -A root raid0
```

To return raid0a to be just an auto-configuring set simply use the **-A** *yes* arguments.

Note that kernels can only be directly read from RAID 1 components on architectures that support that (currently alpha, i386, pmax, sparc, sparc64, and vax). On those architectures, the `FS_RAID` file system is recognized by the bootblocks, and will properly load the kernel directly from a RAID 1 component. For other architectures, or to support the root file system on other RAID sets, some other mechanism must be used to get a kernel booting. For example, a small partition containing only the secondary boot-blocks and an alternate kernel (or two) could be used. Once a kernel is booting however, and an auto-configuring RAID set is found that is eligible to be root, then that RAID set will be auto-configured and used as the root device. If two or more RAID sets claim to be root devices, then the user will be prompted to select the root device. At this time, RAID 0, 1, 4, and 5 sets are all supported as root devices.

A typical RAID 1 setup with root on RAID might be as follows:

1. wd0a - a small partition, which contains a complete, bootable, basic NetBSD installation.
2. wd1a - also contains a complete, bootable, basic NetBSD installation.
3. wd0e and wd1e - a RAID 1 set, raid0, used for the root file system.
4. wd0f and wd1f - a RAID 1 set, raid1, which will be used only for swap space.
5. wd0g and wd1g - a RAID 1 set, raid2, used for `/usr`, `/home`, or other data, if desired.
6. wd0h and wd1h - a RAID 1 set, raid3, if desired.

RAID sets raid0, raid1, and raid2 are all marked as auto-configurable. raid0 is marked as being a root file system. When new kernels are installed, the kernel is not only copied to `/`, but also to wd0a and wd1a. The kernel on wd0a is required, since that is the kernel the system boots from. The kernel on wd1a is also required, since that will be the kernel used should wd0 fail. The important point here is to have redundant copies of the kernel available, in the event that one of the drives fail.

There is no requirement that the root file system be on the same disk as the kernel. For example, obtaining the kernel from wd0a, and using sd0e and sd1e for raid0, and the root file system, is fine. It *is* critical, however, that there be multiple kernels available, in the event of media failure.

Multi-layered RAID devices (such as a RAID 0 set made up of RAID 1 sets) are *not* supported as root devices or auto-configurable devices at this point. (Multi-layered RAID devices *are* supported in general, however, as mentioned earlier.) Note that in order to enable component auto-detection and auto-configuration of RAID devices, the line:

```
options    RAID_AUTOCONFIG
```

must be in the kernel configuration file. See `raid`(4) for more details.

Swapping on RAID

A RAID device can be used as a swap device. In order to ensure that a RAID device used as a swap device is correctly unconfigured when the system is shutdown or rebooted, it is recommended that the line

```
swapoff=YES
```

be added to `/etc/rc.conf`.

Unconfiguration

The final operation performed by **`raidctl`** is to unconfigure a `raid`(4) device. This is accomplished via a simple:

```
raidctl -u raid0
```

at which point the device is ready to be reconfigured.

Performance Tuning

Selection of the various parameter values which result in the best performance can be quite tricky, and often requires a bit of trial-and-error to get those values most appropriate for a given system. A whole range of factors come into play, including:

1. Types of components (e.g., SCSI vs. IDE) and their bandwidth
2. Types of controller cards and their bandwidth
3. Distribution of components among controllers
4. IO bandwidth
5. file system access patterns
6. CPU speed

As with most performance tuning, benchmarking under real-life loads may be the only way to measure expected performance. Understanding some of the underlying technology is also useful in tuning. The goal of this section is to provide pointers to those parameters which may make significant differences in performance.

For a RAID 1 set, a SectPerSU value of 64 or 128 is typically sufficient. Since data in a RAID 1 set is arranged in a linear fashion on each component, selecting an appropriate stripe size is somewhat less critical than it is for a RAID 5 set. However: a stripe size that is too small will cause large IO's to be broken up into a number of smaller ones, hurting performance. At the same time, a large stripe size may cause problems with concurrent accesses to stripes, which may also affect performance. Thus values in the range of 32 to

128 are often the most effective.

Tuning RAID 5 sets is trickier. In the best case, IO is presented to the RAID set one stripe at a time. Since the entire stripe is available at the beginning of the IO, the parity of that stripe can be calculated before the stripe is written, and then the stripe data and parity can be written in parallel. When the amount of data being written is less than a full stripe worth, the 'small write' problem occurs. Since a 'small write' means only a portion of the stripe on the components is going to change, the data (and parity) on the components must be updated slightly differently. First, the 'old parity' and 'old data' must be read from the components. Then the new parity is constructed, using the new data to be written, and the old data and old parity. Finally, the new data and new parity are written. All this extra data shuffling results in a serious loss of performance, and is typically 2 to 4 times slower than a full stripe write (or read). To combat this problem in the real world, it may be useful to ensure that stripe sizes are small enough that a 'large IO' from the system will use exactly one large stripe write. As is seen later, there are some file system dependencies which may come into play here as well.

Since the size of a 'large IO' is often (currently) only 32K or 64K, on a 5-drive RAID 5 set it may be desirable to select a SectPerSU value of 16 blocks (8K) or 32 blocks (16K). Since there are 4 data sectors per stripe, the maximum data per stripe is 64 blocks (32K) or 128 blocks (64K). Again, empirical measurement will provide the best indicators of which values will yeild better performance.

The parameters used for the file system are also critical to good performance. For `newfs`(8), for example, increasing the block size to 32K or 64K may improve performance dramatically. As well, changing the cylinders-per-group parameter from 16 to 32 or higher is often not only necessary for larger file systems, but may also have positive performance implications.

Summary

Despite the length of this man-page, configuring a RAID set is a relatively straight-forward process. All that needs to be done is the following steps:

1. Use `disklabel`(8) to create the components (of type RAID).
2. Construct a RAID configuration file: e.g., `raid0.conf`
3. Configure the RAID set with:
   ```
   raidctl -C raid0.conf raid0
   ```
4. Initialize the component labels with:
   ```
   raidctl -I 123456 raid0
   ```
5. Initialize other important parts of the set with:
   ```
   raidctl -i raid0
   ```
6. Get the default label for the RAID set:
   ```
   disklabel raid0 > /tmp/label
   ```
7. Edit the label:
   ```
   vi /tmp/label
   ```
8. Put the new label on the RAID set:
   ```
   disklabel -R -r raid0 /tmp/label
   ```
9. Create the file system:
   ```
   newfs /dev/rraid0e
   ```

10. Mount the file system:

    ```
    mount /dev/raid0e /mnt
    ```

11. Use:

    ```
    raidctl -c raid0.conf raid0
    ```

    To re-configure the RAID set the next time it is needed, or put `raid0.conf` into `/etc` where it will automatically be started by the `/etc/rc.d` scripts.

SEE ALSO

`ccd`(4), `raid`(4), `rc`(8)

HISTORY

RAIDframe is a framework for rapid prototyping of RAID structures developed by the folks at the Parallel Data Laboratory at Carnegie Mellon University (CMU). A more complete description of the internals and functionality of RAIDframe is found in the paper "RAIDframe: A Rapid Prototyping Tool for RAID Systems", by William V. Courtright II, Garth Gibson, Mark Holland, LeAnn Neal Reilly, and Jim Zelenka, and published by the Parallel Data Laboratory of Carnegie Mellon University.

The **raidctl** command first appeared as a program in CMU's RAIDframe v1.1 distribution. This version of **raidctl** is a complete re-write, and first appeared in NetBSD 1.4.

COPYRIGHT

WARNINGS

Certain RAID levels (1, 4, 5, 6, and others) can protect against some data loss due to component failure. However the loss of two components of a RAID 4 or 5 system, or the loss of a single component of a RAID 0 system will result in the entire file system being lost. RAID is *NOT* a substitute for good backup practices.

Recomputation of parity *MUST* be performed whenever there is a chance that it may have been compromised. This includes after system crashes, or before a RAID device has been used for the first time. Failure to keep parity correct will be catastrophic should a component ever fail — it is better to use RAID 0 and get the additional space and speed, than it is to use parity, but not keep the parity correct. At least with RAID 0 there is no perception of increased data security.

BUGS

Hot-spare removal is currently not available.

NAME

rarpd — Reverse ARP Daemon

SYNOPSIS

rarpd [**-adfl**] [*interface* [*. . .*]]

DESCRIPTION

rarpd services Reverse ARP requests on the Ethernet connected to *interface*. Upon receiving a request, **rarpd** maps the target hardware address to an IP address via its name, which must be present in both the ethers(5) and hosts(5) databases. If a host does not exist in both databases the translation cannot proceed and a reply will not be sent.

In normal operation, **rarpd** forks a copy of itself and runs in the background. Anomalies and errors are reported via syslog(3).

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. This option implies the **-f** option.

-f Run in the foreground.

-l Log all requests to syslog.

FILES

/etc/ethers
/etc/hosts

SEE ALSO

bpf(4)

Finlayson, R., Mann, T., Mogul, J.C., and Theimer, M., A Reverse Address Resolution Protocol, RFC 903.

AUTHORS

Craig Leres (leres@ee.lbl.gov) and Steven McCanne (mccanne@ee.lbl.gov). Lawrence Berkeley Laboratory, University of California, Berkeley, CA.

NAME

rbootd — HP remote boot server

SYNOPSIS

rbootd [**-ad**] [**-i** *interface*] [config_file]

DESCRIPTION

The **rbootd** utility services boot requests from Hewlett-Packard workstations over a local area network. All boot files must reside in the boot file directory; further, if a client supplies path information in its boot request, it will be silently stripped away before processing. By default, **rbootd** only responds to requests from machines listed in its configuration file. If the client doesn't supply a file name (HP700 series machines don't), the first one listed for this machine will be supplied.

The options are as follows:

-a Respond to boot requests from any machine. The configuration file is ignored if this option is specified.

-d Run **rbootd** in debug mode. Packets sent and received are displayed to the terminal.

-i *interface*
Service boot requests on specified interface. If unspecified, **rbootd** searches the system interface list for the lowest numbered, configured ‘‘up’’ interface (excluding loopback). Ties are broken by choosing the earliest match.

Specifying *config_file* on the command line causes **rbootd** to use a different configuration file from the default.

The configuration file is a text file where each line describes a particular machine. A line must start with a machine's Ethernet address followed by an optional list of boot file names. An Ethernet address is specified in hexadecimal with each of its six octets separated by a colon. The boot file names come from the boot file directory. The ethernet address and boot file(s) must be separated by white-space and/or comma characters. A pound sign causes the remainder of a line to be ignored.

Here is a sample configuration file:

```
#
# ethernet addr      boot file(s)          comments
#
08:00:09:0:66:ad     SYSHPBSD              # snake (4.3BSD)
08:00:09:0:59:5b                           # vandy (anything)
8::9:1:C6:75         SYSHPBSD,SYSHPUX      # jaguar (either)
```

rbootd logs status and error messages via syslog(3). A startup message is always logged, and in the case of fatal errors (or deadly signals) a message is logged announcing the server's termination. In general, a non-fatal error is handled by ignoring the event that caused it (e.g. an invalid Ethernet address in the config file causes that line to be invalidated).

The following signals have the specified effect when sent to the server process using the kill(1) command:

SIGHUP	Drop all active connections and reconfigure.
SIGUSR1	Turn on debugging, do nothing if already on.
SIGUSR2	Turn off debugging, do nothing if already off.

FILES

`/dev/bpf`	packet-filter device
`/etc/rbootd.conf`	configuration file
`/tmp/rbootd.dbg`	debug output
`/usr/mdec/rbootd`	directory containing boot files
`/var/run/rbootd.pid`	process id

SEE ALSO

`kill`(1), `socket`(2), `signal`(3), `syslog`(3), `rmp`(4)

BUGS

If multiple servers are started on the same interface, each will receive and respond to the same boot packets.

NAME

rc, **rc.shutdown**, **rc.d/** — startup and shutdown scripts

SYNOPSIS

rc
rc.shutdown
rc.d/

DESCRIPTION

rc is the command script which controls the startup of various services, and is invoked by `init`(8) as part of the process of entering the automatic reboot to multi-user startup, or after the single user mode shell has exited. If `init`(8) is starting the automatic reboot process, **rc** is invoked with the argument of 'autoboot'.

rc.shutdown is the command script which shuts down various services, and is invoked by `shutdown`(8) as part of the process of shutting down the system.

rc.d/ is the directory which contains various `sh`(1) scripts, one for each service, which are called by **rc** at startup, **rc.shutdown** at shutdown, and as necessary during system operation to stop, start, restart, reload, or otherwise control the service.

Operation of rc

1. Source `/etc/rc.subr` to load various `rc.subr`(8) shell functions to use.
2. If autobooting, set **autoboot=yes** and enable a flag (**rc_fast=yes**), which prevents the **rc.d** scripts from performing the check for already running processes (thus speeding up the boot process). This **rc_fast=yes** speedup won't occur when **rc** is started up after exiting the single-user shell.
3. Invoke `rcorder`(8) to order the files in `/etc/rc.d/` that do not have a "nostart" keyword (refer to `rcorder`(8)'s **−s** flag), and assigns the result to a variable.
4. Calls each script in turn using **run_rc_script**() (from `rc.subr`(8)), which sets `$1` to 'start', and sources the script in a subshell. If the script has a '.sh' suffix then it is sourced directly into the current shell.

Operation of rc.shutdown

1. Source `/etc/rc.subr` to load various `rc.subr`(8) shell functions to use.
2. Invoke `rcorder`(8) to order the files in `/etc/rc.d/` that have a "shutdown" keyword (refer to `rcorder`(8)'s **−k** flag), reverses that order, and assigns the result to a variable.
3. Calls each script in turn using **run_rc_script**() (from `rc.subr`(8)), which sets `$1` to 'stop', and sources the script in a subshell. If the script has a '.sh' suffix then it is sourced directly into the current shell.

Contents of rc.d/

rc.d/ is located in `/etc/rc.d`. The following file naming conventions are currently used in **rc.d/**:

`ALLUPPERCASE` Scripts that are 'placeholders' to ensure that certain operations are performed before others. In order of startup, these are:

`NETWORKING`	Ensure basic network services are running, including general network configuration (`network`) and `dhclient`.
`SERVERS`	Ensure basic services (such as `NETWORKING`, `ppp`, `syslogd`, and `kdc`) exist for services that start early (such as `named`), because they're required by `DAEMON` below.

`DAEMON`	Before all general purpose daemons such as `dhcpd`, `lpd`, and `ntpd`.
`LOGIN`	Before user login services (`inetd`, `telnetd`, `rshd`, `sshd`, and `xdm`), as well as before services which might run commands as users (`cron`, `postfix`, and `sendmail`).

`foo.sh` Scripts that are to be sourced into the current shell rather than a subshell have a '`.sh`' suffix. Extreme care must be taken in using this, as the startup sequence will terminate if the script does. `/etc/rc.d/bootconf.sh` uses this behaviour to allow the user to select a different configuration (including `/etc/rc.conf`) early in the boot.

`bar` Scripts that are sourced in a subshell. The boot does not stop if such a script terminates with a non-zero status, but a script can stop the boot if necessary by invoking the **`stop_boot`**() function (from `rc.subr`(8)).

Each script should contain `rcorder`(8) keywords, especially an appropriate "PROVIDE" entry.

The scripts are expected to support at least the following arguments:

start Start the service. This should check that the service is to be started as specified by `rc.conf`(5). Also checks if the service is already running and refuses to start if it is. This latter check is not performed by standard NetBSD scripts if the system is starting directly to multi-user mode, to speed up the boot process.

stop If the service is to be started as specified by `rc.conf`(5), stop the service. This should check that the service is running and complain if it's not.

restart Perform a **stop** then a **start**.

status If the script starts a process (rather than performing a one-off operation), show the status of the process. Otherwise it's not necessary to support this argument. Defaults to displaying the process ID of the program (if running).

poll If the script starts a process (rather than performing a one-off operation), wait for the command to exit. Otherwise it's not necessary to support this argument.

rcvar Display which `rc.conf`(5) variables are used to control the startup of the service (if any).

Other arguments (such as 'reload', 'dumpdb', etc) can be added if necessary.

The argument may have one of the following prefixes to alter its operation:

fast Skip the check for an existing running process. Sets **rc_fast=yes**.

force Skips the `rc.conf`(5) check, ignores a failure result from any of the prerequisite checks, executes the command, and always returns a zero exit status. Sets **rc_force=yes**.

one Skips the `rc.conf`(5) check, but performs all other prerequisite tests.

In order to simplify scripts, the **`run_rc_command`**() function from `rc.subr`(8) may be used.

FILES

`/etc/rc`	Startup script called by `init`(8).
`/etc/rc.d/`	Directory containing control scripts for each service.

`/etc/rc.shutdown`	Shutdown script called by `shutdown`(8).
`/etc/rc.subr`	Contains `rc.subr`(8) functions used by various scripts.
`/etc/rc.conf`	System startup configuration file.

SEE ALSO

`rc.conf`(5), `init`(8), `rc.subr`(8), `rcorder`(8), `reboot`(8), `shutdown`(8)

HISTORY

The **`rc`** command appeared in 4.0BSD. The `/etc/rc.d` support was implemented in NetBSD 1.5 by Luke Mewburn ⟨lukem@NetBSD.org⟩.

NAME

rc.subr — functions used by system shell scripts

SYNOPSIS

. /etc/rc.subr

backup_file *action file current backup*

checkyesno *var*

check_pidfile *pidfile procname* [*interpreter*]

check_process *procname* [*interpreter*]

err *exitval message*

load_rc_config *command*

load_rc_config_var *command var*

mount_critical_filesystems *type*

rc_usage *command* [*...*]

reverse_list *item* [*...*]

run_rc_command *argument*

run_rc_script *file argument*

stop_boot

wait_for_pids [*pid* [*...*]]

warn *message*

DESCRIPTION

rc.subr contains commonly used shell script functions which are used by various scripts such as rc(8), and the periodic system services which are controlled by daily.conf(5), monthly.conf(5), security.conf(5), and weekly.conf(5).

The **rc.subr** functions are accessed by sourcing /etc/rc.subr into the current shell.

The following shell functions are available:

backup_file *action file current backup*

Make a backup copy of *file* into *current*. If the rc.conf(5) variable **backup_uses_rcs** is 'YES', use rcs(1) to archive the previous version of *current*, otherwise save the previous version of *current* as *backup*.

action may be one of the following:

add *file* is now being backed up by or possibly re-entered into this backup mechanism. *current* is created, and if necessary, the rcs(1) files are created as well.

update *file* has changed and needs to be backed up. If *current* exists, it is copied to *backup* or checked into rcs(1) (if the repository file is old), and then *file* is copied to *current*.

remove *file* is no longer being tracked by this backup mechanism. If rcs(1) is being used, an empty file is checked in and *current* is removed, otherwise *current* is moved to *backup*.

checkyesno *var*

Return 0 if *var* is defined to 'YES', 'TRUE', 'ON', or '1'. Return 1 if *var* is defined to 'NO', 'FALSE', 'OFF', or '0'. Otherwise, warn that *var* is not set correctly. The values are case insensitive.

Note that the warning message shown by this function when *var* is not set references a manual page where the user can find more information. Its name is picked up from the **rcvar_manpage** variable.

check_pidfile *pidfile procname* [*interpreter*]

Parses the first word of the first line of *pidfile* for a PID, and ensures that the process with that PID is running and its first argument matches *procname*. Prints the matching PID if successful, otherwise nothing. If *interpreter* is provided, parse the first line of *procname*, ensure that the line is of the form

```
#! interpreter [...]
```

and use *interpreter* with its optional arguments and *procname* appended as the process string to search for.

check_process *procname* [*interpreter*]

Prints the PIDs of any processes that are running with a first argument that matches *procname*. *interpreter* is handled as per **check_pidfile**.

err *exitval message*

Display an error message to *stderr*, log it to the system log using `logger`(1), and **exit** with an exit value of *exitval*. The error message consists of the script name (from **$0**), followed by ": ERROR: ", and then *message*.

load_rc_config *command*

Source in the `rc.conf`(5) configuration files for *command*. First, `/etc/rc.conf` is sourced if it has not yet been read in. Then, `/etc/rc.conf.d/`*command* is sourced if it is an existing file. The latter may also contain other variable assignments to override **run_rc_command** arguments defined by the calling script, to provide an easy mechanism for an administrator to override the behaviour of a given `rc.d`(8) script without requiring the editing of that script.

load_rc_config_var *command var*

Read the `rc.conf`(5) variable *var* for *command* and set in the current shell, using **load_rc_config** in a sub-shell to prevent unwanted side effects from other variable assignments.

mount_critical_filesystems *type*

Go through a list of critical file systems, as found in the `rc.conf`(5) variable **critical_filesystems_***type*, mounting each one that is not currently mounted.

rc_usage *command* [*...*]

Print a usage message for **$0**, with *commands* being the list of valid arguments prefixed by "[fast|force|one]".

reverse_list *item* [*...*]

Print the list of *items* in reverse order.

run_rc_command *argument*

Run the *argument* method for the current `rc.d`(8) script, based on the settings of various shell variables. **run_rc_command** is extremely flexible, and allows fully functional `rc.d`(8) scripts to be implemented in a small amount of shell code.

argument is searched for in the list of supported commands, which may be one of:

start Start the service. This should check that the service is to be started as specified by `rc.conf`(5). Also checks if the service is already running and refuses to start if it is. This latter check is not performed by standard NetBSD scripts if the system is starting directly to multi-user mode, to speed up the boot process.

stop If the service is to be started as specified by `rc.conf`(5), stop the service. This should check that the service is running and complain if it's not.

restart Perform a **stop** then a **start**. Defaults to displaying the process ID of the program (if running).

rcvar Display which `rc.conf`(5) variables are used to control the startup of the service (if any).

If **pidfile** or **procname** is set, also support:

poll Wait for the command to exit.

status Show the status of the process.

Other supported commands are listed in the optional variable **extra_commands**.

argument may have one of the following prefixes which alters its operation:

fast Skip the check for an existing running process, and sets **rc_fast=YES**.

force Skip the checks for **rcvar** being set to yes, and sets **rc_force=YES**. This ignores *argument*_**precmd** returning non-zero, and ignores any of the **required_*** tests failing, and always returns a zero exit status.

one Skip the checks for **rcvar** being set to yes, but performs all the other prerequisite tests.

run_rc_command uses the following shell variables to control its behaviour. Unless otherwise stated, these are optional.

name The name of this script. This is not optional.

rcvar The value of **rcvar** is checked with **checkyesno** to determine if this method should be run.

rcvar_manpage
The manual page containing information about **rcvar**. It will be part of the warning message shown when **rcvar** is undefined. Defaults to `rc.conf`(5).

command Full path to the command. Not required if *argument*_**cmd** is defined for each supported keyword.

command_args
Optional arguments and/or shell directives for **command**.

command_interpreter
command is started with

```
#! command_interpreter [...]
```

which results in its `ps`(1) command being

```
command_interpreter [...] command
```

so use that string to find the PID(s) of the running command rather than `command`.

extra_commands
Extra commands/keywords/arguments supported.

pidfile Path to pid file. Used to determine the PID(s) of the running command. If **pidfile** is set, use

```
check_pidfile $pidfile $procname
```

to find the PID. Otherwise, if **command** is set, use

```
check_process $procname
```

to find the PID.

procname Process name to check for. Defaults to the value of **command**.

required_dirs
Check for the existence of the listed directories before running the default start method.

required_files
Check for the readability of the listed files before running the default start method.

required_vars
Perform **checkyesno** on each of the list variables before running the default start method.

${name}_chdir
Directory to **cd** to before running **command**, if **${name}_chroot** is not provided.

${name}_chroot
Directory to chroot(8) to before running **command**. Only supported after /usr is mounted.

${name}_env
List of additional or modified environment variables to set when starting **command**.

${name}_flags
Arguments to call **command** with. This is usually set in rc.conf(5), and not in the rc.d(8) script. The environment variable 'flags' can be used to override this.

${name}_nice
nice(1) level to run **command** as. Only supported after /usr is mounted.

${name}_user
User to run **command** as, using chroot(8). if **${name}_chroot** is set, otherwise uses su(1). Only supported after /usr is mounted.

${name}_group
Group to run the chrooted **command** as.

${name}_groups
Comma separated list of supplementary groups to run the chrooted **command** with.

*argument*_**cmd**
Shell commands which override the default method for *argument*.

*argument*_**precmd**
Shell commands to run just before running *argument*_**cmd** or the default method for *argument*. If this returns a non-zero exit code, the main method is not performed. If the default method is being executed, this check is performed after the **required_*** checks and process (non-)existence checks.

*argument*_**postcmd**
Shell commands to run if running *argument*_**cmd** or the default method for *argument* returned a zero exit code.

sig_stop	Signal to send the processes to stop in the default **stop** method. Defaults to `SIGTERM`.
sig_reload	Signal to send the processes to reload in the default **reload** method. Defaults to `SIGHUP`.

For a given method *`argument`*, if *`argument`***_cmd** is not defined, then a default method is provided by **run_rc_command**:

Argument	**Default method**
start	If **command** is not running and **`checkyesno`** **rcvar** succeeds, start **command**.
stop	Determine the PIDs of **command** with **`check_pidfile`** or **`check_process`** (as appropriate), **`kill`** **sig_stop** those PIDs, and run **`wait_for_pids`** on those PIDs.
reload	Similar to **stop**, except that it uses **sig_reload** instead, and doesn't run **`wait_for_pids`**.
restart	Runs the **stop** method, then the **start** method.
status	Show the PID of **command**, or some other script specific status operation.
poll	Wait for **command** to exit.
rcvar	Display which `rc.conf`(5) variable is used (if any). This method always works, even if the appropriate `rc.conf`(5) variable is set to 'NO'.

The following variables are available to the methods (such as *`argument`***_cmd**) as well as after **`run_rc_command`** has completed:

rc_arg	Argument provided to **run_rc_command**, after fast and force processing has been performed.
rc_flags	Flags to start the default command with. Defaults to **${name}_flags**, unless overridden by the environment variable '`flags`'. This variable may be changed by the *`argument`***_precmd** method.
rc_pid	PID of **command** (if appropriate).
rc_fast	Not empty if "fast" prefix was used.
rc_force	Not empty if "force" prefix was used.

`run_rc_script` *`file argument`*

Start the script *`file`* with an argument of *`argument`*, and handle the return value from the script.

Various shell variables are unset before *`file`* is started:

name, **command**, **command_args**, **command_interpreter**, **extra_commands**, **pidfile**, **rcvar**, **required_dirs**, **required_files**, **required_vars**, *`argument`***_cmd**, *`argument`***_precmd**. *`argument`***_postcmd**.

The startup behaviour of *`file`* depends upon the following checks:

1. If *`file`* ends in `.sh`, it is sourced into the current shell.
2. If *`file`* appears to be a backup or scratch file (e.g., with a suffix of '~', '#', '.OLD', or '.orig'), ignore it.

3. If *file* is not executable, ignore it.
4. If the rc.conf(5) variable **rc_fast_and_loose** is empty, source *file* in a sub shell, otherwise source *file* into the current shell.

stop_boot
Prevent booting to multiuser mode. If the **autoboot** variable is 'yes', then a **SIGTERM** signal is sent to the parent process (which is assumed to be rc(8)). Otherwise, the shell exits with status 1.

wait_for_pids [*pid* [...]]
Wait until all of the provided *pids* don't exist any more, printing the list of outstanding *pids* every two seconds.

warn *message*
Display a warning message to *stderr* and log it to the system log using logger(1). The warning message consists of the script name (from **$0**), followed by ": WARNING: ", and then *message*.

FILES

/etc/rc.subr The **rc.subr** file resides in /etc.

SEE ALSO

rc.conf(5), rc(8)

HISTORY

rc.subr appeared in NetBSD 1.3. The rc.d(8) support functions appeared in NetBSD 1.5.

NAME

rcorder — print a dependency ordering of interdependent files

SYNOPSIS

rcorder [**−k** *keep*] [**−s** *skip*] *file ...*

DESCRIPTION

rcorder is designed to print out a dependency ordering of a set of interdependent files. Typically it is used to find an execution sequence for a set of shell scripts in which certain files must be executed before others.

Each file passed to **rcorder** should be annotated with special lines (which look like comments to the shell) which indicate the dependencies the files have upon certain points in the sequence, known as "conditions", and which indicate, for each file, which "conditions" may be expected to be filled by that file.

Within each file, a block containing a series of "REQUIRE", "PROVIDE", "BEFORE" and "KEYWORD" lines should appear. The format of the lines is rigid. Each line must begin with a single "#", followed by a single space, followed by "PROVIDE:", "REQUIRE:", "BEFORE:", or "KEYWORD:". No deviation is permitted. Each dependency line is then followed by a series of conditions, separated by whitespace. Multiple "PROVIDE", "REQUIRE", "BEFORE" and "KEYWORD" lines may appear, but all such lines must appear in a sequence without any intervening lines, as once a line that does not follow the format is reached, parsing stops.

The options are as follows:

−k Add the specified keyword to the "keep list". If any **−k** option is given, only those files containing the matching keyword are listed.

−s Add the specified keyword to the "skip list". If any **−s** option is given, files containing the matching keyword are not listed.

An example block follows:

```
# REQUIRE: networking syslog
# REQUIRE: usr
# PROVIDE: dns nscd
```

This block states that the file in which it appears depends upon the "networking", "syslog", and "usr" conditions, and provides the "dns" and "nscd" conditions.

A file may contain zero "PROVIDE" lines, in which case it provides no conditions, and may contain zero "REQUIRE" lines, in which case it has no dependencies. A file containing no "PROVIDE", "REQUIRE", or "BEFORE" lines may be output at an arbitrary position in the dependency ordering.

There must be at least one file with no dependencies in the set of arguments passed to **rcorder** in order for it to find a starting place in the dependency ordering.

DIAGNOSTICS

rcorder may print one of the following error messages and exit with a non-zero status if it encounters an error while processing the file list.

Requirement %s has no providers, aborting. No file has a "PROVIDE" line corresponding to a condition present in a "REQUIRE" line in another file.

Circular dependency on provision %s, aborting. A set of files has a circular dependency which was detected while processing the stated condition.

Circular dependency on file %s, aborting. A set of files has a circular dependency which was detected while processing the stated file.

SEE ALSO

`rc`(8)

HISTORY

The **`rcorder`** program first appeared in NetBSD 1.5.

AUTHORS

Written by Perry E. Metzger ⟨perry@piermont.com⟩ and Matthew R. Green ⟨mrg@eterna.com.au⟩.

NAME

rdate — set the system's date from a remote host

SYNOPSIS

rdate [**−psa**] *host*

DESCRIPTION

rdate displays and sets the local date and time from the host name or address given as the argument. It uses the RFC 868 protocol which is usually implemented as a built-in service of inetd(8).

Available options:

−p Do not set, just print the remote time

−s Do not print the time.

−a Use the adjtime(2) call to gradually skew the local time to the remote time rather than just hopping.

FILES

/var/log/wtmp A record of date resets and time changes.

SEE ALSO

adjtime(2), gettimeofday(2), utmp(5), inetd(8)

NAME

reboot, **poweroff**, **halt** — restarting, powering down and stopping the system

SYNOPSIS

halt [**-dlnpq**]
poweroff [**-dlnq**]
reboot [**-dlnq**] [*arg ...*]

DESCRIPTION

The **poweroff**, **halt** and **reboot** utilities flush the file system cache to disk, send all running processes a SIGTERM, wait for up to 30 seconds for them to die, send a SIGKILL to the survivors and, respectively, power down, halt or restart the system. The action is logged, including entering a shutdown record into the login accounting file and sending a message via syslog(3).

The options are as follows:

-d Create a dump before halting or restarting. This option is useful for debugging system dump procedures or capturing the state of a corrupted or misbehaving system.

-l Suppress sending a message via syslog(3) before halting or restarting.

-n Do not flush the file system cache. This option should be used with extreme caution. It can be used if a disk or the processor is on fire.

-p Attempt to powerdown the system. If the powerdown fails, or the system does not support software powerdown, the system will halt. This option is only valid for **halt**.

-q Do not give processes a chance to shut down before halting or restarting. This option should not normally be used.

If there are any arguments passed to **reboot** they are concatenated with spaces and passed as *bootstr* to the reboot(2) system call. The string is passed to the firmware on platforms that support it.

Normally, the shutdown(8) utility is used when the system needs to be halted or restarted, giving users advance warning of their impending doom.

SEE ALSO

reboot(2), syslog(3), utmp(5), boot(8), init(8), shutdown(8), sync(8)

HISTORY

A **reboot** command appeared in Version 6 AT&T UNIX.

The **poweroff** command first appeared in NetBSD 1.5.

CAVEATS

Once the command has begun its work, stopping it before it completes will probably result in a system so crippled it must be physically reset. To prevent premature termination, the command blocks many signals early in its execution. However, nothing can defend against deliberate attempts to evade this.

This command will stop the system without running any shutdown(8) scripts. Amongst other things, this means that swapping will not be disabled so that raid(4) can shutdown cleanly. You should normally use shutdown(8) unless you are running in single user mode.

BUGS

The single user shell will ignore the SIGTERM signal. To avoid waiting for the timeout when rebooting or halting from the single user shell, you have to **exec reboot** or **exec halt**.

NAME

renice — alter priority of running processes

SYNOPSIS

renice *priority* [[**-p**] *pid ...*] [**-g** *pgrp ...*] [**-u** *user ...*]
renice -n *increment* [[**-p**] *pid ...*] [**-g** *pgrp ...*] [**-u** *user ...*]

DESCRIPTION

renice alters the scheduling priority of one or more running processes. The following *who* parameters are interpreted as process ID's, process group ID's, or user names. **renice**'ing a process group causes all processes in the process group to have their scheduling priority altered. **renice**'ing a user causes all processes owned by the user to have their scheduling priority altered. By default, the processes to be affected are specified by their process ID's.

Options supported by **renice**:

-g Force *who* parameters to be interpreted as process group ID's.

-n Instead of changing the specified processes to the given priority, interpret the following argument as an increment to be applied to the current priority of each process.

-u Force the *who* parameters to be interpreted as user names.

-p Resets the *who* interpretation to be (the default) process ID's.

For example,

```
renice +1 987 -u daemon root -p 32
```

would change the priority of process ID's 987 and 32, and all processes owned by users daemon and root.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to `PRIO_MAX` (20). (This prevents overriding administrative fiats.) The super-user may alter the priority of any process and set the priority to any value in the range `PRIO_MIN` (−20) to `PRIO_MAX`.

Useful priorities are: 0, the "base" scheduling priority; 20, the affected processes will run only when nothing at the base priority wants to; anything negative, the processes will receive a scheduling preference.

FILES

`/etc/passwd` to map user names to user ID's

SEE ALSO

`nice`(1), `getpriority`(2), `setpriority`(2)

HISTORY

The **renice** command appeared in 4.0BSD.

BUGS

Non super-users can not increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

NAME

repquota — summarize quotas for a file system

SYNOPSIS

repquota [**-g**] [**-u**] [**-v**] *filesystem ...*
repquota [**-g**] [**-u**] [**-v**] **-a**

DESCRIPTION

repquota prints a summary of the disk usage and quotas for the specified file systems.

Available options:

-a	Print the quotas of all the filesystems listed in `/etc/fstab`.
-g	Print only group quotas (the default is to print both group and user quotas if they exist).
-u	Print only user quotas (the default is to print both group and user quotas if they exist).
-v	Print a header line before printing each filesystem quotas.

For each user or group, the current number files and amount of space (in kilobytes) is printed, along with any quotas created with `edquota`(8).

Only members of the operator group or the super-user may use this command.

FILES

`quota.user`	at the filesystem root with user quotas
`quota.group`	at the filesystem root with group quotas
`/etc/fstab`	for file system names and locations

DIAGNOSTICS

Various messages about inaccessible files; self-explanatory.

SEE ALSO

`quota`(1), `quotactl`(2), `fstab`(5), `edquota`(8), `quotacheck`(8), `quotaon`(8)

HISTORY

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

NAME

resize_ffs — resize an on-disk file system

SYNOPSIS

resize_ffs *file-system-raw-device newsize*

DESCRIPTION

resize_ffs resizes a file system on disk. *file-system-raw-device* is the name of the raw disk device where the file system resides; *newsize* is the desired new file system size, in sectors. (Sectors are almost always 512 bytes, and **resize_ffs** can both grow and shrink file systems. When growing, the disk device must of course be large enough to contain the new file system; **resize_ffs** simply extends the file system data structures into the new space. When shrinking, **resize_ffs** assumes this. It will not work correctly for file systems with other sector sizes.) **resize_ffs** has to copy anything that currently resides in the space being shrunk away; there must be enough space free on the file system for this to succeed. If there isn't, **resize_ffs** will complain and exit; when this happens, it attempts to always leave the file system in a consistent state, but it is probably a good idea to check the file system with fsck(8).

WARNING

resize_ffs should still be considered experimental. It still needs to be validated with a rigorous regression test suite. *Interrupting* **resize_ffs** *may leave your file system in an inconsistent state and require a restore from backup.* It attempts to write in the proper order to avoid problems, but as it is still considered experimental, you should take great care when using it.

When **resize_ffs** is applied to a consistent file system, it should always produce a consistent file system; if the file system is not consistent to start with, **resize_ffs** may misbehave, anything from dumping core to completely curdling the data. It's probably wise to fsck(8) the file system before and after, just to be safe.

There is a bug somewhere in fsck; it does not check certain data structures enough. A past version of this program had a bug that produced corrupted rotation layout summary tables, which would panic the kernel. This bug is believed fixed, and there are currently no known bugs in the program. However, you should be aware that just because fsck is happy with the file system does not mean it is intact.

EXAMPLES

```
resize_ffs /dev/rsd1e 29574
```

SEE ALSO

fs(5), fsck(8), newfs(8)

HISTORY

The **resize_ffs** command first appeared in NetBSD 2.0.

AUTHORS

der Mouse ⟨mouse@rodents.montreal.qc.ca⟩

A big bug-finding kudos goes to John Kohl for finding the rotational layout bug referred to in the **WARNING** section above.

BUGS

Has not been tested and probably won't work on opposite-endian file systems.

Can fail to shrink a file system when there actually is enough space, because it does not distinguish between a block allocated as a block and a block fully occupied by two or more frags. This is unlikely to occur in practice; except for pathological cases, it can happen only when the new size is extremely close to the minimum possible.

Has no intelligence whatever when it comes to allocating blocks to copy data into when shrinking.

NAME

resize_lfs — resize a mounted log-structured filesystem

SYNOPSIS

resize_lfs [**-v**] [**-s** *new-size*] *mounted-file-system*

DESCRIPTION

resize_lfs grows or shrinks a mounted log-structured filesystem to the specified size. *mounted-file-system* is the name of the filesystem to be resized, and *new-size* is the desired new filesystem size, in sectors. If *new-size* is not specified, **resize_lfs** will default to the current size of the partition containing the filesystem in question.

When growing, the partition must be large enough to contain a filesystem of the specified size; when shrinking, **resize_lfs** must first "clean" the segments that will be invalid when the filesystem is shrunk. If this cleaning process results in these segments becoming redirtied, this indicates that the given new size is not large enough to contain the existing filesystem data, and **resize_lfs** will return an error.

EXAMPLES

To resize the file system mounted at `/home` to 32576 sectors:

```
resize_lfs -s 32576 /home
```

SEE ALSO

fsck_lfs(8), lfs_cleanerd(8), newfs_lfs(8)

HISTORY

The **resize_lfs** command first appeared in NetBSD 3.0.

AUTHORS

Konrad Schroder ⟨perseant@NetBSD.org⟩

BUGS

resize_lfs should be able to resize an unmounted filesystem as well.

NAME

restore, **rrestore** — restore files or file systems from backups made with dump

SYNOPSIS

restore −i [**−cdhmuvyN**] [**−b** *bsize*] [**−D** *algorithm*] [**−f** *file*] [**−M** *mfile*]
[**−s** *fileno*]
restore −R [**−cduvyN**] [**−b** *bsize*] [**−D** *algorithm*] [**−f** *file*] [**−M** *mfile*]
[**−s** *fileno*]
restore −r [**−cduvyN**] [**−b** *bsize*] [**−D** *algorithm*] [**−f** *file*] [**−M** *mfile*]
[**−s** *fileno*]
restore −t [**−cdhuvy**] [**−b** *bsize*] [**−f** *file*] [**−s** *fileno*] [*file ...*]
restore −x [**−cdhmuvyN**] [**−b** *bsize*] [**−D** *algorithm*] [**−f** *file*] [**−M** *mfile*]
[**−s** *fileno*] [*file ...*]

(The 4.3BSD option syntax is implemented for backward compatibility, but is not documented here.)

DESCRIPTION

The **restore** command performs the inverse function of dump(8). A full backup of a file system may be restored and subsequent incremental backups layered on top of it. Single files and directory subtrees may be restored from full or partial backups. **restore** works across a network; to do this see the **−f** flag described below. Other arguments to the command are file or directory names specifying the files that are to be restored. Unless the **−h** flag is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

If any file arguments are given with the **−x** flag, or specified in the command shell with the **−i** flag, the permissions of the root directory *will not* be applied to the current directory, unless one of those file arguments explicitly represents the root inode (e.g.: a literal '.'). This is a change from the traditional behaviour, which used to be to always prompt the user.

Exactly one of the following flags is required:

−i This mode allows interactive restoration of files from a dump. After reading in the directory information from the dump, **restore** provides a shell like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

add [*arg*]
The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, then it and all its descendants are added to the extraction list (unless the **−h** flag is specified on the command line). Files that are on the extraction list are prepended with a "∗" when they are listed by **ls**.

cd *arg* Change the current working directory to the specified argument.

delete [*arg*]
The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, then it and all its descendants are deleted from the extraction list (unless the **−h** flag is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract All the files that are on the extraction list are extracted from the dump. **restore** will ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

help, **?** List a summary of the available commands.

ls [*arg*] List the current or specified directory. Entries that are directories are appended with a "/". Entries that have been marked for extraction are prepended with a "*". If the verbose flag is set the inode number of each entry is also listed.

pwd Print the full pathname of the current working directory.

quit, **xit**
Restore immediately exits, even if the extraction list is not empty.

setmodes All the directories that have been added to the extraction list have their owner, modes, and times set; nothing is extracted from the dump. This is useful for cleaning up after a restore has been prematurely aborted.

verbose The sense of the **−v** flag is toggled. When set, the verbose flag causes the **ls** command to list the inode numbers of all entries. It also causes **restore** to print out information about each file as it is extracted.

what List dump header information.

Debug Enable debugging.

−R **restore** requests a particular tape of a multi volume set on which to restart a full restore (see the **−r** flag below). This is useful if the restore has been interrupted.

−r Restore (rebuild a file system). The target file system should be made pristine with newfs(8), mounted and the user cd(1)'d into the pristine file system before starting the restoration of the initial level 0 backup. If the level 0 restores successfully, the **−r** flag may be used to restore any necessary incremental backups on top of the level 0. The **−r** flag precludes an interactive file extraction and can be detrimental to one's health if not used carefully (not to mention the disk). An example:

```
newfs /dev/rsd0g
mount /dev/sd0g /mnt
cd /mnt

restore rf /dev/rst0
```

Note that **restore** leaves a file restoresymtable in the root directory to pass information between incremental restore passes. This file should be removed when the last incremental has been restored.

restore, in conjunction with newfs(8) and dump(8), may be used to modify file system parameters such as size or block size.

−t The names of the specified files are listed if they occur on the backup. If no file argument is given, then the root directory is listed, which results in the entire content of the backup being listed, unless the **−h** flag has been specified. Note that the **−t** flag replaces the function of the old **dumpdir** program.

−x The named files are read from the given media. If a named file matches a directory whose contents are on the backup and the **−h** flag is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, then the root directory is extracted, which results in the entire content of the backup being extracted, unless the **−h** flag has been specified.

The following additional options may be specified:

−b *bsize*
The number of kilobytes per dump record. If the **−b** option is not specified, **restore** tries to determine the block size dynamically.

−c Normally, **restore** will try to determine dynamically whether the dump was made from an old (pre-4.4) or new format file system. The **−c** flag disables this check, and only allows reading a dump in the old format.

−D *algorithm*
Computes the digest of each regular files using the *algorithm* and output to standard output. The *algorithm* is one of *md5*, *rmd160*, or *sha1*. This option doesn't imply **−N**.

−d Enable debugging.

−f *file*
Read the backup from *file*; *file* may be a special device file like `/dev/rst0` (a tape drive), `/dev/rsd1c` (a disk drive), an ordinary file, or '**−**' (the standard input). If the name of the file is of the form "host:file", or "user@host:file", **restore** reads from the named file on the remote host using rmt(8). If the name of the file is '**−**', **restore** reads from standard input. Thus, dump(8) and **restore** can be used in a pipeline to dump and restore a file system with the command

```
dump 0f - /usr | (cd /mnt; restore xf -)
```

−h Extract the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the dump.

−M *mfile*
Do not set the file flags on restore. Instead, append an mtree(8) specification to *mfile*, which can be used to restore file flags with a command such as

```
sort mfile | mtree -e -i -u
```

−m Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted, and one wants to avoid regenerating the complete pathname to the file.

−N Do not perform actual writing to disk.

−s *fileno*
Read from the specified *fileno* on a multi-file tape. File numbering starts at 1.

−u The **−u** (unlink) flag removes files before extracting them. This is useful when an executable file is in use. Ignored if **−t** or **−N** flag is given.

−v Normally **restore** does its work silently. The **−v** (verbose) flag causes it to type the name of each file it treats preceded by its file type.

−y Do not ask the user whether to abort the restore in the event of an error. Always try to skip over the bad block(s) and continue.

ENVIRONMENT

If the following environment variable exists it will be used by **restore**:

TMPDIR
The directory given in TMPDIR will be used instead of `/tmp` to store temporary files. Refer to environ(7) for more information.

FILES

`/dev/nrst0`	default tape unit to use. Taken from `_PATH_DEFTAPE` in `/usr/include/paths.h`.
`/dev/rst*`	raw SCSI tape interface
`/tmp/rstdir*`	file containing directories on the tape.
`/tmp/rstmode*`	owner, mode, and time stamps for directories.
`./restoresymtable`	information passed between incremental restores.

DIAGNOSTICS

Complains if it gets a read error. If **-y** has been specified, or the user responds 'y', **restore** will attempt to continue the restore.

If a backup was made using more than one tape volume, **restore** will notify the user when it is time to mount the next volume. If the **-x** or **-i** flag has been specified, **restore** will also ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

There are numerous consistency checks that can be listed by **restore**. Most checks are self-explanatory or can "never happen". Common errors are given below.

Converting to new file system format.
: A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

<filename>: not found on tape
: The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file <inumber>, got <inumber>
: A file that was not listed in the directory showed up. This can occur when using a dump created on an active file system.

Incremental dump too low
: When doing incremental restore, a dump that was written before the previous incremental dump, or that has too low an incremental level has been loaded.

Incremental dump too high
: When doing incremental restore, a dump that does not begin its coverage where the previous incremental dump left off, or that has too high an incremental level has been loaded.

Tape read error while restoring <filename>
Tape read error while skipping over inode <inumber>
Tape read error while trying to resynchronize
: A tape (or other media) read error has occurred. If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync restore, skipped <num> blocks
: After a dump read error, **restore** may have to resynchronize itself. This message lists the number of blocks that were skipped over.

SEE ALSO

`rcmd`(1), `rcmd`(3), `environ`(7), `dump`(8), `mount`(8), `newfs`(8), `rmt`(8)

HISTORY

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

BUGS

restore can get confused when doing incremental restores from dumps that were made on active file systems.

A level zero dump must be done after a full restore. Because **restore** runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the content of the files is unchanged.

The temporary files `/tmp/rstdir*` and `/tmp/rstmode*` are generated with a unique name based on the date of the dump and the process ID (see `mktemp`(3)), except for when **−r** or **−R** is used. Because **−R** allows you to restart a **−r** operation that may have been interrupted, the temporary files should be the same across different processes. In all other cases, the files are unique because it is possible to have two different dumps started at the same time, and separate operations shouldn't conflict with each other.

NAME

revnetgroup — generate reverse netgroup data

SYNOPSIS

revnetgroup [**−uh**] [**−f** *netgroup_file*]

DESCRIPTION

revnetgroup processes the contents of a file in netgroup(5) format into what is called reverse netgroup form. That is, where the original file shows netgroup memberships in terms of which members reside in a particular group, the reverse netgroup format specifies what groups are associated with a particular member. This information is used to generate the netgroup.byuser and netgroup.byhosts NIS maps. These reverse netgroup maps are used to help speed up netgroup lookups, particularly for the **innetgr**() library function.

For example, the standard /etc/netgroup file may list a netgroup and a list of its members. Here, the netgroup is considered the key and the member names are the data. By contrast, the reverse netgroup.byusers database lists each unique member as the key and the netgroups to which the members belong become the data. Separate databases are created to hold information pertaining to users and hosts; this allows netgroup username lookups and netgroup hostname lookups to be performed using independent keyspaces.

By constructing these reverse netgroup databases (and the corresponding NIS maps) in advance, the getnetgrent(3) library functions are spared from having to work out the dependencies themselves on the fly. This is important on networks with large numbers of users and hosts, since it can take a considerable amount of time to process very large netgroup databases.

The **revnetgroup** command prints its results on the standard output. It is usually called only by /var/yp/<domain>/Makefile when rebuilding the NIS netgroup maps.

OPTIONS

The **revnetgroup** command supports the following options:

−u Generate netgroup.byuser output; only username information in the original netgroup file is processed.

−h Generate netgroup.byhost output; only hostname information in the original netgroup file is processed. (Note at least one of the **−u** or **−h** flags must be specified.)

[**−f** *netgroup_file*]
The **revnetgroup** command uses /etc/netgroup as its default input file. The **−f** flag allows the user to specify an alternate input file. Specifying "-" as the input file causes **revnetgroup** to read from the standard input.

FILES

/var/yp/<domain>/Makefile	The Makefile that calls **makedbm** and **revnetgroup** to build the NIS databases.
/etc/netgroup	The default netgroup database file. This file is most often found only on the NIS master server.

SEE ALSO

getnetgrent(3), netgroup(5), makedbm(8), nis(8)

AUTHORS

Bill Paul ⟨wpaul@ctr.columbia.edu⟩

NAME

revoke — call revoke(2)

SYNOPSIS

revoke *file*

DESCRIPTION

The **revoke** utility performs the system call **revoke**(*file*).

file must be the pathname of an existing file.

EXIT STATUS

The **revoke** utility returns EXIT_SUCCESS on success and EXIT_FAILURE if an error occurs.

SEE ALSO

revoke(2)

NAME

rexecd — remote execution server

SYNOPSIS

rexecd

DESCRIPTION

rexecd is the server for the `rexec`(3) routine. The server provides remote execution facilities with authentication based on user names and passwords.

rexecd listens for service requests at the port indicated in the ‘‘exec’’ service specification; see `services`(5). When a service request is received the following protocol is initiated:

1. The server reads characters from the socket up to a NUL (‘`\0`’) byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the *stderr*. A second connection is then created to the specified port on the client's machine.
3. A NUL terminated user name of at most 16 characters is retrieved on the initial socket.
4. A NUL terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.
5. A NUL terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.
7. A NUL byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

username too long
The name is longer than 16 characters.

password too long
The password is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
No password file entry for the user name existed.

Password incorrect.
The wrong password was supplied.

No remote directory.
The `chdir`(2) to the home directory failed.

Try again.
A `fork`(2) by the server failed.

<shellname>: ...
The user's login shell could not be started. This message is returned on the connection associated with the *stderr*, and is not preceded by a flag byte.

SEE ALSO

`rexec`(3)

HISTORY

The **`rexecd`** command appeared in 4.2BSD.

BUGS

Indicating ‘‘Login incorrect’’ as opposed to ‘‘Password incorrect’’ is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.

SECURITY CONSIDERATIONS

As the passwords exchanged by the client and **`rexecd`** are not encrypted, it is *strongly* recommended that this service is not enabled.

NAME

rip6query — RIPng debugging tool

SYNOPSIS

rip6query [**-I** *interface*] [**-w** *time*] *destination*

DESCRIPTION

rip6query requests remote RIPng daemon on *destination* to dump RIPng routing information. **-I** lets you specify outgoing *interface* for the query packet, and is useful when link-local address is specified for *destination*. **-w** specifies the time in seconds to wait for the initial response from a gateway. The default value is 5 seconds.

SEE ALSO

route6d(8)

HISTORY

The **rip6query** command first appeared in WIDE Hydrangea IPv6 protocol stack kit.

NAME

rlogind — remote login server

SYNOPSIS

rlogind [**-alnL**]

DESCRIPTION

rlogind is the server for the rlogin(1) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

Options supported by **rlogind**:

-a	Ask hostname for verification.
-l	Prevent any authentication based on the user's ".rhosts" file, unless the user is logging in as the superuser.
-n	Disable keep-alive messages.
-L	Log all successful accesses to syslogd(8) as auth.info messages.

rlogind listens for service requests at the port indicated in the "login" service specification; see services(5). When a service request is received the following protocol is initiated:

1. The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection.
2. The server checks the client's source address and requests the corresponding host name (see getnameinfo(3), hosts(5) and named(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the **-a** option is given, the addresses for the hostname are requested, verifying that the name and address correspond. Normal authentication is bypassed if the address verification fails.

Once the source port and address have been checked, **rlogind** proceeds with the authentication process described in rshd(8). It then allocates a pseudo terminal (see pty(4)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the *stdin*, *stdout*, and *stderr* for a login process. The login process is an instance of the login(1) program, invoked with the **-f** option if authentication has succeeded. If automatic authentication fails, the user is prompted to log in as if on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo terminal, operating as an intermediary between the login process and the client instance of the rlogin(1) program. In normal operation, the packet protocol described in pty(4) is invoked to provide ^S/^Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, TERM; see environ(7). The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudo terminal.

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

At the end of a login session, **rlogind** invokes the ttyaction(3) facility with an action of "rlogind" and user "root" to execute site-specific commands.

DIAGNOSTICS

All initial diagnostic messages are indicated by a leading byte with a value of 1, after which any network connections are closed. If there are no errors before login(1) is invoked, a null byte is returned as in indication of success.

Try again.
A `fork`(2) by the server failed.

SEE ALSO

`login`(1), `ruserok`(3), `ttyaction`(3), `rshd`(8)

HISTORY

The **`rlogind`** command appeared in 4.2BSD.

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ‘‘open’’ environment.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol should be used.

`rlogind` intentionally rejects accesses from IPv4 mapped address on top of `AF_INET6` socket, since IPv4 mapped address complicates host-address based authentication. If you would like to accept connections from IPv4 peers, you will need to run **`rlogind`** on top of `AF_INET` socket, not `AF_INET6` socket.

NAME

rmt — remote magtape protocol module

SYNOPSIS

rmt

DESCRIPTION

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. **rmt** is normally started up with an `rexec`(3) or `rcmd`(3) call.

The **rmt** program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of:

> **A***number*\n

Number is an ASCII representation of a decimal number. Unsuccessful commands are responded to with:

> **E***error-number*\n*error-message*\n

Error-number is one of the possible error numbers described in `intro`(2) and *error-message* is the corresponding error string as printed from a call to `perror`(3). The protocol comprises the following commands, which are sent as indicated - no spaces are supplied between the command and its arguments, or between its arguments, and '`\n`' indicates that a newline should be supplied:

O*device*\n*mode*\n
: Open the specified *device* using the indicated *mode*. *Device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to `open`(2). If a device had already been opened, it is closed before a new open is performed.

C*device*\n
: Close the currently open device. The *device* specified is ignored.

L*offset*\n*whence*\n
: Perform an `lseek`(2) operation using the specified parameters. The response value is that returned from the `lseek`(2) call.

W*count*\n
: Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from the `write`(2) call.

R*count*\n
: Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. **rmt** then performs the requested `read`(2) and responds with **A***count-read*\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

I*operation*\n*count*\n
: Perform a `MTIOCOP` `ioctl`(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the `ioctl`(2) call. The return value is the *count* parameter when the operation is successful.

S
: Return the status of the open device, as obtained with a `MTIOCGET` `ioctl`(2) call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

Any other command causes **rmt** to exit.

DIAGNOSTICS

All responses are of the form described above.

SEE ALSO

rcmd(3), rexec(3), mtio(4), rdump(8), rrestore(8)

HISTORY

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

BUGS

People should be discouraged from using this for a remote file access protocol.

NAME

rndc−confgen − rndc key generation tool

SYNOPSIS

rndc−confgen [**−a**] [**−b** *keysize*] [**−c** *keyfile*] [**−h**] [**−k** *keyname*] [**−p** *port*] [**−r** *randomfile*] [**−s** *address*] [**−t** *chrootdir*] [**−u** *user*]

DESCRIPTION

rndc−confgen generates configuration files for **rndc**. It can be used as a convenient alternative to writing the *rndc.conf* file and the corresponding **controls** and **key** statements in *named.conf* by hand. Alternatively, it can be run with the **−a** option to set up a *rndc.key* file and avoid the need for a *rndc.conf* file and a **controls** statement altogether.

OPTIONS

−a

Do automatic **rndc** configuration. This creates a file *rndc.key* in */etc* (or whatever *sysconfdir* was specified as when BIND was built) that is read by both **rndc** and **named** on startup. The *rndc.key* file defines a default command channel and authentication key allowing **rndc** to communicate with **named** on the local host with no further configuration.

Running **rndc−confgen −a** allows BIND 9 and **rndc** to be used as drop−in replacements for BIND 8 and **ndc**, with no changes to the existing BIND 8 *named.conf* file.

If a more elaborate configuration than that generated by **rndc−confgen −a** is required, for example if rndc is to be used remotely, you should run **rndc−confgen** without the **−a** option and set up a *rndc.conf* and *named.conf* as directed.

−b *keysize*

Specifies the size of the authentication key in bits. Must be between 1 and 512 bits; the default is 128.

−c *keyfile*

Used with the **−a** option to specify an alternate location for *rndc.key*.

−h

Prints a short summary of the options and arguments to **rndc−confgen**.

−k *keyname*

Specifies the key name of the rndc authentication key. This must be a valid domain name. The default is **rndc−key**.

−p *port*

Specifies the command channel port where **named** listens for connections from **rndc**. The default is 953.

−r *randomfile*

Specifies a source of random data for generating the authorization. If the operating system does not provide a */dev/random* or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value *keyboard* indicates that keyboard input should be used.

−s *address*

Specifies the IP address where **named** listens for command channel connections from **rndc**. The default is the loopback address 127.0.0.1.

−t *chrootdir*

Used with the **−a** option to specify a directory where **named** will run chrooted. An additional copy of the *rndc.key* will be written relative to this directory so that it will be found by the chrooted **named**.

−u *user*

Used with the **−a** option to set the owner of the *rndc.key* file generated. If **−t** is also specified only the file in the chroot area has its owner changed.

EXAMPLES

To allow **rndc** to be used with no manual configuration, run

rndc−confgen −a

To print a sample *rndc.conf* file and corresponding **controls** and **key** statements to be manually inserted into *named.conf*, run

rndc−confgen

SEE ALSO

rndc(8), **rndc.conf**(5), **named**(8), BIND 9 Administrator Reference Manual.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

rndc − name server control utility

SYNOPSIS

rndc [**−b** *source−address*] [**−c** *config−file*] [**−k** *key−file*] [**−s** *server*] [**−p** *port*] [**−V**] [**−y** *key_id*]
{command}

DESCRIPTION

rndc controls the operation of a name server. It supersedes the **ndc** utility that was provided in old BIND releases. If **rndc** is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

rndc communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. In the current versions of **rndc** and **named** named the only supported authentication algorithm is HMAC−MD5, which uses a shared secret on each end of the connection. This provides TSIG−style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server.

rndc reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

OPTIONS

−b *source−address*
Use *source−address* as the source address for the connection to the server. Multiple instances are permitted to allow setting of both the IPv4 and IPv6 source addresses.

−c *config−file*
Use *config−file* as the configuration file instead of the default, */etc/rndc.conf*.

−k *key−file*
Use *key−file* as the key file instead of the default, */etc/rndc.key*. The key in */etc/rndc.key* will be used to authenticate commands sent to the server if the *config−file* does not exist.

−s *server*
server is the name or address of the server which matches a server statement in the configuration file for **rndc**. If no server is supplied on the command line, the host named by the default−server clause in the option statement of the configuration file will be used.

−p *port*
Send commands to TCP port *port* instead of BIND 9's default control channel port, 953.

−V
Enable verbose logging.

−y *keyid*
Use the key *keyid* from the configuration file. *keyid* must be known by named with the same algorithm and secret string in order for control message validation to succeed. If no *keyid* is specified, **rndc** will first look for a key clause in the server statement of the server being used, or if no server statement is present for that host, then the default−key clause of the options statement. Note that the configuration file contains shared secrets which are used to send authenticated control commands to name servers. It should therefore not have general read or write access.

For the complete set of commands supported by **rndc**, see the BIND 9 Administrator Reference Manual or run **rndc** without arguments to see its help message.

LIMITATIONS

rndc does not yet support all the commands of the BIND 8 **ndc** utility.

There is currently no way to provide the shared secret for a **key_id** without using the configuration file.

Several error messages could be clearer.

SEE ALSO

rndc.conf(5), **named**(8), **named.conf**(5) **ndc**(8), BIND 9 Administrator Reference Manual.

AUTHOR

Internet Systems Consortium

COPYRIGHT

NAME

rndctl — in-kernel random number generator management tool

SYNOPSIS

rndctl **−CcEe** [**−d** *devname*] [**−t** *devtype*]
rndctl **−ls** [**−d** *devname*] [**−t** *devtype*]

DESCRIPTION

The **rndctl** program displays statistics on the current state of the rnd(4) pseudo-driver, and allows the administrator to control which sources are allowed to contribute to the randomness pool maintained by rnd(4), as well as whether a given source counts as strongly random.

The following options are available:

−C Disable collection of timing information for the given device name or device type.

−c Enable collection of timing information for the given device name of device type.

−d Only the device named *devname* is altered or displayed. See also **−t**.

−E Disable entropy estimation from the collected timing information for the given device name or device type. If collection is still enabled, timing information is still collected and mixed into the internal entropy pool, but no entropy is assumed to be present.

−e Enable entropy estimation using the collected timing information for the given device name or device type.

−l List all sources, or, if the **−t** or **−d** flags are specified, only those specified by the *devtype* or *devname* specified.

−s Display statistics on the current state of the random collection pool.

−t All devices of type *devtype* are altered or displayed. See also **−d**.

FILES

/dev/random	Returns "good" values only.
/dev/urandom	Always returns data, degenerates to a pseudo-random generator.

SEE ALSO

rnd(4), rnd(9)

HISTORY

The **rndctl** program was first made available in NetBSD 1.3.

AUTHORS

The **rndctl** program was written by Michael Graff ⟨explorer@flame.org⟩.

BUGS

Turning on entropy estimation from unsafe or predictable sources will weaken system security, while turning on entropy collection from such sources may weaken system security.

Care should be taken when using this command.

NAME

route — manually manipulate the routing tables

SYNOPSIS

route [**−fnqSsv**] *command* [[*modifiers*] *args*]

DESCRIPTION

route is a utility used to manually manipulate the network routing tables. Except for setting up the default route, it is normally not needed, as a system routing table management daemon such as routed(8), should tend to this task.

route can be used to modify nearly any aspect of the routing policy, except packet forwarding, which can be manipulated through the sysctl(8) command.

The **route** utility supports a limited number of general options, but a rich command language, enabling the user to specify any arbitrary request that could be delivered via the programmatic interface discussed in route(4).

−f Remove all routes (as per **flush**). If used in conjunction with the **add**, **change**, **delete** or **get** commands, **route** removes the routes before performing the command.

−n Bypasses attempts to print host and network names symbolically when reporting actions. (The process of translating between symbolic names and numerical equivalents can be quite time consuming, and may require correct operation of the network; thus it may be expedient to forgo this, especially when attempting to repair networking operations).

−q Suppress all output from commands that manipulate the routing table.

−S Print a space when a flag is missing so that flags are vertically aligned instead of printing the flags that are set as a contiguous string.

−s (short) Suppresses all output from a **get** command except for the actual gateway that will be used. How the gateway is printed depends on the type of route being looked up.

−v (verbose) Print additional details.

The **route** utility provides several commands:

add	Add a route.
flush	Remove all routes.
flushall	Remove all routes including the default gateway.
delete	Delete a specific route.
change	Change aspects of a route (such as its gateway).
get	Lookup and display the route for a destination.
show	Print out the route table similar to "netstat −r" (see netstat(1)).
monitor	Continuously report any changes to the routing information base, routing lookup misses, or suspected network partitionings.

The monitor command has the syntax

route [**−n**] **monitor**

The flush command has the syntax

route [**−n**] **flush** [*family*]

If the **flush** command is specified, **route** will "flush" the routing tables of all gateway entries. When the address family is specified by any of the **−osi**, **−xns**, **−atalk**, **−inet**, or **−inet6** modifiers, only routes having destinations with addresses in the delineated family will be manipulated.

The other commands have the following syntax:

> **route** [**−n**] *command* [**−net** | **−host**] *destination gateway*

where *destination* is the destination host or network, and *gateway* is the next-hop intermediary via which packets should be routed. Routes to a particular host may be distinguished from those to a network by interpreting the Internet address specified as the *destination* argument. The optional modifiers **−net** and **−host** force the destination to be interpreted as a network or a host, respectively. Otherwise, if the *destination* has a "local address part" of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. Optionally, the *destination* can also be specified in the *net*/*bits* format.

For example, 128.32 is interpreted as **−host** 128.0.0.32; 128.32.130 is interpreted as **−host** 128.32.0.130; **−net** 128.32 is interpreted as 128.32.0.0; and **−net** 128.32.130 is interpreted as 128.32.130.0.

The keyword **default** can be used as the *destination* to set up a default route to a smart *gateway*. If no other routes match, this default route will be used as a last resort.

If the destination is directly reachable via an interface requiring no intermediary system to act as a gateway, the **−interface** modifier should be specified; the gateway given is the address of this host on the common network, indicating the interface to be used for transmission.

The optional modifiers **−xns**, **−osi**, **−atalk**, and **−link** specify that all subsequent addresses are in the XNS, OSI, or AppleTalk address families, or are specified as link-level addresses, and the names must be numeric specifications rather than symbolic names.

The optional **−netmask** qualifier is intended to achieve the effect of an OSI ESIS redirect with the netmask option, or to manually add subnet routes with netmasks different from that of the implied network interface (as would otherwise be communicated using the OSPF or ISIS routing protocols). One specifies an additional ensuing address parameter (to be interpreted as a network mask). The implicit network mask generated in the AF_INET case can be overridden by making sure this option follows the destination parameter. **−prefixlen** is also available for similar purpose, in IPv4 and IPv6 case.

Routes have associated flags which influence operation of the protocols when sending to destinations matched by the routes. These flags may be set (or sometimes cleared) by indicating the following corresponding modifiers:

```
-cloning    RTF_CLONING    - generates a new route on use
-nocloning ~RTF_CLONING    - stop generating new routes on use
-cloned     RTF_CLONED     - cloned route generated by RTF_CLONING
-nocloned  ~RTF_CLONED     - prevent removal with RTF_CLONING parent
-xresolve   RTF_XRESOLVE   - emit mesg on use (for external lookup)
-iface     ~RTF_GATEWAY    - destination is directly reachable
-static     RTF_STATIC     - manually added route
-nostatic  ~RTF_STATIC     - pretend route added by kernel or daemon
-reject     RTF_REJECT     - emit an ICMP unreachable when matched
-blackhole  RTF_BLACKHOLE  - silently discard pkts (during updates)
-proto1     RTF_PROTO1     - set protocol specific routing flag #1
-proto2     RTF_PROTO2     - set protocol specific routing flag #2
-llinfo     RTF_LLINFO     - validly translates proto addr to link addr
```

The optional modifiers **−rtt**, **−rttvar**, **−sendpipe**, **−recvpipe**, **−mtu**, **−hopcount**, **−expire**, and **−ssthresh** provide initial values to quantities maintained in the routing entry by transport level protocols, such as TCP or TP4. These may be individually locked by preceding each such modifier to be locked by the **−lock** meta-modifier, or one can specify that all ensuing metrics may be locked by the **−lockrest** meta-modifier.

In a **change** or **add** command where the destination and gateway are not sufficient to specify the route (as in the ISO case where several interfaces may have the same address), the **−ifp** or **−ifa** modifiers may be used to determine the interface or interface address.

All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using gethostbyname(3). If this lookup fails, getnetbyname(3) is then used to interpret the name as that of a network.

route uses a routing socket and the new message types RTM_ADD, RTM_DELETE, RTM_GET, and RTM_CHANGE. As such, only the super-user may modify the routing tables.

EXIT STATUS

The **route** utility exits 0 on success, and >0 if an error occurs. This includes the use of the **get** command to look up a route that is incomplete.

EXAMPLES

This sets the default route to 192.168.0.1:

```
route add default 192.168.0.1
```

This shows all routes, without DNS resolution (this is useful if the DNS is not available):

```
route -n show
```

To install a static route through 10.200.0.1 to reach the network 192.168.1.0/28, use this:

```
route add -net 192.168.1.0 -netmask 255.255.255.240 10.200.0.1
```

DIAGNOSTICS

add [host | network] %s: gateway %s flags %x
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the ioctl(2) call. If the gateway address used was not the primary address of the gateway (the first one returned by gethostbyname(3)), the gateway address is printed numerically as well as symbolically.

delete [host | network] %s: gateway %s flags %x
As above, but when deleting an entry.

%s %s done
When the **flush** command is specified, each routing table entry deleted is indicated with a message of this form.

Network is unreachable
An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

not in table
A delete operation was attempted for an entry which wasn't present in the tables.

routing table overflow
An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

Permission denied
The attempted operation is privileged. Only root may modify the routing tables. These privileges are enforced by the kernel.

SEE ALSO

esis(4), netintro(4), route(4), routed(8), sysctl(8)

HISTORY

The **route** command appeared in 4.2BSD. IPv6 support was added by WIDE/KAME project.

BUGS

The first paragraph may have slightly exaggerated `routed`(8)’s abilities.

Some uses of the **-ifa** or **-ifp** modifiers with the add command will incorrectly fail with a “Network is unreachable” message if there is no default route. See case `RTM_ADD` in `sys/net/rtsock.c:route_output` for details.

NAME

route6d — RIP6 routing daemon

SYNOPSIS

route6d [**-adDhlnqsS**] [**-R** *routelog*] [**-A** *prefix/preflen,if1[,if2...]*] [**-L** *prefix/preflen,if1[,if2...]*] [**-N** *if1[,if2...]*] [**-O** *prefix/preflen,if1[,if2...]*] [**-T** *if1[,if2...]*] [**-t** *tag*]

DESCRIPTION

The **route6d** is a routing daemon which supports RIP over IPv6.

Options are:

-a Enables aging of the statically defined routes. With this option, any statically defined routes will be removed unless corresponding updates arrive as if the routes are received at the startup of **route6d**.

-R *routelog*
This option makes **route6d** log route changes (add/delete) to the file *routelog*.

-A *prefix/preflen,if1[,if2...]*
This option is used for aggregating routes. *prefix/preflen* specifies the prefix and the prefix length of the aggregated route. When advertising routes, **route6d** filters specific routes covered by the aggregate and advertises the aggregated route *prefix/preflen* to the interfaces specified in the comma-separated interface list *if1[,if2...]*. **route6d** creates a static route to *prefix/preflen*, with the RTF_REJECT flag set, into the kernel routing table.

-d Enables output of debugging messages. This option also instructs **route6d** to run in foreground mode (i.e., it does not become a daemon process) .

-D Enables extensive output of debugging messages. This option also instructs **route6d** to run in foreground mode (i.e., it does not become a daemon process) .

-h Disables split horizon processing.

-l By default, **route6d** will not exchange site local routes for safety reasons. This is because the semantics of site local address space are rather vague, as the specification is still being worked on, and there is no good way to define the site local boundary. With **-l**, **route6d** will exchange site local routes as well. It must not be used on site boundary routers, since **-l** assumes that all interfaces are in the same site.

-L *prefix/preflen,if1[,if2...]*
Filter incoming routes from interfaces *if1,[if2...]*. **route6d** will accept incoming routes that are in *prefix/preflen*. If multiple **-L** options are specified, all routes that match any of the options are accepted. `::/0` is treated specially as default route, not "any route that has longer prefix length than, or equal to 0". If you would like to accept any route, specify no **-L** option. For example, with "**-L** `3ffe::/16,if1` **-L** `::/0,if1`" **route6d** will accept the default route and routes in the 6bone test address range, but no others.

-n Do not update the kernel routing table.

-N *if1[,if2...]*
Do not listen to, or advertise, route from/to interfaces specified by *if1,[if2...]*.

-O *prefix/preflen,if1[,if2...]*
Restrict route advertisement toward interfaces specified by *if1,[if2...]*. With this option **route6d** will only advertise routes that match *prefix/preflen*.

-q Makes **route6d** use listen-only mode. No advertisement is sent.

-s Makes **route6d** advertise the statically defined routes which exist in the kernel routing table when **route6d** is invoked. Announcements obey the regular split horizon rule.

-S This option is the same as **-s**, except that the split horizon rule does apply.

-T *if1[,if2...]*
Advertise only the default route toward *if1,[if2...]*.

-t *tag*
Attach the route tag *tag* to originated route entries. *tag* can be decimal, octal prefixed by `0`, or hexadecimal prefixed by `0x`.

Upon receipt of signal `SIGINT` or `SIGUSR1`, **route6d** will dump the current internal state into `/var/run/route6d_dump`.

FILES

`/var/run/route6d_dump` contains the internal state dumps created if **route6d** receives a `SIGINT` or `SIGUSR1` signal

SEE ALSO

G. Malkin and R. Minnear, *RIPng for IPv6*, RFC 2080, January 1997.

NOTES

route6d uses the advanced IPv6 API, defined in RFC 3542, for communicating with peers using link-local addresses.

Internally **route6d** embeds interface identifiers into bits 32 to 63 of link-local addresses (`fe80::xx` and `ff02::xx`) so they will be visible in the internal state dump file (`/var/run/route6d_dump`).

Routing table manipulation differs from IPv6 implementation to implementation. Currently **route6d** obeys the WIDE Hydrangea/KAME IPv6 kernel, and will not be able to run on other platforms.

Currently, **route6d** does not reduce the rate of the triggered updates when consecutive updates arrive.

NAME

routed, **rdisc** — network RIP and router discovery routing daemon

SYNOPSIS

routed [**−sqdghmAtv**] [**−T** *tracefile*] [**−F** *net*[/mask[,metric]]] [**−P** *parms*]

DESCRIPTION

routed is a daemon invoked at boot time to manage the network routing tables. It uses Routing Information Protocol, RIPv1 (RFC 1058), RIPv2 (RFC 1723), and Internet Router Discovery Protocol (RFC 1256) to maintain the kernel routing table. The RIPv1 protocol is based on the reference 4.3BSD daemon.

It listens on the udp(4) socket for the route(8) service (see services(5)) for Routing Information Protocol packets. It also sends and receives multicast Router Discovery ICMP messages. If the host is a router, **routed** periodically supplies copies of its routing tables to any directly connected hosts and networks. It also advertises or solicits default routes using Router Discovery ICMP messages.

When started (or when a network interface is later turned on), **routed** uses an AF_ROUTE address family facility to find those directly connected interfaces configured into the system and marked "up". It adds necessary routes for the interfaces to the kernel routing table. Soon after being first started, and provided there is at least one interface on which RIP has not been disabled, **routed** deletes all pre-existing non-static routes in kernel table. Static routes in the kernel table are preserved and included in RIP responses if they have a valid RIP metric (see route(8)).

If more than one interface is present (not counting the loopback interface), it is assumed that the host should forward packets among the connected networks. After transmitting a RIP *request* and Router Discovery Advertisements or Solicitations on a new interface, the daemon enters a loop, listening for RIP request and response and Router Discovery packets from other hosts.

When a *request* packet is received, **routed** formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16 or greater is considered "infinite"). The advertised metric for a route reflects the metrics associated with interfaces (see ifconfig(8)) through which it is received and sent, so setting the metric on an interface is an effective way to steer traffic. See also **adj_inmetric** and **adj_outmetric** parameters below.

Responses do not include routes with a first hop on the requesting network to implement in part *split-horizon*. Requests from query programs such as rtquery(8) are answered with the complete table.

The routing table maintained by the daemon includes space for several gateways for each destination to speed recovery from a failing router. RIP *response* packets received are used to update the routing tables provided they are from one of the several currently recognized gateways or advertise a better metric than at least one of the existing gateways.

When an update is applied, **routed** records the change in its own tables and updates the kernel routing table if the best route to the destination changes. The change in the kernel routing table is reflected in the next batch of *response* packets sent. If the next response is not scheduled for a while, a *flash update* response containing only recently changed routes is sent.

In addition to processing incoming packets, **routed** also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed until the route has been advertised with an infinite metric to ensure the invalidation is propagated throughout the local internet. This is a form of *poison reverse*.

Routes in the kernel table that are added or changed as a result of ICMP Redirect messages are deleted after a while to minimize *black-holes*. When a TCP connection suffers a timeout, the kernel tells **routed**, which deletes all redirected routes through the gateway involved, advances the age of all RIP routes through the gateway to allow an alternate to be chosen, and advances of the age of any relevant Router Discovery Protocol default routes.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. These RIP responses are sent to the broadcast address on nets that support broadcasting, to the destination address on point-to-point links, and to the router's own address on other networks. If RIPv2 is enabled, multicast packets are sent on interfaces that support multicasting.

If no response is received on a remote interface, if there are errors while sending responses, or if there are more errors than input or output (see `netstat`(1)), then the cable or some other part of the interface is assumed to be disconnected or broken, and routes are adjusted appropriately.

The *Internet Router Discovery Protocol* is handled similarly. When the daemon is supplying RIP routes, it also listens for Router Discovery Solicitations and sends Advertisements. When it is quiet and listening to other RIP routers, it sends Solicitations and listens for Advertisements. If it receives a good Advertisement and it is not multi-homed, it stops listening for broadcast or multicast RIP responses. It tracks several advertising routers to speed recovery when the currently chosen router dies. If all discovered routers disappear, the daemon resumes listening to RIP responses. It continues listening to RIP while using Router Discovery if multi-homed to ensure all interfaces are used.

The Router Discovery standard requires that advertisements have a default "lifetime" of 30 minutes. That means should something happen, a client can be without a good route for 30 minutes. It is a good idea to reduce the default to 45 seconds using **−P rdisc_interval=45** on the command line or **rdisc_interval=45** in the `/etc/gateways` file.

While using Router Discovery (which happens by default when the system has a single network interface and a Router Discover Advertisement is received), there is a single default route and a variable number of redirected host routes in the kernel table. On a host with more than one network interface, this default route will be via only one of the interfaces. Thus, multi-homed hosts running with **−q** might need **no_rdisc** described below.

See the **pm_rdisc** facility described below to support "legacy" systems that can handle neither RIPv2 nor Router Discovery.

By default, neither Router Discovery advertisements nor solicitations are sent over point to point links (e.g. PPP). The netmask associated with point-to-point links (such as SLIP or PPP, with the IFF_POINTOPOINT flag) is used by **routed** to infer the netmask used by the remote system when RIPv1 is used.

The following options are available:

−s force **routed** to supply routing information. This is the default if multiple network interfaces are present on which RIP or Router Discovery have not been disabled, and if the sysctl net.inet.ip.forwarding=1.

−q is the opposite of the **−s** option. This is the default when only one interface is present. With this explicit option, the daemon is always in "quiet-mode" for RIP and does not supply routing information to other computers.

−d do not run in the background. This option is meant for interactive use.

−g used on internetwork routers to offer a route to the "default" destination. It is equivalent to **−F 0/0,1** and is present mostly for historical reasons. A better choice is **−P pm_rdisc** on the command line or **pm_rdisc** in the `/etc/gateways` file. since a larger metric will be used, reducing the spread of the potentially dangerous default route. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers. Notice that because a metric of 1 is used, this feature is dangerous. It is more commonly accidentally used to create chaos with a routing loop than to solve problems.

-h cause host or point-to-point routes to not be advertised, provided there is a network route going the same direction. That is a limited kind of aggregation. This option is useful on gateways to Ethernets that have other gateway machines connected with point-to-point links such as SLIP.

-m cause the machine to advertise a host or point-to-point route to its primary interface. It is useful on multi-homed machines such as NFS servers. This option should not be used except when the cost of the host routes it generates is justified by the popularity of the server. It is effective only when the machine is supplying routing information, because there is more than one interface. The **-m** option overrides the **-q** option to the limited extent of advertising the host route.

-A do not ignore RIPv2 authentication if we do not care about RIPv2 authentication. This option is required for conformance with RFC 1723. However, it makes no sense and breaks using RIP as a discovery protocol to ignore all RIPv2 packets that carry authentication when this machine does not care about authentication.

-t increase the debugging level, which causes more information to be logged on the tracefile specified with **-T** or standard out. The debugging level can be increased or decreased with the *SIGUSR1* or *SIGUSR2* signals or with the `rtquery`(8) command.

-T *tracefile*
increases the debugging level to at least 1 and causes debugging information to be appended to the trace file. Note that because of security concerns, it is wisest to not run **routed** routinely with tracing directed to a file.

-v displays and logs the version of daemon.

-F *net[/mask][,metric]*
minimize routes in transmissions via interfaces with addresses that match *net/mask*, and synthesizes a default route to this machine with the *metric*. The intent is to reduce RIP traffic on slow, point-to-point links such as PPP links by replacing many large UDP packets of RIP information with a single, small packet containing a "fake" default route. If *metric* is absent, a value of 14 is assumed to limit the spread of the "fake" default route. This is a dangerous feature that when used carelessly can cause routing loops. Notice also that more than one interface can match the specified network number and mask. See also **-g**.

-P *parms*
is equivalent to adding the parameter line *parms* to the `/etc/gateways` file.

Any other argument supplied is interpreted as the name of a file in which the actions of **routed** should be logged. It is better to use **-T** instead of appending the name of the trace file to the command.

routed also supports the notion of "distant" *passive* or *active* gateways. When **routed** is started, it reads the file `/etc/gateways` to find such distant gateways which may not be located using only information from a routing socket, to discover if some of the local gateways are *passive*, and to obtain other parameters. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange RIP packets. Routes through *passive* gateways are installed in the kernel's routing tables once upon startup and are not included in transmitted RIP responses.

Distant active gateways are treated like network interfaces. RIP responses are sent to the distant *active* gateway. If no responses are received, the associated route is deleted from the kernel table and RIP responses advertised via other interfaces. If the distant gateway resumes sending RIP responses, the associated route is restored.

Such gateways can be useful on media that do not support broadcasts or multicasts but otherwise act like classic shared media like Ethernets such as some ATM networks. One can list all RIP routers reachable on the HIPPI or ATM network in `/etc/gateways` with a series of "host" lines. Note that it is usually desirable to use RIPv2 in such situations to avoid generating lists of inferred host routes.

Gateways marked *external* are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to indicate that another routing process will install such a route if necessary, and that other routes to that destination should not be installed by **routed**. Such entries are only required when both routers may learn of routes to the same destination.

The `/etc/gateways` file is comprised of a series of lines, each in one of the following two formats or consist of parameters described later. Blank lines and lines starting with '#' are comments.

net *Nname[/mask]* **gateway** *Gname* **metric** *value* <**passive** | **active** | **extern**>

host *Hname* **gateway** *Gname* **metric** *value* <**passive** | **active** | **extern**>

Nname or *Hname* is the name of the destination network or host. It may be a symbolic network name or an Internet address specified in "dot" notation (see inet(3)). (If it is a name, then it must either be defined in `/etc/networks` or `/etc/hosts`, or named(8), must have been started before **routed**.)

Mask is an optional number between 1 and 32 indicating the netmask associated with *Nname*.

Gname is the name or address of the gateway to which RIP responses should be forwarded.

Value is the hop count to the destination host or network. *Host hname* is equivalent to *net nname/32* .

One of the keywords **passive**, **active** or **external** must be present to indicate whether the gateway should be treated as **passive** or **active** (as described above), or whether the gateway is **external** to the scope of the RIP protocol.

As can be seen when debugging is turned on with **-t**, such lines create pseudo-interfaces. To set parameters for remote or external interfaces, a line starting with **if=alias(Hname)**, **if=remote(Hname)**, etc. should be used.

Parameters

Lines that start with neither "net" nor "host" must consist of one or more of the following parameter settings, separated by commas or blanks:

if=*ifname*
: indicates that the other parameters on the line apply to the interface name *ifname*.

subnet=*nname[/mask][,metric]*
: advertises a route to network *nname* with mask *mask* and the supplied metric (default 1). This is useful for filling "holes" in CIDR allocations. This parameter must appear by itself on a line. The network number must specify a full, 32-bit value, as in 192.0.2.0 instead of 192.0.2.

  Do not use this feature unless necessary. It is dangerous.

ripv1_mask=*nname/mask1,mask2*
: specifies that netmask of the network of which **nname/mask1** is a subnet should be **mask2**. For example **ripv1_mask**=*192.0.2.16/28,27* marks 192.0.2.16/28 as a subnet of 192.0.2.0/27 instead of 192.0.2.0/24. It is better to turn on RIPv2 instead of using this facility, for example with **ripv2_out**.

passwd=*XXX[|KeyID[start|stop]]*
: specifies a RIPv2 cleartext password that will be included on all RIPv2 responses sent, and checked on all RIPv2 responses received. Any blanks, tab characters, commas, or '#', '|', or NULL characters in the password must be escaped with a backslash (\). The common escape sequences \n, \r, \t, \b, and \xxx have their usual meanings. The **KeyID** must be unique but is ignored for cleartext passwords. If present, **start** and **stop** are timestamps in the form year/month/day@hour:minute.

They specify when the password is valid. The valid password with the most future is used on output packets, unless all passwords have expired, in which case the password that expired most recently is used, or unless no passwords are valid yet, in which case no password is output. Incoming packets can carry any password that is valid, will be valid within the next 24 hours, or that was valid within the preceding 24 hours. To protect the secrets, the passwd settings are valid only in the */etc/gateways* file and only when that file is readable only by UID 0.

md5_passwd=*XXX|KeyID[start|stop]*
: specifies a RIPv2 MD5 password. Except that a **KeyID** is required, this keyword is similar to **passwd**.

no_ag turns off aggregation of subnets in RIPv1 and RIPv2 responses.

no_super_ag
: turns off aggregation of networks into supernets in RIPv2 responses.

passive
: marks the interface to not be advertised in updates sent via other interfaces, and turns off all RIP and router discovery through the interface.

no_rip
: disables all RIP processing on the specified interface. If no interfaces are allowed to process RIP packets, **routed** acts purely as a router discovery daemon.

Note that turning off RIP without explicitly turning on router discovery advertisements with **rdisc_adv** or **−s** causes **routed** to act as a client router discovery daemon, not advertising.

no_rip_mcast
: causes RIPv2 packets to be broadcast instead of multicast.

no_ripv1_in
: causes RIPv1 received responses to be ignored.

no_ripv2_in
: causes RIPv2 received responses to be ignored.

ripv2_out
: turns on RIPv2 output and causes RIPv2 advertisements to be multicast when possible.

ripv2 is equivalent to **no_ripv1_in** and **no_ripv1_out**. This enables RIPv2.

no_rdisc
: disables the Internet Router Discovery Protocol.

no_solicit
: disables the transmission of Router Discovery Solicitations.

send_solicit
: specifies that Router Discovery solicitations should be sent, even on point-to-point links, which by default only listen to Router Discovery messages.

no_rdisc_adv
: disables the transmission of Router Discovery Advertisements.

rdisc_adv
: specifies that Router Discovery Advertisements should be sent, even on point-to-point links, which by default only listen to Router Discovery messages.

bcast_rdisc
specifies that Router Discovery packets should be broadcast instead of multicast.

rdisc_pref=*N*
sets the preference in Router Discovery Advertisements to the optionally signed integer *N*. The default preference is 0. Default routes with larger preferences are preferred by clients.

rdisc_interval=*N*
sets the nominal interval with which Router Discovery Advertisements are transmitted to N seconds and their lifetime to 3∗N.

fake_default=*metric*
has an identical effect to **−F** *net[/mask][=metric]* with the network and mask coming from the specified interface.

pm_rdisc
is similar to **fake_default**. When RIPv2 routes are multicast, so that RIPv1 listeners cannot receive them, this feature causes a RIPv1 default route to be broadcast to RIPv1 listeners. Unless modified with **fake_default**, the default route is broadcast with a metric of 14. That serves as a "poor man's router discovery" protocol.

adj_inmetric=*delta*
adjusts the hop count or metric of received RIP routes by *delta*. The metric of every received RIP route is increased by the sum of two values associated with the interface. One is the adj_inmetric value and the other is the interface metric set with ifconfig(8).

adj_outmetric=*delta*
adjusts the hop count or metric of advertised RIP routes by *delta*. The metric of every received RIP route is increased by the metric associated with the interface by which it was received, or by 1 if the interface does not have a non-zero metric. The metric of the received route is then increased by the adj_outmetric associated with the interface. Every advertised route is increased by a total of four values, the metric set for the interface by which it was received with ifconfig(8), the **adj_inmetric** *delta* of the receiving interface, the metric set for the interface by which it is transmitted with ifconfig(8), and the **adj_outmetric** *delta* of the transmitting interface.

trust_gateway=*rname[|net1/mask1|net2/mask2|...]*
causes RIP packets from router *rname* and other routers named in other **trust_gateway** keywords to be accepted, and packets from other routers to be ignored. If networks are specified, then routes to other networks will be ignored from that router.

redirect_ok
allows the kernel to listen ICMP Redirect messages when the system is acting as a router and forwarding packets. Otherwise, ICMP Redirect messages are overridden and deleted when the system is acting as a router.

FILES

/etc/gateways for distant gateways

SEE ALSO

icmp(4), udp(4), rtquery(8)

Internet Transport Protocols, XSIS 028112, Xerox System Integration Standard.

HISTORY

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

BUGS

It does not always detect unidirectional failures in network interfaces, for example, when the output side fails.

NAME

bootparamd, **rpc.bootparamd** — boot parameter server

SYNOPSIS

bootparamd [**-ds**] [**-i** *interface*] [**-r** *router*] [**-f** *file*]

DESCRIPTION

bootparamd is a server process that provides information to diskless clients necessary for booting. It consults the file "`/etc/bootparams`". It should normally be started from "`/etc/rc`".

This version will allow the use of aliases on the hostname in the "`/etc/bootparams`" file. The hostname returned in response to the booting client's whoami request will be the name that appears in the config file, not the canonical name. In this way you can keep the answer short enough so that machines that cannot handle long hostnames won't fail during boot.

While parsing, if a line containing just "+" is found, and the YP subsystem is active, the YP map `bootparams` will be searched immediately.

OPTIONS

-d Display the debugging information. The daemon does not fork in this case.

-i *interface*
Specify the interface to become the default router. **bootparamd** picks the first IPv4 address it finds on the system by default. With **-i**, you can control which interface to be used to obtain the default router address. **-r** overrides **-i**.

-s Log the debugging information with `syslog`(3).

-r Set the default router (a hostname or IP-address). This defaults to the machine running the server.

-f Specify the file to use as boot parameter file instead of "`/etc/bootparams`".

FILES

`/etc/bootparams` default configuration file

SEE ALSO

`bootparams`(5)

AUTHORS

Originally written by Klas Heggemann ⟨klas@nada.kth.se⟩.

BUGS

You may find the syslog messages too verbose.

It's not clear if the non-canonical hack mentioned above is a good idea.

NAME

rpc.lockd — NFS file locking daemon

SYNOPSIS

rpc.lockd [**-d** *debug_level*] [**-g** *grace period*]

DESCRIPTION

The **rpc.lockd** daemon provides monitored and unmonitored file and record locking services in an NFS environment. To monitor the status of hosts requesting locks, the locking daemon typically operates in conjunction with rpc.statd(8).

Options and operands available for **rpc.lockd** :

-d The **-d** option causes debugging information to be written to syslog, recording all RPC transactions to the daemon. These messages are logged with level LOG_DEBUG and facility LOG_DAEMON. Specifying a debug_level of 1 results in the generation of one log line per protocol operation. Higher debug levels can be specified, causing display of operation arguments and internal operations of the daemon.

-g The **-g** option allow to specify the grace period, in seconds. During the grace period **rpc.lockd** only accepts requests from hosts which are reinitialising locks which existed before the server restart. Default is 30 seconds.

Error conditions are logged to syslog, irrespective of the debug level, using log level LOG_ERR and facility LOG_DAEMON.

The **rpc.lockd** daemon must NOT be invoked by inetd(8) because the protocol assumes that the daemon will run from system start time. Instead, it should be configured in rc.conf(5) to run at system startup.

FILES

/usr/include/rpcsvc/nlm_prot.x	RPC protocol specification for the network lock manager protocol.

SEE ALSO

syslog(3), rc.conf(5), rpc.statd(8)

STANDARDS

The implementation is based on the specification in X/Open CAE Specification C218, "Protocols for X/Open PC Interworking: XNFS, Issue 4", ISBN 1 872630 66 9

HISTORY

A version of **rpc.lockd** appeared in SunOS 4.

BUGS

The current implementation provides only the server side of the protocol (i.e. clients running other OS types can establish locks on a NetBSD fileserver, but there is currently no means for a NetBSD client to establish locks).

The current implementation serialises locks requests that could be shared.

NAME

rpc.rquotad, **rquotad** — remote quota server

SYNOPSIS

/usr/libexec/rpc.rquotad

DESCRIPTION

rpc.rquotad is a rpc(3) server which returns quotas for a user of a local filesystem which is NFS-mounted onto a remote machine. quota(1) uses the results to display user quotas for remote filesystems. **rpc.rquotad** is normally invoked by inetd(8).

rpc.rquotad uses an RPC protocol defined in /usr/include/rpcsvc/rquota.x.

SEE ALSO

quota(1)

NAME

rpc.rstatd, **rstatd** — kernel statistics server

SYNOPSIS

/usr/libexec/rpc.rstatd [*interval*]

DESCRIPTION

rpc.rstatd is a server which returns performance statistics obtained from the kernel. These statistics are read using the `rup`(1) command. The **rpc.rstatd** daemon is normally invoked by `inetd`(8).

The *interval* argument specifies the number of seconds that **rpc.rstatd** should stay active, updating its internal statistics every second. If no value is specified, 20 seconds will be used. After *interval* seconds with no new RPC requests, if **rpc.rstatd** was invoked from `inetd`(8), **rpc.rstatd** exits. Otherwise, **rpc.rstatd** loops, becoming dormant until it receives a new RPC request, and staying active until *interval* seconds pass with no new requests.

rpc.rstatd uses an RPC protocol defined in `/usr/include/rpcsvc/rstat.x`.

SEE ALSO

`rup`(1), `inetd`(8)

NAME

rpc.rusersd, **rusersd** — logged in users server

SYNOPSIS

/usr/libexec/rpc.rusersd

DESCRIPTION

rpc.rusersd is a server which returns information about users currently logged in to the system.

The currently logged in users are queried using the `rusers`(1) command. The **rpc.rusersd** daemon is normally invoked by `inetd`(8).

rpc.rusersd uses an RPC protocol defined in `/usr/include/rpcsvc/rnusers.x`.

SEE ALSO

`rusers`(1), `w`(1), `who`(1), `inetd`(8)

NAME

rpc.rwalld, **rwalld** — write messages to users currently logged in server

SYNOPSIS

/usr/libexec/rpc.rwalld

DESCRIPTION

rpc.rwalld is a server which will send a message to users currently logged in to the system. This server invokes the `wall`(1) command to actually write the messages to the system.

Messages are sent to this server by the `rwall`(1) command. The **rpc.rwalld** daemon is normally invoked by `inetd`(8).

rpc.rwalld uses an RPC protocol defined in `/usr/include/rpcsvc/rwall.x`.

SEE ALSO

`rwall`(1), `wall`(1), `inetd`(8)

NAME

rpc.sprayd, **sprayd** — spray server

SYNOPSIS

/usr/libexec/rpc.sprayd

DESCRIPTION

rpc.sprayd is a server which records packets sent by the spray(8) command and sends a traffic report to the originator of the packets. The **rpc.sprayd** daemon is normally invoked by inetd(8).

rpc.sprayd uses an RPC protocol defined in /usr/include/rpcsvc/spray.x.

SEE ALSO

spray(8)

SECURITY CONSIDERATIONS

As **rpc.sprayd** responds to packets generated by spray(8), remote users can cause a denial of network service against the local host by saturating requests to **rpc.sprayd**.

NAME

rpc.statd — host status monitoring daemon

SYNOPSIS

rpc.statd [**-d**]

DESCRIPTION

rpc.statd is a daemon which co-operates with rpc.statd daemons on other hosts to provide a status monitoring service. The daemon accepts requests from programs running on the local host (typically, `rpc.lockd`(8), the NFS file locking daemon) to monitor the status of specified hosts. If a monitored host crashes and restarts, the remote daemon will notify the local daemon, which in turn will notify the local program(s) which requested the monitoring service. Conversely, if this host crashes and restarts, when **rpc.statd** restarts, it will notify all of the hosts which were being monitored at the time of the crash.

Options and operands available for **rpc.statd** :

-d The **-d** option causes debugging information to be written to syslog, recording all RPC transactions to the daemon. These messages are logged with level LOG_DEBUG and facility LOG_DAEMON. Error conditions are logged irrespective of this option, using level LOG_ERR.

The **rpc.statd** daemon must NOT be invoked by `inetd`(8) because the protocol assumes that the daemon will run from system start time. Instead, it should be configured in `rc.conf`(5) to run at system startup.

FILES

`/var/db/statd.status`	non-volatile record of currently monitored hosts.
`/usr/include/rpcsvc/sm_inter.x`	RPC protocol specification used by local applications to register monitoring requests.

SEE ALSO

`syslog`(3), `rc.conf`(5), `rpc.lockd`(8)

STANDARDS

The implementation is based on the specification in X/Open CAE Specification C218, "Protocols for X/Open PC Interworking: XNFS, Issue 4", ISBN 1 872630 66 9

HISTORY

A version of **rpc.statd** appeared in SunOS 4.

BUGS

There is no means for the daemon to tell when a monitored host has disappeared permanently (e.g., catastrophic hardware failure), as opposed to transient failure of the host or an intermediate router. At present, it will retry notification attempts at frequent intervals for 10 minutes, then hourly, and finally gives up after 24 hours.

The protocol requires that symmetric monitor requests are made to both the local and remote daemon in order to establish a monitored relationship. This is convenient for the NFS locking protocol, but probably reduces the usefulness of the monitoring system for other applications.

The current implementation uses more than 1Kbyte per monitored host in the status file (and also in VM). This may be inefficient for NFS servers with large numbers of clients.

NAME

rpc.yppasswdd — NIS update password file daemon

SYNOPSIS

rpc.yppasswdd [**-d** *directory*] [**-noshell**] [**-nogecos**] [**-nopw**]
[**-m** *arg1* [*arg2 ...*]]

DESCRIPTION

rpc.yppasswdd must be running on the NIS master server to allow users to change information in the password file.

The options are as follows:

-d *directory* Change the root directory of the password file from "/" to *directory*. It is important to create the binary database files (pwd.db and spwd.db) when using this switch or the password change will fail. The databases need to be created only once with the following command:

```
pwd_mkdb -d directory directory/etc/master.passwd
```

-noshell Don't allow changes of the shell field in the passwd file.

-nogecos Don't allow changes of the gecos field in the passwd file.

-nopw Don't allow changes of the password in the passwd file.

-m *arg1* [*arg2 ...*]
Additional arguments to pass to *make* in */var/yp*.

FILES

/etc/passwd
/etc/master.passwd

SEE ALSO

chpass(1), passwd(1), hosts_access(5), nis(8), ypbind(8), ypserv(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

rpcbind — universal addresses to RPC program number mapper

SYNOPSIS

rpcbind [**-dilLs**]

DESCRIPTION

rpcbind is a server that converts RPC program numbers into universal addresses. It must be running on the host to be able to make RPC calls on a server on that machine.

When an RPC service is started, it tells **rpcbind** the address at which it is listening, and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts **rpcbind** on the server machine to determine the address where RPC requests should be sent.

rpcbind should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so **rpcbind** must be started before port monitors are invoked.

When **rpcbind** is started, it checks that certain name-to-address translation-calls function correctly. If they fail, the network configuration databases may be corrupt. Since RPC services cannot function correctly in this situation, **rpcbind** reports the condition and terminates.

rpcbind can only be started by the super-user.

Access control is provided by `/etc/hosts.allow` and `/etc/hosts.deny`, as described in `hosts_access`(5) with daemon name **rpcbind**.

OPTIONS

-d Run in debug mode. In this mode, **rpcbind** will not fork when it starts, will print additional information during operation, and will abort on certain errors. With this option, the name-to-address translation consistency checks are shown in detail.

-i "insecure" mode. Allows calls to SET and UNSET from any host. Normally **rpcbind** accepts these requests only from the loopback interface for security reasons. This change is necessary for programs that were compiled with earlier versions of the rpc library and do not make those requests using the loopback interface.

-l Turns on libwrap connection logging.

-s Causes **rpcbind** to change to the user daemon as soon as possible. This causes **rpcbind** to use non-privileged ports for outgoing connections, preventing non-privileged clients from using **rpcbind** to connect to services from a privileged port.

-L Allow old-style local connections over the loopback interface. Without this flag, local connections are only allowed over a local socket, `/var/run/rpcbind.sock`

NOTES

All RPC servers must be restarted if **rpcbind** is restarted.

FILES

`/var/run/rpcbind.sock`	
`/etc/hosts.allow`	explicit remote host access list.
`/etc/hosts.deny`	explicit remote host denial of service list.

SEE ALSO

`rpcbind`(3), `hosts_access`(5), `hosts_options`(5), `rpcinfo`(8)

NAME

rpcinfo — report RPC information

SYNOPSIS

rpcinfo [**−m | −s**] [*host*]
rpcinfo −p [*host*]
rpcinfo −T *transport host prognum* [*versnum*]
rpcinfo −l [**−T** *transport*] *host prognum* [*versnum*]
rpcinfo [**−n** *portnum*] **−u** *host prognum* [*versnum*]
rpcinfo [**−n** *portnum*] [**−t**] *host prognum* [*versnum*]
rpcinfo −a *serv_address* **−T** *transport prognum* [*versnum*]
rpcinfo −b [**−T** *transport*] *prognum versnum*
rpcinfo −d [**−T** *transport*] *prognum versnum*

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

In the first synopsis, **rpcinfo** lists all the registered RPC services with **rpcbind** on *host*. If *host* is not specified, the local host is the default. If **−s** is used, the information is displayed in a concise format.

In the second synopsis, **rpcinfo** lists all the RPC services registered with **rpcbind**, version 2. Also note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the **rpcbind** protocol).

The third synopsis makes an RPC call to procedure 0 of *prognum* and *versnum* on the specified *host* and reports whether a response was received. *transport* is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote **rpcbind**.

The *prognum* argument is a number that represents an RPC program number. If a *versnum* is specified, **rpcinfo** attempts to call that version of the specified *prognum*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *prognum* by calling version 0, which is presumed not to exist; if it does exist, **rpcinfo** attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note: the version number is required for **−b** and **−d** options.

OPTIONS

−T *transport*
Specify the transport on which the service is required. If this option is not specified, **rpcinfo** uses the transport specified in the NETPATH environment variable, or if that is unset or null, the transport in the netconfig(5) database is used. This is a generic option, and can be used in conjunction with other options as shown in the SYNOPSIS.

−a *serv_address*
Use *serv_address* as the (universal) address for the service on *transport* to ping procedure 0 of the specified *prognum* and report whether a response was received. The **−T** option is required with the **−a** option.

If *versnum* is not specified, **rpcinfo** tries to ping all available version numbers for that program number. This option avoids calls to remote **rpcbind** to find the address of the service. The *serv_address* is specified in universal address format of the given transport.

−b Make an RPC broadcast to procedure 0 of the specified *prognum* and *versnum* and report all hosts that respond. If *transport* is specified, it broadcasts its request only on the specified transport. If broadcasting is not supported by any transport, an error message is printed. Use of broadcasting should be limited because of the potential for adverse effect on other systems.

−d Delete registration for the RPC service of the specified *prognum* and *versnum*. If *transport* is specified, unregister the service on only that transport, otherwise unregister the service on all the transports on which it was registered. Only the owner of a service can delete a registration, except the super-user who can delete any service.

−l Display a list of entries with a given *prognum* and *versnum* on the specified *host*. Entries are returned for all transports in the same protocol family as that used to contact the remote **rpcbind**.

−m Display a table of statistics of **rpcbind** operations on the given *host*. The table shows statistics for each version of **rpcbind** (versions 2, 3 and 4), giving the number of times each procedure was requested and successfully serviced, the number and type of remote call requests that were made, and information about RPC address lookups that were handled. This is useful for monitoring RPC activities on *host*.

−n *portnum*
Use *portnum* as the port number for the **−t** and **−u** options instead of the port number given by **rpcbind**. Use of this option avoids a call to the remote **rpcbind** to find out the address of the service. This option is made obsolete by the **−a** option.

−p Probe **rpcbind** on *host* using version 2 of the **rpcbind** protocol, and display a list of all registered RPC programs. If *host* is not specified, it defaults to the local host. Note: Version 2 of the **rpcbind** protocol was previously known as the portmapper protocol.

−s Display a concise list of all registered RPC programs on *host*. If *host* is not specified, it defaults to the local host.

−t Make an RPC call to procedure 0 of *prognum* on the specified *host* using TCP, and report whether a response was received. This option is made obsolete by the **−T** option as shown in the third synopsis.

−u Make an RPC call to procedure 0 of *prognum* on the specified *host* using UDP, and report whether a response was received. This option is made obsolete by the **−T** option as shown in the third synopsis.

EXAMPLES

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo
```

To show all of the RPC services registered with **rpcbind** on the machine named **klaxon** use:

```
example% rpcinfo klaxon
```

The information displayed by the above commands can be quite lengthy. Use the **−s** option to display a more concise list:

```
example$ rpcinfo -s klaxon
```

```
program  version(s)                         netid(s)   service     owner
100000   2,3,4      local,tcp,udp,tcp6,udp6 rpcbind    super-user
100008   1          udp,tcp,udp6,tcp6       walld      super-user
100002   2,1        udp,udp6                rusersd    super-user
100001   2,3,4      udp,udp6                rstatd     super-user
100012   1          udp,tcp                 sprayd     super-user
100007   3          udp,tcp                 ypbind     super-user
```

To show whether the RPC service with program number *prognum* and version *versnum* is registered on the machine named **klaxon** for the transport TCP use:

```
example% rpcinfo -T tcp klaxon prognum versnum
```

To show all RPC services registered with version 2 of the **rpcbind** protocol on the local machine use:

```
example% rpcinfo -p
```

To delete the registration for version 1 of the **walld** (program number 100008) service for all transports use:

```
example# rpcinfo -d 100008 1
```

or

```
example# rpcinfo -d walld 1
```

SEE ALSO

rpc(3), netconfig(5), rpc(5), rpcbind(8)

NAME

rshd — remote shell server

SYNOPSIS

rshd [**−aiklnvxPL**] [**−p** *port*]

DESCRIPTION

rshd is the server for the rsh(1) program. It provides an authenticated remote command execution service. Supported options are:

−n, **−-no-keepalive**
Disables keep-alive messages. Keep-alives are packets sent at certain intervals to make sure that the client is still there, even when it doesn't send any data.

−k, **−-kerberos**
Assume that clients connecting to this server will use some form of Kerberos authentication. See the **EXAMPLES** section for a sample inetd.conf(5) configuration.

−x, **−-encrypt**
For Kerberos 4 this means that the connections are encrypted. Kerberos 5 can negotiate encryption even without this option, but if it's present **rshd** will deny unencrypted connections. This option implies **−k**.

−v, **−-vacuous**
If the connecting client does not use any Kerberised authentication, print a message that complains about this fact, and exit. This is helpful if you want to move away from old port-based authentication.

−P When using the AFS filesystem, users' authentication tokens are put in something called a PAG (Process Authentication Group). Multiple processes can share a PAG, but normally each login session has its own PAG. This option disables the **setpag**() call, so all tokens will be put in the default (uid-based) PAG, making it possible to share tokens between sessions. This is only useful in peculiar environments, such as some batch systems.

−i, **−-no-inetd**
The **−i** option will cause **rshd** to create a socket, instead of assuming that its stdin came from inetd(8). This is mostly useful for debugging.

−p *port*, **−-port=***port*
Port to use with **−i**.

−a This flag is for backwards compatibility only.

−L This flag enables logging of connections to syslogd(8). This option is always on in this implementation.

FILES

/etc/hosts.equiv
˜/.rhosts

EXAMPLES

The following can be used to enable Kerberised rsh in inetd.cond(5), while disabling non-Kerberised connections:

```
shell   stream  tcp  nowait  root  /usr/libexec/rshd  rshd -v
kshell  stream  tcp  nowait  root  /usr/libexec/rshd  rshd -k
ekshell stream  tcp  nowait  root  /usr/libexec/rshd  rshd -kx
```

SEE ALSO

`rsh`(1), `iruserok`(3)

HISTORY

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

AUTHORS

This implementation of **rshd** was written as part of the Heimdal Kerberos 5 implementation.

NAME

rshd — remote shell server

SYNOPSIS

rshd [**−aLln**]

DESCRIPTION

The **rshd** server is the server for the rcmd(3) routine and, consequently, for the rsh(1) program. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts.

The **rshd** server listens for service requests at the port indicated in the "cmd" service specification; see services(5). When a service request is received the following protocol is initiated:

1. The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection.
2. The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.
3. If the number received in step 2 is non-zero, it is interpreted as the port number of a secondary stream to be used for the *stderr*. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 512-1023.
4. The server checks the client's source address and requests the corresponding host name (see getnameinfo(3), hosts(5), and named(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the **−a** option is given, the addresses for the hostname are requested, verifying that the name and address correspond. If address verification fails, the connection is aborted with the message "Host address mismatch."
5. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the *client*'s machine.
6. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the **server**'s machine.
7. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
8. **rshd** then validates the user using ruserok(3), which uses the file /etc/hosts.equiv and the .rhosts file found in the user's home directory. The **−l** option prevents ruserok(3) from doing any validation based on the user's ".rhosts" file, unless the user is the superuser.
9. If the file /etc/nologin exists and the user is not the superuser, the connection is closed.
10. A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd**.

Transport-level keepalive messages are enabled unless the **−n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

The **−L** option causes all successful accesses to be logged to syslogd(8) as auth.info messages.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 10 above upon successful completion of all the steps prior to the execution of the login shell).

Locuser too long.
The name of the user on the client's machine is longer than 16 characters.

Ruser too long.
The name of the user on the remote machine is longer than 16 characters.

Command too long.
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
No password file entry for the user name existed.

Remote directory.
The `chdir`(2) to the home directory failed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.
The pipe needed for the *stderr*, wasn't created.

Can't fork; try again.
A `fork`(2) by the server failed.

<shellname>: ...
The user's login shell could not be started. This message is returned on the connection associated with the *stderr*, and is not preceded by a flag byte.

SEE ALSO

`rsh`(1), `ssh`(1), `rcmd`(3), `ruserok`(3), `hosts_access`(5), `login.conf`(5), `sshd`(8)

BUGS

The authentication procedure used here assumes the integrity of every machine and every network that can reach the rshd/rlogind ports on the server. This is insecure, but is useful in an "open" environment. `sshd`(8) or a Kerberized version of this server are much more secure.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol (such as Telnet) should be used.

`rshd` intentionally rejects accesses from IPv4 mapped address on top of `AF_INET6` socket, since IPv4 mapped address complicates host-address based authentication. If you would like to accept connections from IPv4 peers, you will need to run **`rshd`** on top of an `AF_INET` socket, not an `AF_INET6` socket.

NAME

rtadvd — router advertisement daemon

SYNOPSIS

rtadvd [**−DdfMRs**] [**−c** *configfile*] *interface ...*

DESCRIPTION

rtadvd sends router advertisement packets to the specified interfaces.

The program will daemonize itself on invocation. It will then send router advertisement packets periodically, as well as in response to router solicitation messages sent by end hosts.

Router advertisements can be configured on a per-interface basis, as described in `rtadvd.conf`(5).

If there is no configuration file entry for an interface, or if the configuration file does not exist at all, **rtadvd** sets all the parameters to their default values. In particular, **rtadvd** reads all the interface routes from the routing table and advertises them as on-link prefixes.

rtadvd also watches the routing table. If an interface direct route is added on an advertising interface and no static prefixes are specified by the configuration file, **rtadvd** adds the corresponding prefix to its advertising list.

Similarly, when an interface direct route is deleted, **rtadvd** will start advertising the prefixes with zero valid and preferred lifetimes to help the receiving hosts switch to a new prefix when renumbering. Note, however, that the zero valid lifetime cannot invalidate the autoconfigured addresses at a receiving host immediately. According to the specification, the host will retain the address for a certain period, which will typically be two hours. The zero lifetimes rather intend to make the address deprecated, indicating that a new non-deprecated address should be used as the source address of a new connection. This behavior will last for two hours. Then **rtadvd** will completely remove the prefix from the advertising list, and succeeding advertisements will not contain the prefix information.

Moreover, if the status of an advertising interface changes, **rtadvd** will start or stop sending router advertisements according to the latest status.

The **−s** option may be used to disable this behavior; **rtadvd** will not watch the routing table and the whole functionality described above will be suppressed.

Basically, hosts MUST NOT send Router Advertisement messages at any time (RFC 2461, Section 6.2.3). However, it would sometimes be useful to allow hosts to advertise some parameters such as prefix information and link MTU. Thus, **rtadvd** can be invoked if router lifetime is explicitly set to zero on every advertising interface.

The command line options are:

−c *configfile*
: Specify an alternate location, *configfile*, for the configuration file. By default, `/etc/rtadvd.conf` is used.

−D
: Even more debugging information than that offered by the **−d** option is printed.

−d
: Print debugging information.

−f
: Foreground mode (useful when debugging). Log messages will be dumped to stderr when this option is specified.

−M
: Specify an interface to join the all-routers site-local multicast group. By default, **rtadvd** tries to join the first advertising interface appearing on the command line. This option has meaning only with the **−R** option, which enables routing renumbering protocol support.

-R Accept router renumbering requests. If you enable it, an `ipsec`(4) setup is suggested for security reasons. This option is currently disabled, and is ignored by **rtadvd** with a warning message.

-s Do not add or delete prefixes dynamically. Only statically configured prefixes, if any, will be advertised.

Upon receipt of signal `SIGUSR1`, **rtadvd** will dump the current internal state into `/var/run/rtadvd.dump`.

Use `SIGTERM` to kill **rtadvd** gracefully. In this case, **rtadvd** will transmit router advertisement with router lifetime 0 to all the interfaces (in accordance with RFC 2461 6.2.5) .

EXIT STATUS

The **rtadvd** utility exits 0 on success, and >0 if an error occurs.

FILES

`/etc/rtadvd.conf`	The default configuration file.
`/var/run/rtadvd.pid`	Contains the PID of the currently running **rtadvd**.
`/var/run/rtadvd.dump`	The file in which **rtadvd** dumps its internal state.

SEE ALSO

`rtadvd.conf`(5), `rtsol`(8)

HISTORY

The **rtadvd** command first appeared in the WIDE Hydrangea IPv6 protocol stack kit.

BUGS

There used to be some text that recommended users not to let **rtadvd** advertise Router Advertisement messages on an upstream link to avoid undesirable `icmp6`(4) redirect messages. However, based on later discussion in the IETF IPng working group, all routers should rather advertise the messages regardless of the network topology, in order to ensure reachability.

NAME

rtcalarm — Display or set x68k's RTC alarm timer

SYNOPSIS

rtcalarm [**-w** *day-of-the-week*] [**-d** *day-of-the-month*] [[**-m** *minites*] |
[**-s** *seconds*]] [**-c -channel**] *hh:mm*
rtcalarm off
rtcalarm

DESCRIPTION

rtcalarm displays or sets x68k's RTC alarm timer.

The first form of **rtcalarm** sets the alarm timer to specified time and enables it. *Day-of-the-week* is a number from 0 to 6, which means Sunday, Monday,.. etc.

If **-c** is omitted, the alarm timer starts the x68k in computer mode.

The second form disables the alarm timer.

If no arguments are supplied (the third form), **rtcalarm** displays the current settings of the alarm timer.

Note that setting the alarm timer is allowed only for the super user.

SEE ALSO

pow(4)

AUTHORS

rtcalarm was written by MINOURA Makoto <minoura@flab.fujitsu.co.jp>.

NAME

rtquery — query routing daemons for their routing tables

SYNOPSIS

rtquery [**−np1**] [**−w** *timeout*] [**−r** *addr*] [**−a** *secret*] *host ...*
rtquery [**−t** *op*] *host ...*

DESCRIPTION

rtquery is used to query a RIP network routing daemon, routed(8) or gated(8), for its routing table by sending a *request* or *poll* command. The routing information in any routing *response* packets returned is displayed numerically and symbolically.

rtquery by default uses the *request* command. When the **−p** option is specified, **rtquery** uses the *poll* command, an undocumented extension to the RIP protocol supported by gated(8). When querying gated(8), the *poll* command is preferred over the *request* command because the response is not subject to Split Horizon and/or Poisoned Reverse, and because some versions of gated do not answer the *request* command. routed(8) does not answer the *poll* command, but recognizes *requests* coming from **rtquery** and so answers completely.

rtquery is also used to turn tracing on or off in routed(8).

The following options are available:

−n displays only the numeric network and host numbers instead of both numeric and symbolic.

−p uses the *poll* command to request full routing information from gated(8). This is an undocumented extension RIP protocol supported only by gated(8).

−1 queries using RIP version 1 instead of RIP version 2.

−w *timeout*
changes the delay for an answer from each host. By default, each host is given 15 seconds to respond.

−r *addr*
asks about the route to destination *addr*.

−a *passwd=XXX*

−a *md5_passwd=XXX|KeyID*
causes the query to be sent with the indicated cleartext or MD5 password.

−t *op* changes tracing, where *op* is one of the following. Requests from processes not running with UID 0 or on distant networks are generally ignored by the daemon except for a message in the system log. gated(8) is likely to ignore these debugging requests.

on=tracefile
turns tracing on into the specified file. That file must usually have been specified when the daemon was started or be the same as a fixed name, often `/etc/routed.trace`.

more increases the debugging level.

off turns off tracing.

dump dumps the daemon's routing table to the current tracefile.

SEE ALSO

`gated`(8), `routed`(8)
RFC 1058 - Routing Information Protocol, RIPv1
RFC 1723 - Routing Information Protocol, RIPv2

NAME

rtsold — router solicitation daemon

SYNOPSIS

rtsold [**−1Ddfm**] *interface ...*
rtsold [**−1Ddfm**] **−a**
rtsol [**−Dd**] *interface ...*
rtsol [**−Dd**] **−a**

DESCRIPTION

rtsold is the daemon program to send ICMPv6 Router Solicitation messages on the specified interfaces. If a node (re)attaches to a link, **rtsold** sends some Router Solicitations on the link destined to the link-local scope all-routers multicast address to discover new routers and to get non link-local addresses.

rtsold should be used on IPv6 hosts (non-router nodes) only.

If you invoke the program as **rtsol**, it will transmit probes from the specified *interface*, without becoming a daemon. In other words, **rtsol** behaves as "rtsold -f1 interface ...".

Specifically, **rtsold** sends at most 3 Router Solicitations on an interface after one of the following events:

- Just after invocation of **rtsold** daemon.
- The interface is up after a temporary interface failure. **rtsold** detects such failures by periodically probing to see if the status of the interface is active or not. Note that some network cards and drivers do not allow the extraction of link state. In such cases, **rtsold** cannot detect the change of the interface status.
- Every 60 seconds if the **−m** option is specified and the **rtsold** daemon cannot get the interface status. This feature does not conform to the IPv6 neighbor discovery specification, but is provided for mobile stations. The default interval for router advertisements, which is on the order of 10 minutes, is slightly long for mobile stations. This feature is provided for such stations so that they can find new routers as soon as possible when they attach to another link.

Once **rtsold** has sent a Router Solicitation, and has received a valid Router Advertisement, it refrains from sending additional solicitations on that interface, until the next time one of the above events occurs.

When sending a Router Solicitation on an interface, **rtsold** includes a Source Link-layer address option if the interface has a link-layer address.

Upon receipt of signal SIGUSR1, **rtsold** will dump the current internal state into /var/run/rtsold.dump. Also note that **rtsold** will not be able to update the kernel routing tables unless sysctl(8) reports that net.inet6.ip6.accept_rtadv=1.

The options are as follows:

−1 Perform only one probe. Transmit Router Solicitation packets until at least one valid Router Advertisement packet has arrived on each *interface*, then exit.

−a Autoprobe outgoing interface. **rtsold** will try to find a non-loopback, non-point-to-point, IPv6-capable interface. If **rtsold** finds multiple interfaces, **rtsold** will exit with error.

−D Enable more debugging (than that offered by the **−d** option) including the printing of internal timer information.

−d Enable debugging.

-f This option prevents **rtsold** from becoming a daemon (foreground mode). Warning messages are generated to standard error instead of `syslog`(3).

-m Enable mobility support. If this option is specified, **rtsold** sends probing packets to default routers that have advertised Router Advertisements when the node (re)attaches to an interface. Moreover, if the option is specified, **rtsold** periodically sends Router Solicitation on an interface that does not support `SIOCGIFMEDIA` ioctl.

EXIT STATUS

The **rtsold** utility exits 0 on success, and >0 if an error occurs.

FILES

`/var/run/rtsold.pid`	The PID of the currently running **rtsold**.
`/var/run/rtsold.dump`	Internal state dump file.

SEE ALSO

`rtadvd`(8), `sysctl`(8)

HISTORY

The **rtsold** command is based on the **rtsol** command, which first appeared in WIDE/KAME IPv6 protocol stack kit. **rtsol** is now integrated into **rtsold**.

BUGS

In some operating systems, when a PCMCIA network card is removed and reinserted, the corresponding interface index is changed. However, **rtsold** assumes such changes will not occur, and always uses the index that it got at invocation. As a result, **rtsold** may not work if you reinsert a network card. In such a case, **rtsold** should be killed and restarted.

You may see kernel error messages if you try to autoconfigure a host with multiple interfaces.

NAME

rwhod — system status server

SYNOPSIS

rwhod [**-i** *interval*] [**-u** *user*]

DESCRIPTION

rwhod is the server which maintains the database used by the rwho(1) and ruptime(1) programs. Its operation is predicated on the ability to *broadcast* messages on a network.

The following options are available:

-i *interval*
: Allows for the broadcast interval to be changed from the default 3 minutes. The *interval* argument is the number of seconds to change the interval to, or if the value is suffixed by "m" then it is interpreted as minutes. The maximum allowed value for the broadcast interval is 11 minutes because higher values will cause ruptime(1) to mark the host as being down.

-u *user*
: Drop privileges and become the user *user*.

rwhod operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other **rwhod** servers' status messages, validating them, then recording them in a collection of files located in the directory /var/rwho.

The server transmits and receives messages at the port indicated in the "who" service specification; see services(5). The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];          /* tty name */
        char    out_name[8];          /* user id */
        int32_t out_time;             /* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int32_t wd_sendtime;
        int32_t wd_recvtime;
        char    wd_hostname[32];
        int32_t wd_loadav[3];
        int32_t wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int32_t we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the w(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission; they are multiplied by 100 for representation in an integer. The host name included is that returned by the gethostname(3) function call, with any trailing domain name omitted. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the `utmp`(5) entry for each non-idle terminal line and a value indicating the time in seconds since a character was last received on the terminal line.

Messages received by the `rwho`(1) server are discarded unless they originated at an `rwho`(1) server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named `whod.hostname` in the directory `/var/rwho`. These files contain only the most recent message, in the format described above.

Status messages are generated by default approximately once every 3 minutes.

SEE ALSO

`ruptime`(1), `rwho`(1)

HISTORY

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

BUGS

There should be a way to relay status information between networks. Status information should be sent only upon request rather than continuously. People often interpret the server dying or network communication failures as a machine going down.

NAME

sa — print system accounting statistics

SYNOPSIS

sa [**−abcdDfijkKlmnqrstu**] [**−v** *cutoff*] [*file ...*]

DESCRIPTION

The **sa** utility reports on, cleans up, and generally maintains system accounting files.

sa is able to condense the information in `/var/account/acct` into the summary files `/var/account/savacct` and `/var/account/usracct`, which contain system statistics according to command name and login id, respectively. This condensation is desirable because on a large system, `/var/account/acct` can grow by hundreds of blocks per day. The summary files are normally read before the accounting file, so that reports include all available information.

If file names are supplied, they are read instead of `/var/account/acct`. After each file is read, if the summary files are being updated, an updated summary will be saved to disk. Only one report is printed, after the last file is processed.

The labels used in the output indicate the following, except where otherwise specified by individual options:

`avio`	Average number of I/O operations per execution
`cp`	Sum of user and system time, in minutes
`cpu`	Same as `cp`
`k`	CPU-time averaged core usage, in 1k units
`k*sec`	CPU storage integral, in 1k-core seconds
`re`	Real time, in minutes
`s`	System time, in minutes
`tio`	Total number of I/O operations
`u`	User time, in minutes

The options to **sa** are:

−a List all command names, including those containing unprintable characters and those used only once. By default, **sa** places all names containing unprintable characters and those used only once under the name "***other".

−b If printing command statistics, sort output by the sum of user and system time divided by number of calls.

−c In addition to the number of calls and the user, system and real times for each command, print their percentage of the total over all commands.

−d If printing command statistics, sort by the average number of disk I/O operations. If printing user statistics, print the average number of disk I/O operations per user.

−D If printing command statistics, sort and print by the total number of disk I/O operations.

−f Force no interactive threshold comparison with the **−v** option.

−i Do not read in the summary files.

-j Instead of the total minutes per category, give seconds per call.

-k If printing command statistics, sort by the CPU-time average memory usage. If printing user statistics, print the CPU-time average memory usage.

-K If printing command statistics, print and sort by the CPU-storage integral.

-l Separate system and user time; normally they are combined.

-m Print per-user statistics rather than per-command statistics.

-n Sort by number of calls.

-q Create no output other than error messages.

-r Reverse order of sort.

-s Truncate the accounting files when done and merge their data into the summary files.

-t For each command, report the ratio of real time to the sum of user and system CPU times. If the CPU time is too small to report, ‘‘*ignore*’’ appears in this field.

-u Superseding all other flags, for each entry in the accounting file, print the user ID, total seconds of CPU usage, total memory usage, number of I/O operations performed, and command name.

-v *cutoff*
For each command used *cutoff* times or fewer, print the command name and await a reply from the terminal. If the reply begins with ‘‘y’’, add the command to the category ‘‘**junk**’’. This flag is used to strip garbage from the report.

By default, per-command statistics will be printed. The number of calls, the total elapsed time in minutes, total CPU and user time in minutes, average number of I/O operations, and CPU-time averaged core usage will be printed. If the **-m** option is specified, per-user statistics will be printed, including the user name, the number of commands invoked, total CPU time used (in minutes), total number of I/O operations, and CPU storage integral for each user. If the **-u** option is specified, the uid, user and system time (in seconds), CPU storage integral, I/O usage, and command name will be printed for each entry in the accounting data file.

If the **-u** flag is specified, all flags other than **-q** are ignored. If the **-m** flag is specified, only the **-b**, **-d**, **-i**, **-k**, **-q**, and **-s** flags are honored.

The **sa** utility exits 0 on success, and >0 if an error occurs.

FILES

`/var/account/acct`	raw accounting data file
`/var/account/savacct`	per-command accounting summary database
`/var/account/usracct`	per-user accounting summary database

SEE ALSO

lastcomm(1), acct(5), ac(8), accton(8)

HISTORY

sa was written for NetBSD 1.0 from the specification provided by various systems' manual pages. Its date of origin is unknown to the author.

AUTHORS

Chris G. Demetriou ⟨cgd@postgres.berkeley.edu⟩.

BUGS

The number of options to this program is absurd, especially considering that there's not much logic behind their lettering.

The field labels should be more consistent.

NetBSD's VM system does not record the CPU storage integral.

CAVEATS

While the behavior of the options in this version of **sa** was modeled after the original version, there are some intentional differences and undoubtedly some unintentional ones as well. In particular, the **−q** option has been added, and the **−m** option now understands more options than it used to.

The formats of the summary files created by this version of **sa** are very different than the those used by the original version. This is not considered a problem, however, because the accounting record format has changed as well (since user ids are now 32 bits).

NAME

savecore — save a core dump of the operating system

SYNOPSIS

savecore [**−fvz**] [**−N** *system*] [**−Z** *level*] *directory*
savecore −c [**−v**] [**−N** *system*]
savecore −n [**−v**] [**−N** *system*]

DESCRIPTION

When the NetBSD kernel encounters a fatal error, the `panic`(9) routine arranges for a snapshot of the contents of physical memory to be written into a dump area, typically in the swap partition.

Upon a subsequent reboot, **savecore** is typically run out of `rc`(8), before swapping is enabled, to copy the kernel and the saved memory image into *directory*, and enters a reboot message and information about the core dump into the system log.

The kernel and core file can then be analyzed using various tools, including `dmesg`(8), `fstat`(1), `gdb`(1), `iostat`(8), `netstat`(1), `ps`(1), and `pstat`(8), to attempt to deduce the cause of the crash.

Crashes are usually the result of kernel bugs; if this is suspected, a full bug report should be filed using `send-pr`(1), containing as much information as possible about the circumstances of the crash. Since crash dumps are typically very large and may contain whatever (potentially confidential) information was in memory at the time of the crash, do *NOT* include a copy of the crash dump file in the bug report; instead, save it somewhere in the event that a NetBSD developer wants to examine it.

The options are as follows:

−c Only clears the dump without saving it, so that future invocations of **savecore** will ignore it.

−f Forces a dump to be taken even if the dump doesn't appear correct or there is insufficient disk space.

−n Check whether a dump is present without taking further action. The command exits with zero status if a dump is present, or with non-zero status otherwise.

−N Use *system* as the kernel instead of the default (returned by `getbootfile`(3)). Note that `getbootfile`(3) uses `secure_path`(3) to check that kernel file is "secure" and will default to `/netbsd` if the check fails.

−v Prints out some additional debugging information.

−z Compresses the core dump and kernel (see `gzip`(1)).

−Z *level* Set the compression level for **−z** to *level*. Defaults to 1 (the fastest compression mode). Refer to `gzip`(1) for more information regarding the compression level.

savecore checks the core dump in various ways to make sure that it is current and that it corresponds to the currently running system. If it passes these checks, it saves the core image in *directory*/`netbsd.#.core` and the system in *directory*/`netbsd.#` (or in *directory*/`netbsd.#.core.gz` and *directory*/`netbsd.#.gz`, respectively, if the **−z** option is used). The "#" is the number from the first line of the file *directory*/`bounds`, and it is incremented and stored back into the file each time **savecore** successfully runs.

savecore also checks the available disk space before attempting to make the copies. If there is insufficient disk space in the file system containing *directory*, or if the file *directory*/`minfree` exists and the number of free kilobytes (for non-superusers) in the file system after the copies were made would be less than the number in the first line of this file, the copies are not attempted.

If **savecore** successfully copies the kernel and the core dump, the core dump is cleared so that future invocations of **savecore** will ignore it.

SEE ALSO

fstat(1), gdb(1), gzip(1), netstat(1), ps(1), send-pr(1), secure_path(3), dmesg(8), iostat(8), pstat(8), rc(8), syslogd(8), panic(9)

HISTORY

The **savecore** command appeared in 4.1BSD.

BUGS

The minfree code does not consider the effect of compression.

NAME

scache − Postfix shared connection cache server

SYNOPSIS

scache [generic Postfix daemon options]

DESCRIPTION

The **scache**(8) server maintains a shared multi-connection cache. This information can be used by, for example, Postfix SMTP clients or other Postfix delivery agents.

The connection cache is organized into logical destination names, physical endpoint names, and connections.

As a specific example, logical SMTP destinations specify (transport, domain, port), and physical SMTP endpoints specify (transport, IP address, port). An SMTP connection may be saved after a successful mail transaction.

In the general case, one logical destination may refer to zero or more physical endpoints, one physical endpoint may be referenced by zero or more logical destinations, and one endpoint may refer to zero or more connections.

The exact syntax of a logical destination or endpoint name is application dependent; the **scache**(8) server does not care. A connection is stored as a file descriptor together with application-dependent information that is needed to re-activate a connection object. Again, the **scache**(8) server is completely unaware of the details of that information.

All information is stored with a finite time to live (ttl). The connection cache daemon terminates when no client is connected for **max_idle** time units.

This server implements the following requests:

save_endp *ttl endpoint endpoint_properties file_descriptor*
: Save the specified file descriptor and connection property data under the specified endpoint name. The endpoint properties are used by the client to re-activate a passivated connection object.

find_endp *endpoint*
: Look up cached properties and a cached file descriptor for the specified endpoint.

save_dest *ttl destination destination_properties endpoint*
: Save the binding between a logical destination and an endpoint under the destination name, together with destination specific connection properties. The destination properties are used by the client to re-activate a passivated connection object.

find_dest *destination*
: Look up cached destination properties, cached endpoint properties, and a cached file descriptor for the specified logical destination.

SECURITY

The **scache**(8) server is not security-sensitive. It does not talk to the network, and it does not talk to local users. The **scache**(8) server can run chrooted at fixed low privilege.

The **scache**(8) server is not a trusted process. It must not be used to store information that is security sensitive.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

The session cache cannot be shared among multiple machines.

When a connection expires from the cache, it is closed without the appropriate protocol specific handshake.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **scache**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

RESOURCE CONTROLS

connection_cache_ttl_limit (2s)
The maximal time-to-live value that the **scache**(8) connection cache server allows.

connection_cache_status_update_time (600s)
How frequently the **scache**(8) server logs usage statistics with connection cache hit and miss rates for logical destinations and for physical endpoints.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

smtp(8), SMTP client
postconf(5), configuration parameters
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
CONNECTION_CACHE_README, Postfix connection cache

LICENSE

The Secure Mailer license must be distributed with this software.

HISTORY

This service was introduced with Postfix version 2.2.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704

Yorktown Heights, NY 10598, USA

NAME

scan_ffs, **scan_lfs** — find FFSv1/FFSv2/LFS partitions on a disk or file

SYNOPSIS

scan_ffs [**-blv**] [**-e** *end*] [**-F** *file*] [**-s** *start*] *device*

DESCRIPTION

scan_ffs will take a raw disk device that covers the whole disk or a file and will find all possible FFSv[12]/LFS partitions, independent of block sizes on it. It will show the file system type (FFSv1, FFSv2, or LFS), size, and offset. Also it has an option to show the values with a disklabel-alike output.

The options are as follows:

-b	Report every superblock found with its sector address, rather than trying to report the partition boundaries. This option can be useful to find the other superblocks in a partition if the first superblock has become corrupted. It is most useful if *device* refers to the raw device for the partition, rather than the entire disk.
-e *end*	Where to stop searching for file systems. The *end* argument specifies the last sector that will be searched. Default is the last sector of *device*.
-F *file*	Path to a file containing possible partitions inside of it.
-l	Print out a string looking much like the input to disklabel. With a little massaging, this output can usually be used by disklabel(8).
-s *start*	Where to start searching for file systems. This makes it easier to skip swap partitions or other large non-UFS/FFS partitions. The *start* argument specifies the first sector that will be searched. Default is the first sector of *device*.
-v	Be verbose about what **scan_ffs** is doing, and what has been found.

The *device* argument specifies which device **scan_ffs** should scan for file systems.

scan_lfs is just another name for the same program, both behave in exactly the same way.

SEE ALSO

disklabel(8)

HISTORY

The **scan_ffs** program first appeared in OpenBSD 2.3 and then in NetBSD 3.1. Support for searching in files was added in NetBSD 4.0.

AUTHORS

scan_ffs was written for OpenBSD by Niklas Hallqvist and Tobias Weingartner. It was ported to NetBSD by Juan Romero Pardines, who added support for LFS/FFSv2, partitions with fragsize/blocksize greater than 2048/16384 for FFSv1, searching on files, etc.

BUGS

Currently **scan_ffs** won't find partitions with fragsize/blocksize greater than 8192/65536.

NAME

schedctl — control scheduling of processes and threads

SYNOPSIS

schedctl [**-A** *cpus*] [**-C** *class*] [**-P** *pri*] [**-t** *lid*] **-p** *pid* | *command*

DESCRIPTION

The **schedctl** command can be used to control the scheduling of processes and threads. It also returns information about the current scheduling parameters of the process or thread. Only the super-user may change the scheduling parameters. **schedctl** can also be used to start a new command using the specified parameters.

Available options:

-A *cpus*
Set of the processors on which process or thread should run, that is, affinity. Processors are defined as numbers (starting from zero) and separated by commas. A value of −1 is used to unset the affinity.

-C *class*
Scheduling class (policy), one of:

SCHED_OTHER	Time-sharing (TS) scheduling policy. The default policy in NetBSD.
SCHED_FIFO	First in, first out (FIFO) scheduling policy.
SCHED_RR	Round-robin scheduling policy.

-P *pri*
Priority which will be set for the process or thread. For all scheduling classes, priority might be set to any value in the range from SCHED_PRI_MIN (0) to SCHED_PRI_MAX (63).

-p *pid*
The target process which will be affected. If the process has more than one thread, all of them will be affected.

If **-p** is not given, a command to execute must be given on the command line.

-t *lid*
Thread in the specified process. If specified, only this thread in the process will be affected. May only be specified if **-p** is also given.

EXAMPLES

Show scheduling information about the process whose ID is "123":

```
# schedctl -p 123
```

Set the affinity to CPU 0 and CPU 1, policy to SCHED_RR, and priority to 63 for thread whose ID is "1" in process whose ID is "123":

```
# schedctl -p 123 -t 1 -A 0,1 -C 2 -P 63
```

Run the top(1) command with real-time priority:

```
# schedctl -C SCHED_FIFO top
```

SEE ALSO

nice(1), getpriority(2), setpriority(2), renice(8)

HISTORY

The **schedctl** command first appeared in NetBSD 5.0.

NAME

scsictl — a program to manipulate SCSI devices and busses

SYNOPSIS

scsictl *device command* [*arg* [...]]

DESCRIPTION

scsictl allows a user or system administrator to issue commands to and otherwise control SCSI devices and busses. It is used by specifying a device or bus to manipulate, the command to perform, and any arguments the command may require. **scsictl** determines if the specified device is an actual device or a SCSI bus automatically, and selects the appropriate command set.

For commands which **scsictl** issues a SCSI command to the device directly, any returned sense information will be decoded by **scsictl** and displayed to the standard output.

DEVICE COMMANDS

The following commands are supported for SCSI devices:

defects [primary] [grown] [block|byte|physical]

Read the primary and/or grown defect lists from the specified device in block, byte from index, or physical sector format. The default is to return both the primary and grown defect lists in physical sector format. This command is only supported on direct access devices.

format [blocksize [immediate]]

(Low level) format the named device. If the optional `blocksize` parameter is provided, the device geometry will be modified to use the specified `blocksize`. If this parameter is different form the Current or Default Mode Page 3 parameters, the device will update Mode Page 3 at the successful completion of the Format. Device geometry may change as a result of using a new device `blocksize`. When the optional `blocksize` parameter is specified, the Defect List on the drive will revert to the original primary defect list created at the time of manufacture if available. The drive will usually recertify itself during the Format and add any other defective blocks to the new Defect List. Some disks may not support the ability to change the blocksize and may enter a Degraded Mode when fed a Format command of this type. If this happens the standard recovery for the drive requires issuing a correct Format command, i.e. one without the blocksize parameter.

When the `immediate` parameter is also specified, the disk is instructed to return from the format command right away. It continues to format, and every ten seconds **scsictl** issues a TEST UNIT READY command to check the associated sense data. This associated sense data has a progress indicator which indicates how far the format is progressing. Note well that most SCSI disk drives prior to a few years ago do not support this option.

identify

Identify the specified device, displaying the device's SCSI bus, target, and lun, as well as the device's vendor, product, and revision strings.

reassign *blkno* [blkno [...]]

Issues a `REASSIGN BLOCKS` command to the device, adding the specified blocks to the grown defect list. This command is only supported on direct access devices.

release

Send a "RELEASE" command to the device to release a reservation on it.

reserve

Send a "RESERVE" command to the device to place a reservation on it.

reset

Reset the device. This command is only supported for devices which support the `SCIOCRESET` ioctl.

start

Send a "START" command to the device. This is useful typically only for disk devices.

stop

Send a "STOP" command to the device. This is useful typically only for disk devices.

tur

Send a "TEST UNIT READY" command to the device. This is useful for generating current device status.

getcache

Returns basic cache parameters for the device.

setcache *none*|*r*|*w*|*rw* [*save*]

Set basic cache parameters for the device. The cache may be disabled (none), the read cache enabled (r), the write cache enabled (w), or both read and write cache enabled (rw). If the drive's cache parameters are savable, specifying *save* after the cache enable state will cause the parameters to be saved in non-volatile storage.

flushcache

Explicitly flushes the write cache.

setspeed *speed*

Set the highest speed that the optical drive should use for reading data. The units are multiples of a single speed CDROM (150 KB/s). Specify 0 to use the drive's fastest speed.

BUS COMMANDS

The following commands are supported for SCSI busses:

reset

Reset the SCSI bus. This command is only supported if the host adapter supports the `SCBUSIORESET` ioctl.

scan *target lun*

Scan the SCSI bus for devices. This is useful if a device was not connected or powered on when the system was booted. The *target* and *lun* arguments specify which SCSI target and lun on the bus is to be scanned. Either may be wildcarded by specifying the keyword "any" or "all".

detach *target lun*

Detach the specified device from the bus. Useful if a device is powered down after use. The *target* and *lun* arguments have the same meaning as for the **scan** command, and may also be wildcarded.

NOTES

When scanning the SCSI bus, information about newly recognized devices is printed to console. No information is printed for already probed devices.

FILES

`/dev/scsibus*` - for commands operating on SCSI busses

SEE ALSO

`ioctl`(2), `cd`(4), `ch`(4), `sd`(4), `se`(4), `ss`(4), `st`(4), `uk`(4), `atactl`(8), `dkctl`(8)

HISTORY

The **scsictl** command first appeared in NetBSD 1.4.

AUTHORS

The **scsictl** command was written by Jason R. Thorpe of the Numerical Aerospace Simulation Facility, NASA Ames Research Center.

NAME

sdpd — Bluetooth Service Discovery Protocol daemon

SYNOPSIS

sdpd [**−dh**] [**−c** *path*] [**−G** *group*] [**−g** *group*] [**−u** *user*]

DESCRIPTION

The **sdpd** daemon keeps track of the Bluetooth services registered on the host and responds to Service Discovery inquiries from the remote Bluetooth devices.

In order to use any service remote Bluetooth device need to send Service Search and Service Attribute or Service Search Attribute request over Bluetooth L2CAP connection on SDP PSM (0x0001). The **sdpd** daemon will try to find matching Service Record in its Service Database and will send appropriate response back. The remote device then will process the response, extract all required information and will make a separate connection in order to use the service.

Bluetooth applications, running on the host, register services with the local **sdpd** daemon. Operation like service registration, service removal and service change are performed over the control socket. It is possible to query entire content of the **sdpd** Service Database with `sdpquery`(1) by issuing **browse** command on the control socket.

The command line options are as follows:

−c *path*
Specify path to the control socket. The default path is `/var/run/sdp`.

−d Do not detach from the controlling terminal.

−G *group*
Grant permission to members of the *group* to modify the **sdpd** Service Database.

−g *group*
Specifies the group the **sdpd** should run as after it initializes. The value specified may be either a group name or a numeric group ID. This only works if **sdpd** was started as root. The default group name is “`_sdpd`”.

−h Display usage message and exit.

−u *user*
Specifies the user the **sdpd** should run as after it initializes. The value specified may be either a user name or a numeric user ID. This only works if **sdpd** was started as root. The default user name is “`_sdpd`”.

CAVEAT

The **sdpd** daemon will listen for incoming L2CAP connections on a wildcard BD_ADDR.

In case of multiple Bluetooth devices connected to the same host it is possible to specify which services should be “bound” to which Bluetooth device. Such assignment should be done at service registration time.

Requests to register, remove or change service can only be made via the control socket. The **sdpd** daemon will check the peer's credentials and will only accept the request when the peer is the superuser, of if the peer is a member of the group specified with the **−G** option.

The **sdpd** daemon does not check for duplicated Service Records. It only performs minimal checking on the service data sent in the Service Register request. It is assumed that application must obtain all required resources such as RFCOMM channels etc., before registering the service.

FILES

`/var/run/sdp`

SEE ALSO

`sdpquery`(1), `sdp`(3)

HISTORY

The **`sdpd`** daemon first appeared in FreeBSD 5.3 and was imported into NetBSD 4.0 by Iain Hibbert under the sponsorship of Itronix, Inc.

AUTHORS

Maksim Yevmenkin ⟨m_evmenkin@yahoo.com⟩

BUGS

Most likely. Please report if found.

NAME

security — NetBSD security features

DESCRIPTION

NetBSD supports a variety of security features. Below is a brief description of them with some quick usage examples that will help you get started.

Contents:
- Veriexec (file integrity)
- Exploit mitigation
- Per-user `/tmp` directory
- Information filtering

VERIEXEC

Veriexec is a file integrity subsystem.

For more information about it, and a quick guide on how to use it, please see `veriexec`(8).

In a nutshell, once enabled, *Veriexec* can be started as follows:

```
# veriexecgen && veriexec load
```

EXPLOIT MITIGATION

NetBSD incorporates some exploit mitigation features. The purpose of exploit mitigation features is to interfere with the way exploits work, in order to prevent them from succeeding. Due to that, some features may have other impact on the system, so be sure to fully understand the implications of each feature.

NetBSD provides the following exploit mitigation features:
- PaX ASLR (Address Space Layout Randomization)
- PaX MPROTECT (`mprotect`(2) restrictions)
- PaX SegvGuard
- `gcc`(1) stack-smashing protection (SSP)

PaX ASLR

PaX ASLR implements Address Space Layout Randomization, meant to complement non-executable mappings. Its purpose is to harden prediction of the address space layout, namely location of library and application functions that can be used by an attacker to circumvent non-executable mappings by using a technique called "return to library" to bypass the need to write new code to (potentially executable) regions of memory.

When *PaX ASLR* is used, it is more likely the attacker will fail to predict the addresses of such functions, causing the application to segfault. To detect cases where an attacker might try and brute-force the return address of respawning services, *PaX Segvguard* can be used (see below).

For non-PIE (Position Independent Executable) executables, the NetBSD *PaX ASLR* implementation introduces randomization to the following memory regions:
1. The data segment
2. The stack

For PIE executables:
1. The program itself (exec base)
2. All shared libraries
3. The data segment
4. The stack

While it can be enabled globally, NetBSD provides a tool, `paxctl`(8), to enable *PaX ASLR* on a per-program basis.

Example usage:

```
# paxctl +A /usr/sbin/sshd
```

Enabling *PaX ASLR* globally:

```
# sysctl -w security.pax.aslr.global=1
```

PaX MPROTECT

PaX MPROTECT implements memory protection restrictions, meant to complement non-executable mappings. Their purpose is to prevent situations where malicious code attempts to mark writable memory regions as executable, often by trashing arguments to an `mprotect`(2) call.

While it can be enabled globally, NetBSD provides a tool, `paxctl`(8), to enable *PaX MPROTECT* on a per-program basis.

Example usage:

```
# paxctl +M /usr/sbin/sshd
```

Enabling *PaX MPROTECT* globally:

```
# sysctl -w security.pax.mprotect.global=1
```

PaX Segvguard

PaX Segvguard monitors the number of segmentation faults in a program on a per-user basis, in an attempt to detect on-going exploitation attempts and possibly prevent them. For instance, *PaX Segvguard* can help detect when an attacker tries to brute-force a function return address, when attempting to perform a return-to-lib attack.

PaX Segvguard consumes kernel memory, so use it wisely. While it provides rate-limiting protections, records are tracked for all users on a per-program basis, meaning that irresponsible use may result in tracking all segmentation faults in the system, possibly consuming all kernel memory.

For this reason, it is highly recommended to have *PaX Segvguard* enabled explicitly only for network services, etc. Enabling *PaX Segvguard* explicitly works like this:

```
# paxctl +G /usr/sbin/sshd
```

However, a global knob is still provided, for use in strict environments with no local users (some network appliances, embedded devices, firewalls, etc.):

```
# sysctl -w security.pax.segvguard.global=1
```

Explicitly disabling *PaX Segvguard* is also possible:

```
# paxctl +g /bin/ls
```

In addition, *PaX Segvguard* provides several tunable options. For example, to limit a program to 5 segmentation faults from the same user in a 60 second timeframe:

```
# sysctl -w security.pax.segvguard.max_crashes=5
# sysctl -w security.pax.segvguard.expiry_timeout=60
```

The number of seconds a user will be suspended from running the culprit program is also configurable. For example, 10 minutes seem like a sane setting:

```
# sysctl -w security.pax.segvguard.suspend_timeout=600
```

GCC Stack Smashing Protection (SSP)

As of NetBSD 4.0, `gcc`(1) includes *SSP*, a set of compiler extensions to raise the bar on exploitation attempts by detecting corruption of variables and buffer overruns, which may be used to affect program control flow.

Upon detection of a buffer overrun, *SSP* will immediately abort execution of the program and send a log message to `syslog`(3).

The system (userland and kernel) can be built with *SSP* by using the "USE_SSP" flag in `/etc/mk.conf`:

```
USE_SSP=yes
```

You are encouraged to use *SSP* for software you build, by providing one of the **`−fstack-protector`** or **`−fstack-protector-all`** flags to `gcc`(1). Keep in mind, however, that *SSP* will not work for functions that make use of `alloca`(3), as the latter modifies the stack size during run-time, while *SSP* relies on it being a compile-time static.

Use of *SSP* is especially encouraged on platforms without per-page execute bit granularity such as *i386*.

PER-USER TEMPORARY STORAGE

It is possible to configure per-user temporary storage to avoid potential security issues (race conditions, etc.) in programs that do not make secure usage of `/tmp`.

To enable per-user temporary storage, add the following line to `rc.conf`(5):

```
per_user_tmp=YES
```

If `/tmp` is a mount point, you will also need to update its `fstab`(5) entry to use "/private/tmp" (or whatever directory you want, if you override the default using the "per_user_tmp_dir" `rc.conf`(5) keyword) instead of "/tmp".

Following that, run:

```
# /etc/rc.d/perusertmp start
```

INFORMATION FILTERING

NetBSD provides administrators the ability to restrict information passed from the kernel to userland so that users can only view information they "own".

The hooks that manage this restriction are located in various parts of the system and affect programs such as `ps`(1), `fstat`(1), and `netstat`(1). Information filtering is enabled as follows:

```
# sysctl -w security.curtain=1
```

SEE ALSO

`sysctl`(3), `options`(4), `paxctl`(8), `sysctl`(8), `veriexec`(8), `veriexecctl`(8), `veriexecgen`(8)

AUTHORS

Elad Efrat ⟨elad@NetBSD.org⟩

NAME

services_mkdb — generate the services databases

SYNOPSIS

services_mkdb [**−q**] [**−o** *database*] [file]
services_mkdb −u [file]

DESCRIPTION

services_mkdb creates a db(3) database for the specified file. If no file is specified, then /etc/services is used. The database is installed into /var/db/services.db. The file must be in the correct format (see services(5)).

The options are as follows:

−u Print the services file to stdout, omitting duplicate entries and comments.

−o *database*
Put the output databases in the named file.

−q Don't warn about duplicate services.

The databases are used by the C library services routines (see getservent(3)).

services_mkdb exits zero on success, non-zero on failure.

FILES

/var/db/services.db	The current services database.
/var/db/services.db.tmp	A temporary file.
/etc/services	The current services file.

SEE ALSO

db(3), getservent(3), services(5)

BUGS

Because **services_mkdb** guarantees not to install a partial destination file it must build a temporary file in the same file system and if successful use rename(2) to install over the destination file.

If **services_mkdb** fails it will leave the previous version of the destination file intact.

NAME

sesd — monitor SCSI Environmental Services Devices

SYNOPSIS

sesd [**-d**] [**-t** *poll-interval*] *device* [*device ...*]

DESCRIPTION

sesd monitors SCSI Environmental Services (or SAF-TE) devices for changes in state and logs such changes changes to the system error logger (see syslogd(8)). At least one device must be specified. When no other options are supplied, **sesd** detaches and becomes a daemon, by default waking up every 30 seconds to poll each device for a change in state.

The following options may be used:

-p *poll-interval*
Change the interval of polling from the default 30 seconds to the number of seconds specified.

-d Instead of detaching and becoming a daemon, stay attached to the controlling terminal and log changes there as well as via the system logger.

The user may then use getencstat(8) to get more detailed information about the state of the over enclosure device or objects within the enclosure device.

FILES

/dev/ses*N*
SCSI Environmental Services Devices

SEE ALSO

ses(4), getencstat(8), setencstat(8), setobjstat(8), syslogd(8)

BUGS

This is something of a toy, but it is better than nothing.

NAME

setencstat — set SCSI Environmental Services Device enclosure status

SYNOPSIS

setencstat *device enclosure_status*

DESCRIPTION

setencstat sets summary status for a SCSI Environmental Services (or SAF-TE) device. The enclosure status argument may take on the values:

0	Set the status to an **OK** state.
1	Set the status to an **UNRECOVERABLE** state.
2	Set the status to an **CRITICAL** state.
4	Set the status to an **NON-CRITICAL** state.
8	Set the status to an **INFORMATIONAL** state.

All the non-zero options may be combined.

Note that devices may simply and silently ignore the setting of these values.

FILES

`/dev/ses`*N*
SCSI Environmental Services Devices

SEE ALSO

ses(4), getencstat(8), sesd(8), setobjstat(8)

NAME

setkey — manually manipulate the IPsec SA/SP database

SYNOPSIS

setkey [**-knrv**] *file ...*
setkey [**-knrv**] **-c**
setkey [**-krv**] **-f** *filename*
setkey [**-aklPrv**] **-D**
setkey [**-Pvp**] **-F**
setkey [**-H**] **-x**
setkey [**-?V**]

DESCRIPTION

setkey adds, updates, dumps, or flushes Security Association Database (SAD) entries as well as Security Policy Database (SPD) entries in the kernel.

setkey takes a series of operations from standard input (if invoked with **-c**) or the file named *filename* (if invoked with **-f** *filename*).

(no flag)
Dump the SAD entries or SPD entries contained in the specified *file*.

-? Print short help.

-a **setkey** usually does not display dead SAD entries with **-D**. If **-a** is also specified, the dead SAD entries will be displayed as well. A dead SAD entry is one that has expired but remains in the system because it is referenced by some SPD entries.

-D Dump the SAD entries. If **-P** is also specified, the SPD entries are dumped. If **-p** is specified, the ports are displayed.

-F Flush the SAD entries. If **-P** is also specified, the SPD entries are flushed.

-H Add hexadecimal dump in **-x** mode.

-h On NetBSD, synonym for **-H**. On other systems, synonym for **-?**.

-k Use semantics used in kernel. Available only in Linux. See also **-r**.

-l Loop forever with short output on **-D**.

-n No action. The program will check validity of the input, but no changes to the SPD will be made.

-r Use semantics described in IPsec RFCs. This mode is default. For details see section **RFC vs Linux kernel semantics**. Available only in Linux. See also **-k**.

-x Loop forever and dump all the messages transmitted to the PF_KEY socket. **-xx** prints the unformatted timestamps.

-V Print version string.

-v Be verbose. The program will dump messages exchanged on the PF_KEY socket, including messages sent from other processes to the kernel.

Configuration syntax

With **-c** or **-f** on the command line, **setkey** accepts the following configuration syntax. Lines starting with hash signs ('#') are treated as comment lines.

add [**−46n**] *src dst protocol spi* [*extensions*] *algorithm ...* ;
: Add an SAD entry. add can fail for multiple reasons, including when the key length does not match the specified algorithm.

get [**−46n**] *src dst protocol spi* ;
: Show an SAD entry.

delete [**−46n**] *src dst protocol spi* ;
: Remove an SAD entry.

deleteall [**−46n**] *src dst protocol* ;
: Remove all SAD entries that match the specification.

flush [*protocol*] ;
: Clear all SAD entries matched by the options. **−F** on the command line achieves the same functionality.

dump [*protocol*] ;
: Dumps all SAD entries matched by the options. **−D** on the command line achieves the same functionality.

spdadd [**−46n**] *src_range dst_range upperspec label policy* ;
: Add an SPD entry.

spdadd tagged *tag policy* ;
: Add an SPD entry based on a PF tag. *tag* must be a string surrounded by double quotes.

spddelete [**−46n**] *src_range dst_range upperspec* **−P** *direction* ;
: Delete an SPD entry.

spdflush ;
: Clear all SPD entries. **−FP** on the command line achieves the same functionality.

spddump ;
: Dumps all SPD entries. **−DP** on the command line achieves the same functionality.

Meta-arguments are as follows:

src
dst Source/destination of the secure communication is specified as an IPv4/v6 address, and an optional port number between square brackets. **setkey** can resolve a FQDN into numeric addresses. If the FQDN resolves into multiple addresses, **setkey** will install multiple SAD/SPD entries into the kernel by trying all possible combinations. **−4**, **−6**, and **−n** restrict the address resolution of FQDN in certain ways. **−4** and **−6** restrict results into IPv4/v6 addresses only, respectively. **−n** avoids FQDN resolution and requires addresses to be numeric addresses.

protocol
: *protocol* is one of following:

esp	ESP based on rfc2406
esp-old	ESP based on rfc1827
ah	AH based on rfc2402
ah-old	AH based on rfc1826
ipcomp	IPComp
tcp	TCP-MD5 based on rfc2385

spi Security Parameter Index (SPI) for the SAD and the SPD. *spi* must be a decimal number, or a hexadecimal number with a "0x" prefix. SPI values between 0 and 255 are reserved for future use by IANA and cannot be used. TCP-MD5 associations must use 0x1000 and therefore only have per-host granularity at this time.

extensions

take some of the following:

-m *mode* Specify a security protocol mode for use. *mode* is one of following: `transport`, `tunnel`, or `any`. The default value is `any`.

-r *size* Specify window size of bytes for replay prevention. *size* must be decimal number in 32-bit word. If *size* is zero or not specified, replay checks don't take place.

-u *id* Specify the identifier of the policy entry in the SPD. See *policy*.

-f *pad_option*

defines the content of the ESP padding. *pad_option* is one of following:

`zero-pad`	All the paddings are zero.
`random-pad`	A series of randomized values are used.
`seq-pad`	A series of sequential increasing numbers started from 1 are used.

-f `nocyclic-seq`

Don't allow cyclic sequence numbers.

-lh *time*
-ls *time* Specify hard/soft life time duration of the SA measured in seconds.

-bh *bytes*
-bs *bytes*

Specify hard/soft life time duration of the SA measured in bytes transported.

-ctx *doi algorithm context-name*

Specify an access control label. The access control label is interpreted by the LSM (e.g., SELinux). Ultimately, it enables MAC on network communications.

doi The domain of interpretation, which is used by the IKE daemon to identify the domain in which negotiation takes place.

algorithm

Indicates the LSM for which the label is generated (e.g., SELinux).

context-name

The string representation of the label that is interpreted by the LSM.

algorithm

-E *ealgo key*

Specify an encryption algorithm *ealgo* for ESP.

-E *ealgo key* **-A** *aalgo key*

Specify an encryption algorithm *ealgo*, as well as a payload authentication algorithm *aalgo*, for ESP.

-A *aalgo key*

Specify an authentication algorithm for AH.

-C *calgo* [**-R**]

Specify a compression algorithm for IPComp. If **-R** is specified, the *spi* field value will be used as the IPComp CPI (compression parameter index) on wire as-is. If **-R** is not specified, the kernel will use well-known CPI on wire, and *spi* field will be used only as an index for kernel internal usage.

key must be a double-quoted character string, or a series of hexadecimal digits preceded by "`0x`".

Possible values for *ealgo*, *aalgo*, and *calgo* are specified in the **Algorithms** sections.

src_range
dst_range

These select the communications that should be secured by IPsec. They can be an IPv4/v6 address or an IPv4/v6 address range, and may be accompanied by a TCP/UDP port specification. This takes the following form:

address
address/prefixlen
address[port]
address/prefixlen[port]

prefixlen and *port* must be decimal numbers. The square brackets around *port* are really necessary, they are not man page meta-characters. For FQDN resolution, the rules applicable to *src* and *dst* apply here as well.

upperspec
Upper-layer protocol to be used. You can use one of the words in `/etc/protocols` as *upperspec*, or `icmp6`, `ip4`, or `any`. `any` stands for "any protocol". You can also use the protocol number. You can specify a type and/or a code of ICMPv6 when the upper-layer protocol is ICMPv6. The specification can be placed after `icmp6`. A type is separated from a code by single comma. A code must always be specified. When a zero is specified, the kernel deals with it as a wildcard. Note that the kernel can not distinguish a wildcard from an ICPMv6 type of zero. For example, the following means that the policy doesn't require IPsec for any inbound Neighbor Solicitation.

```
spdadd ::/0 ::/0 icmp6 135,0 -P in none;
```

Note: *upperspec* does not work against forwarding case at this moment, as it requires extra reassembly at the forwarding node (not implemented at this moment). There are many protocols in `/etc/protocols`, but all protocols except of TCP, UDP, and ICMP may not be suitable to use with IPsec. You have to consider carefully what to use.

label *label* is the access control label for the policy. This label is interpreted by the LSM (e.g., SELinux). Ultimately, it enables MAC on network communications. When a policy contains an access control label, SAs negotiated with this policy will contain the label. It's format:
−ctx *doi algorithm context-name*

doi The domain of interpretation, which is used by the IKE daemon to identify the domain in which negotiation takes place.

algorithm
Indicates the LSM for which the label is generated (e.g., SELinux).

context-name
The string representation of the label that is interpreted by the LSM.

policy
policy is in one of the following three formats:

−P *direction [priority specification]* discard
−P *direction [priority specification]* none
−P *direction [priority specification]* ipsec
    protocol/mode/src-dst/level [...]

You must specify the direction of its policy as *direction*. Either *out*, *in*, or *fwd* can be used.

priority specification is used to control the placement of the policy within the SPD. Policy position is determined by a signed integer where higher priorities indicate the policy is placed closer to the beginning of the list and lower priorities indicate the policy is placed closer to the end of the list. Policies with equal priorities are added at the end of groups of such policies.

Priority can only be specified when setkey has been compiled against kernel headers that support policy priorities (Linux >= 2.6.6). If the kernel does not support priorities, a warning message will be printed the first time a priority specification is used. Policy priority takes one of the following formats:

{priority,prio} offset
: *offset* is an integer in the range from −2147483647 to 214783648.

{priority,prio} base {+,−} offset
: *base* is either `low (−1073741824)`, `def (0)`, or `high (1073741824)`

  offset is an unsigned integer. It can be up to 1073741824 for positive offsets, and up to 1073741823 for negative offsets.

`discard` means the packet matching indexes will be discarded. `none` means that IPsec operation will not take place onto the packet. `ipsec` means that IPsec operation will take place onto the packet.

The *protocol/mode/src-dst/level* part specifies the rule how to process the packet. Either `ah`, `esp`, or `ipcomp` must be used as *protocol*. *mode* is either `transport` or `tunnel`. If *mode* is `tunnel`, you must specify the end-point addresses of the SA as *src* and *dst* with '-' between these addresses, which is used to specify the SA to use. If *mode* is `transport`, both *src* and *dst* can be omitted. *level* is to be one of the following: `default`, `use`, `require`, or `unique`. If the SA is not available in every level, the kernel will ask the key exchange daemon to establish a suitable SA. `default` means the kernel consults the system wide default for the protocol you specified, e.g. the `esp_trans_deflev` sysctl variable, when the kernel processes the packet. `use` means that the kernel uses an SA if it's available, otherwise the kernel keeps normal operation. `require` means SA is required whenever the kernel sends a packet matched with the policy. `unique` is the same as `require`; in addition, it allows the policy to match the unique out-bound SA. You just specify the policy level `unique`, `racoon`(8) will configure the SA for the policy. If you configure the SA by manual keying for that policy, you can put a decimal number as the policy identifier after `unique` separated by a colon ':' like: `unique:number` in order to bind this policy to the SA. `number` must be between 1 and 32767. It corresponds to *extensions* **−u** of the manual SA configuration. When you want to use SA bundle, you can define multiple rules. For example, if an IP header was followed by an AH header followed by an ESP header followed by an upper layer protocol header, the rule would be:
      `esp/transport//require ah/transport//require`;
The rule order is very important.

When NAT-T is enabled in the kernel, policy matching for ESP over UDP packets may be done on endpoint addresses and port (this depends on the system. System that do not perform the port check cannot support multiple endpoints behind the same NAT). When using ESP over UDP, you can specify port numbers in the endpoint addresses to get the correct matching. Here is an example:

```
spdadd 10.0.11.0/24[any] 10.0.11.33/32[any] any -P out ipsec
    esp/tunnel/192.168.0.1[4500]-192.168.1.2[30000]/require ;
```

These ports must be left unspecified (which defaults to 0) for anything other than ESP over UDP. They can be displayed in SPD dump using **setkey −DPp**.

Note that "`discard`" and "`none`" are not in the syntax described in `ipsec_set_policy`(3). There are a few differences in the syntax. See `ipsec_set_policy`(3) for detail.

Algorithms

The following list shows the supported algorithms. **protocol** and **algorithm** are almost orthogonal. These authentication algorithms can be used as *aalgo* in **−A** *aalgo* of the *protocol* parameter:

algorithm	keylen (bits)	
hmac-md5	128	ah: rfc2403
	128	ah-old: rfc2085
hmac-sha1	160	ah: rfc2404

	160	ah-old: 128bit ICV (no document)
keyed-md5	128	ah: 96bit ICV (no document)
	128	ah-old: rfc1828
keyed-sha1	160	ah: 96bit ICV (no document)
	160	ah-old: 128bit ICV (no document)
null	0 to 2048	for debugging
hmac-sha256	256	ah: 96bit ICV (draft-ietf-ipsec-ciph-sha-256-00)
	256	ah-old: 128bit ICV (no document)
hmac-sha384	384	ah: 96bit ICV (no document)
	384	ah-old: 128bit ICV (no document)
hmac-sha512	512	ah: 96bit ICV (no document)
	512	ah-old: 128bit ICV (no document)
hmac-ripemd160	160	ah: 96bit ICV (RFC2857)
		ah-old: 128bit ICV (no document)
aes-xcbc-mac	128	ah: 96bit ICV (RFC3566)
	128	ah-old: 128bit ICV (no document)
tcp-md5	8 to 640	tcp: rfc2385

These encryption algorithms can be used as *ealgo* in **-E** *ealgo* of the *protocol* parameter:

algorithm	keylen (bits)	
des-cbc	64	esp-old: rfc1829, esp: rfc2405
3des-cbc	192	rfc2451
null	0 to 2048	rfc2410
blowfish-cbc	40 to 448	rfc2451
cast128-cbc	40 to 128	rfc2451
des-deriv	64	ipsec-ciph-des-derived-01
3des-deriv	192	no document
rijndael-cbc	128/192/256	rfc3602
twofish-cbc	0 to 256	draft-ietf-ipsec-ciph-aes-cbc-01
aes-ctr	160/224/288	draft-ietf-ipsec-ciph-aes-ctr-03
camellia-cbc	128/192/256	rfc4312

Note that the first 128 bits of a key for `aes-ctr` will be used as AES key, and the remaining 32 bits will be used as nonce.

These compression algorithms can be used as *calgo* in **-C** *calgo* of the *protocol* parameter:

algorithm	
deflate	rfc2394

RFC vs Linux kernel semantics

The Linux kernel uses the *fwd* policy instead of the *in* policy for packets what are forwarded through that particular box.

In *kernel* mode, **setkey** manages and shows policies and SAs exactly as they are stored in the kernel.

In *RFC* mode, **setkey**

creates *fwd* policies for every *in* policy inserted

(not implemented yet) filters out all *fwd* policies

RETURN VALUES

The command exits with 0 on success, and non-zero on errors.

EXAMPLES

```
add 3ffe:501:4819::1 3ffe:501:481d::1 esp 123457
        -E des-cbc 0x3ffe05014819ffff ;

add -6 myhost.example.com yourhost.example.com ah 123456
        -A hmac-sha1 "AH SA configuration!" ;

add 10.0.11.41 10.0.11.33 esp 0x10001
        -E des-cbc 0x3ffe05014819ffff
        -A hmac-md5 "authentication!!" ;

get 3ffe:501:4819::1 3ffe:501:481d::1 ah 123456 ;

flush ;

dump esp ;

spdadd 10.0.11.41/32[21] 10.0.11.33/32[any] any
        -P out ipsec esp/tunnel/192.168.0.1-192.168.1.2/require ;

add 10.1.10.34 10.1.10.36 tcp 0x1000 -A tcp-md5 "TCP-MD5 BGP secret" ;

add 10.0.11.41 10.0.11.33 esp 0x10001
        -ctx 1 1 "system_u:system_r:unconfined_t:SystemLow-SystemHigh"
        -E des-cbc 0x3ffe05014819ffff;

spdadd 10.0.11.41 10.0.11.33 any
        -ctx 1 1 "system_u:system_r:unconfined_t:SystemLow-SystemHigh"
        -P out ipsec esp/transport//require ;
```

SEE ALSO

`ipsec_set_policy`(3), `racoon`(8), `sysctl`(8)

Changed manual key configuration for IPsec, October 1999, http://www.kame.net/newsletter/19991007/.

HISTORY

The **setkey** command first appeared in the WIDE Hydrangea IPv6 protocol stack kit. The command was completely re-designed in June 1998.

BUGS

setkey should report and handle syntax errors better.

For IPsec gateway configuration, *src_range* and *dst_range* with TCP/UDP port numbers does not work, as the gateway does not reassemble packets (it cannot inspect upper-layer headers) .

NAME

setnetbootinfo — configure Alpha network bootstrap program

SYNOPSIS

/usr/mdec/setnetbootinfo [**-vf**] [**-o** *outfile*] [**-a** *ether-address* | **-h** *ether-host*] *infile*
/usr/mdec/setnetbootinfo [**-v**] **-u -o** *outfile infile*

DESCRIPTION

The **setnetbootinfo** utility configures the NetBSD/alpha network bootstrap program so that it can be used to bootstrap systems with old firmware revisions.

The NetBSD/alpha network bootstrap program needs to have the ethernet address of the interface being used to boot the system available when querying other hosts on the network for bootstrapping information. Alpha systems with old firmware revisions provide no way for network bootstrap programs to determine the ethernet address of the interface that they are booting from, and so the NetBSD/alpha network bootstrap program must find that information in another way. (Newer firmware revisions include the ethernet address in the name of the device that is being booted from.) The **setnetbootinfo** utility encodes an ethernet address (and other information) directly into the network bootstrap program.

The options recognized by **setnetbootinfo** are as follows:

-a *ether-address*
Encode the given ethernet address into the network bootstrap program. (This option and the **-h** option are mutually exclusive.)

-f Force the address information being encoded in the bootstrap program to be used regardless of whether or not the bootstrap program can get address information from the booting system's firmware.

-h *ether-host*
Encode the ethernet address of the specified host into the network bootstrap program. The host's name is translated to an ethernet address using the `ether_hostton`(3) function. (This option and the **-a** option are mutually exclusive.)

-o *outfile*
Output the resulting bootstrap program into the file named by *outfile*, replacing it if it already exists. If the **-o** flag is not specified, the output file name will be the name of the input bootstrap program concatenated with a period and the digits of the ethernet address being encoded. For instance, if the input file is named `/usr/mdec/netboot` and is being configured to encode the ethernet address `08:00:2b:bd:5d:fd`, then the default output file name would be `/usr/mdec/netboot.08002bbd5dfd`. It is safe to set the output file name to be the same as the input file name; the input file is read in its entirety before the output file is modified.

-u Remove configuration information from the specified network bootstrap program. If this option is used, an output file name must be specified with the **-o** option, and neither the **-a** or the **-h** options may be specified.

-v Verbose mode.

FILES

`/usr/mdec/netboot`	network bootstrap program

SEE ALSO

`boot`(8), `bootpd`(8)

HISTORY

The NetBSD/alpha **`setnetbootinfo`** command first appeared in NetBSD 1.3.

AUTHORS

The **`setnetbootinfo`** utility was written by Chris Demetriou.

NAME

setobjstat — set SCSI Environmental Services Device object status

SYNOPSIS

setobjstat *device objectid stat0 stat1 stat2 stat3*

DESCRIPTION

setobjstat sets the object status for a SCSI Environmental Services (or SAF-TE) device. The *objectid* argument may be determined by running getencstat(8).

The status fields are partially common (first byte only, which must have a value of 0x80 contained in it), but otherwise quite device specific. A complete discussion of the possible values is impractical here. Please refer to the ANSI SCSI specification (available on the FTP site ftp.t10.org).

Note that devices may simply and silently ignore the setting of these values.

FILES

/dev/ses*N*
SCSI Environmental Services Devices

SEE ALSO

ses(4), getencstat(8), sesd(8), setencstat(8)

NAME

sftp-server — SFTP server subsystem

SYNOPSIS

sftp-server [**-f** *log_facility*] [**-l** *log_level*]

DESCRIPTION

sftp-server is a program that speaks the server side of SFTP protocol to stdout and expects client requests from stdin. **sftp-server** is not intended to be called directly, but from sshd(8) using the **Subsystem** option.

Command-line flags to **sftp-server** should be specified in the **Subsystem** declaration. See sshd_config(5) for more information.

Valid options are:

-f *log_facility*
: Specifies the facility code that is used when logging messages from **sftp-server**. The possible values are: DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is AUTH.

-l *log_level*
: Specifies which messages will be logged by **sftp-server**. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. INFO and VERBOSE log transactions that **sftp-server** performs on behalf of the client. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output. The default is ERROR.

SEE ALSO

sftp(1), ssh(1), sshd_config(5), sshd(8)

T. Ylonen and S. Lehtinen, *SSH File Transfer Protocol*, draft-ietf-secsh-filexfer-00.txt, January 2001, work in progress material.

HISTORY

sftp-server first appeared in OpenBSD 2.8.

AUTHORS

Markus Friedl ⟨markus@openbsd.org⟩

NAME

/usr/mdec/sgivol — configure SGI Volume Header

SYNOPSIS

/usr/mdec/sgivol [**-fq**] *device*
/usr/mdec/sgivol [**-fq**] **-i** [**-h** *vhsize*] *device*
/usr/mdec/sgivol [**-fq**] **-r** *vhfilename diskfilename device*
/usr/mdec/sgivol [**-fq**] **-w** *vhfilename diskfilename device*
/usr/mdec/sgivol [**-fq**] **-d** *vhfilename device*
/usr/mdec/sgivol [**-fq**] **-p** *partno partfirst partblocks parttype device*

DESCRIPTION

The **/usr/mdec/sgivol** program prepares an SGI Volume Header to be used to boot NetBSD. The PROM is able to load executables within the header, which in turn are used to load the kernel from another file system.

OPTIONS

The following options are available:

-f Force the operation. Do not ask the user before proceeding.

-h Set the size of the newly initialized volume header in blocks. One block is 512 bytes. The default volume header size is 3135 blocks (1.53MB).

-q Suppress output.

PARTITION TYPES

The numerical partition types for the volume header include:

0:	Volume Header
1:	Replicated Tracks
2:	Replicated Sectors
3:	Raw
4:	BSD4.2 file system
5:	SysV file system
6:	Entire Volume (all disk blocks)
7:	EFS
8:	Logical Volume
9:	Raw Logical Volume
10:	XFS
11:	XFS Log
12:	XLV Volume
13:	XVM Volume

EXAMPLES

To display the existing volume header and partition table on disk "sd0":

```
sgivol sd0
```

To initialize a new volume header 42 512-byte blocks large on disk "sd0":

```
sgivol -i -h 42 sd0
```

To copy a file `boot` from the volume header to local file `/tmp/boot` on disk "sd0":

```
sgivol -r boot /tmp/boot sd0
```

To copy a local file `/usr/mdec/ip2xboot` to the volume header as `boot` on disk “sd0”:

```
sgivol -w boot /usr/mdec/ip2xboot sd0
```

To delete the existing file `boot` from the volume header on disk “sd0”:

```
sgivol -d boot sd0
```

To change partition 0 to type 4 (BSD4.2) beginning at block offset 3200 and continue for 28000 blocks on disk “sd0”:

```
sgivol -p 0 3200 28000 4 sd0
```

SEE ALSO

`boot`(8)

NAME

showmount — show remote NFS mounts on host

SYNOPSIS

showmount [**−ade3**] [*host*]

DESCRIPTION

showmount shows status information about the NFS server on *host*. By default it prints the names of all hosts that have NFS file systems mounted on the host. See *NFS: Network File System Protocol Specification*, RFC 1094, Appendix A, and *NFS: Network File System Version 3 Protocol Specification*, Appendix I, for a detailed description of the protocol.

−a List all mount points in the form:
  host:*dirpath*

−d List directory paths of mount points instead of hosts

−e Show the *host*'s exports list

−3 Use mount protocol Version 3, compatible with NFS Version 3.

SEE ALSO

mount(8), mountd(8)

HISTORY

The **showmount** utility first appeared in 4.4BSD.

BUGS

The mount daemon running on the server only has an idea of the actual mounts, since the NFS server is stateless. **showmount** will only display the information as accurately as the mount daemon reports it.

NAME

showq − list the Postfix mail queue

SYNOPSIS

showq [generic Postfix daemon options]

DESCRIPTION

The **showq**(8) daemon reports the Postfix mail queue status. It is the program that emulates the sendmail ‘mailq’ command.

The **showq**(8) daemon can also be run in stand-alone mode by the superuser. This mode of operation is used to emulate the ‘mailq’ command while the Postfix mail system is down.

SECURITY

The **showq**(8) daemon can run in a chroot jail at fixed low privilege, and takes no input from the client. Its service port is accessible to local untrusted users, so the service can be susceptible to denial of service attacks.

STANDARDS

None. The **showq**(8) daemon does not interact with the outside world.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **showq**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

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.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

duplicate_filter_limit (1000)
The maximal number of addresses remembered by the address duplicate filter for **aliases**(5) or **virtual**(5) alias expansion, or for **showq**(8) queue displays.

empty_address_recipient (MAILER-DAEMON)
The recipient of mail addressed to the null address.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

FILES

/var/spool/postfix, queue directories

SEE ALSO

pickup(8), local mail pickup service
cleanup(8), canonicalize and enqueue mail
qmgr(8), queue manager
postconf(5), configuration parameters
master(8), process manager
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

shutdown — close down the system at a given time

SYNOPSIS

shutdown [**-b** *bootstr*] [**-Ddfhknpr**] *time* [*message ...* | -]

DESCRIPTION

shutdown provides an automated shutdown procedure for super-users to nicely notify users when the system is shutting down, saving them from system administrators, hackers, and gurus, who would otherwise not bother with such niceties.

Available friendlinesses:

-b *bootstr*
: The given *bootstr* is passed to reboot(8) for the benefit of those systems that can pass boot arguments to the firmware. Currently, this only affects sun3 and sparc machines.

-d **shutdown** will pass the **-d** flag to reboot(8) or halt(8) to request a kernel core dump. If neither the **-h** or **-r** flags are specified, then **-d** also implies **-r**.

-f **shutdown** arranges, in the manner of fastboot(8), for the file systems *not to be* checked on reboot.

-h The system is halted at the specified *time*, using halt(8).

-k Kick everybody off. The **-k** option does not actually halt the system, but leaves the system multi-user with logins disabled (for all but super-user).

-n Prevent the normal sync(2) before stopping.

-p The system is powered down at the specified *time*, using halt(8). If the powerdown fails, or the system does not support software powerdown, the system will simply halt instead.

-r The system is rebooted at the specified *time*, using reboot(8).

-D Prevents **shutdown** from detaching from the tty with fork(2)/ exit(3).

time *Time* is the time at which **shutdown** will bring the system down and may be the word *now* or a future time in one of two formats: *+number*, or *[[[[[cc]yy]mm]dd]hh]mm*, where the century, year, month, day, and hour may be defaulted to the current system values. The first form brings the system down *number* minutes from the current time; the second brings the system down at the absolute time specified. If the century is not specified, it defaults to 1900 for years between 69 and 99, or 2000 for years between 0 and 68. A leading zero in the "yy" value is *not* optional.

message ...
: Any other arguments comprise the warning message that is broadcast to users currently logged into the system.

- If - is supplied as the only argument after the time, the warning message is read from the standard input.

BEHAVIOR

At intervals, becoming more frequent as apocalypse approaches and starting at ten hours before shutdown, warning messages are displayed on the terminals of all users logged in. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating /etc/nologin and copying the warning message there. If this file exists when a user attempts to log in, login(1) prints its contents and exits. The file is removed just before **shutdown** exits.

At shutdown time, a message is written in the system log containing the time of shutdown, who initiated the shutdown, and the reason. Next a message is printed announcing the start of the system shutdown hooks. Then the shutdown hooks in `/etc/rc.shutdown` are run, and a message is printed indicating that they have completed. After a short delay, **shutdown** runs `halt`(8) or `reboot`(8), or sends a terminate signal to `init`(8) to bring the system down to single-user mode, depending on the choice of options.

The time of the shutdown and the warning message are placed in `/etc/nologin` and should be used to tell the users why the system is going down, when it will be back up, and to share any other pertinent information.

FILES

`/etc/nologin`	tells `login`(1) not to let anyone log in
`/fastboot`	tells `rc`(8) not to run `fsck`(8) when rebooting
`/etc/rc.shutdown`	System shutdown commands

SEE ALSO

`login`(1), `wall`(1), `fastboot`(8), `halt`(8), `init`(8), `poweroff`(8), `reboot`(8)

BACKWARD COMPATIBILITY

The hours and minutes in the second time format may be separated by a colon (":") for backward compatibility.

HISTORY

The **shutdown** command appeared in 4.0BSD.

NAME

slapacl − Check access to a list of attributes.

SYNOPSIS

SBINDIR/slapacl −b DN [−d level] [−D authcDN | −U authcID] [−f slapd.conf] [−F confdir] [−o name[=value]] [−u] [−v] [−X authzID | −o authzDN=DN] [attr[/access][:value]] [...]

DESCRIPTION

Slapacl is used to check the behavior of the slapd in verifying access to data according to ACLs, as specified in **slapd.access**(5). It opens the **slapd.conf**(5) configuration file, reads in the **access** directives, and then parses the **attr** list given on the command-line; if none is given, access to the **entry** pseudo-attribute is tested.

OPTIONS

−b *DN* specify the **DN** which access is requested to; the corresponding entry is fetched from the database, and thus it must exist. The DN is also used to determine what rules apply; thus, it must be in the naming context of a configured database. See also **−u**.

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−D *authcDN*
specify a DN to be used as identity through the test session when selecting appropriate **<by>** clauses in access lists.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

Possible options/values specific to **slapacl** are:

```
authzDN
domain
peername
sasl_ssf
sockname
sockurl
ssf
tls_ssf
transport_ssf
```

See the related fields in **slapd.access**(5) for details.

−u do not fetch the entry from the database. In this case, if the entry does not exist, a fake entry with the DN given with the **−b** option is used, with no attributes. As a consequence, those rules that depend on the contents of the target object will not behave as with the real object. The DN given with the **−b** option is still used to select what rules apply; thus, it must be in the naming context of a configured database. See also **−b**.

−U *authcID*
specify an ID to be mapped to a **DN** as by means of **authz-regexp** or **authz-rewrite** rules (see **slapd.conf**(5) for details); mutually exclusive with **−D**.

−v enable verbose mode.

−X *authzID*
specify an authorization ID to be mapped to a **DN** as by means of **authz-regexp** or **authz-rewrite** rules (see **slapd.conf**(5) for details); mutually exclusive with **−o** *authzDN=DN*.

EXAMPLES

The command

```
    SBINDIR/slapacl -f /ETCDIR/slapd.conf -v \
  -U bjorn -b "o=University of Michigan,c=US" \
      "o/read:University of Michigan"
```

tests whether the user *bjorn* can access the attribute *o* of the entry *o=University of Michigan,c=US* at *read* level.

SEE ALSO

ldap(3), **slapd**(8) **slaptest**(8) **slapauth**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slapacl − Check access to a list of attributes.

SYNOPSIS

/usr/sbin/slapacl −b DN [−d level] [−D authcDN | −U authcID] [−f slapd.conf] [−F confdir] [−o name[=value]] [−u] [−v] [−X authzID | −o authzDN=DN] [attr[/access][:value]] [...]

DESCRIPTION

Slapacl is used to check the behavior of the slapd in verifying access to data according to ACLs, as specified in **slapd.access**(5). It opens the **slapd.conf**(5) configuration file, reads in the **access** directives, and then parses the **attr** list given on the command-line; if none is given, access to the **entry** pseudo-attribute is tested.

OPTIONS

−b *DN* specify the **DN** which access is requested to; the corresponding entry is fetched from the database, and thus it must exist. The DN is also used to determine what rules apply; thus, it must be in the naming context of a configured database. See also **−u**.

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−D *authcDN*
specify a DN to be used as identity through the test session when selecting appropriate **<by>** clauses in access lists.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

Possible options/values specific to **slapacl** are:

```
authzDN
domain
peername
sasl_ssf
sockname
sockurl
ssf
tls_ssf
transport_ssf
```

See the related fields in **slapd.access**(5) for details.

−u do not fetch the entry from the database. In this case, if the entry does not exist, a fake entry with the DN given with the **−b** option is used, with no attributes. As a consequence, those rules that depend on the contents of the target object will not behave as with the real object. The DN given with the **−b** option is still used to select what rules apply; thus, it must be in the naming context of a configured database. See also **−b**.

−U *authcID*
specify an ID to be mapped to a **DN** as by means of **authz-regexp** or **authz-rewrite** rules (see **slapd.conf**(5) for details); mutually exclusive with **−D**.

−v enable verbose mode.

−X *authzID*
specify an authorization ID to be mapped to a **DN** as by means of **authz-regexp** or **authz-rewrite** rules (see **slapd.conf**(5) for details); mutually exclusive with **−o** *authzDN=DN*.

EXAMPLES

The command

```
    /usr/sbin/slapacl -f //etc/openldap/slapd.conf -v \
  -U bjorn -b "o=University of Michigan,c=US" \
      "o/read:University of Michigan"
```

tests whether the user *bjorn* can access the attribute *o* of the entry *o=University of Michigan,c=US* at *read* level.

SEE ALSO

ldap(3), **slapd**(8) **slaptest**(8) **slapauth**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slapadd − Add entries to a SLAPD database

SYNOPSIS

SBINDIR/slapadd [−b suffix] [−c] [−d level] [−f slapd.conf] [−F confdir] [−g] [−j lineno] [−l ldif-file] [−n dbnum] [−o name[=value]] [−q] [−s] [−S SID] [−u] [−v] [−w]

DESCRIPTION

Slapadd is used to add entries specified in LDAP Directory Interchange Format (LDIF) to a **slapd**(8) database. It opens the given database determined by the database number or suffix and adds entries corresponding to the provided LDIF to the database. Databases configured as **subordinate** of this one are also updated, unless **-g** is specified. The LDIF input is read from standard input or the specified file.

All files eventually created by **slapadd** will belong to the identity **slapadd** is run as, so make sure you either run **slapadd** with the same identity **slapd**(8) will be run as (see option **−u** in **slapd**(8)), or change file ownership before running **slapd**(8).

OPTIONS

−b *suffix*
Use the specified *suffix* to determine which database to add entries to. The −b cannot be used in conjunction with the **−n** option.

−c enable continue (ignore errors) mode.

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored. If dryrun mode is also specified, no conversion will occur.

−g disable subordinate gluing. Only the specified database will be processed, and not its glued subordinates (if any).

−j *lineno*
Jump to the specified line number in the LDIF file before processing any entries. This allows a load that was aborted due to errors in the input LDIF to be resumed after the errors are corrected.

−l *ldif-file*
Read LDIF from the specified file instead of standard input.

−n *dbnum*
Add entries to the *dbnum*−th database listed in the configuration file. The **−n** cannot be used in conjunction with the **−b** option.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−q enable quick (fewer integrity checks) mode. Does fewer consistency checks on the input data, and no consistency checks when writing the database. Improves the load time but if any errors or interruptions occur the resulting database will be unusable.

-s disable schema checking. This option is intended to be used when loading databases containing special objects, such as fractional objects on a partial replica. Loading normal objects which do not conform to schema may result in unexpected and ill behavior.

−S SID
Server ID to use in generated entryCSN. Also used for contextCSN if ‘−w’ is set as well. Defaults to 0.

−u enable dry-run (don’t write to backend) mode.

−v enable verbose mode.

−w write syncrepl context information. After all entries are added, the contextCSN will be updated with the greatest CSN in the database.

LIMITATIONS

Your **slapd**(8) should not be running when you do this to ensure consistency of the database.

slapadd may not provide naming or schema checks. It is advisable to use **ldapadd**(1) when adding new entries into an existing directory.

EXAMPLES

To import the entries specified in file **ldif** into your **slapd**(8) database give the command:

```
SBINDIR/slapadd -l ldif
```

SEE ALSO

ldap(3), **ldif**(5), **slapcat**(8), **ldapadd**(1), **slapd**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slapadd − Add entries to a SLAPD database

SYNOPSIS

/usr/sbin/slapadd [−b suffix] [−c] [−d level] [−f slapd.conf] [−F confdir] [−g] [−j lineno] [−l ldif-file] [−n dbnum] [−o name[=value]] [−q] [−s] [−S SID] [−u] [−v] [−w]

DESCRIPTION

Slapadd is used to add entries specified in LDAP Directory Interchange Format (LDIF) to a **slapd**(8) database. It opens the given database determined by the database number or suffix and adds entries corresponding to the provided LDIF to the database. Databases configured as **subordinate** of this one are also updated, unless **-g** is specified. The LDIF input is read from standard input or the specified file.

All files eventually created by **slapadd** will belong to the identity **slapadd** is run as, so make sure you either run **slapadd** with the same identity **slapd**(8) will be run as (see option **−u** in **slapd**(8)), or change file ownership before running **slapd**(8).

OPTIONS

−b *suffix*
Use the specified *suffix* to determine which database to add entries to. The −b cannot be used in conjunction with the **−n** option.

−c enable continue (ignore errors) mode.

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored. If dryrun mode is also specified, no conversion will occur.

−g disable subordinate gluing. Only the specified database will be processed, and not its glued subordinates (if any).

−j *lineno*
Jump to the specified line number in the LDIF file before processing any entries. This allows a load that was aborted due to errors in the input LDIF to be resumed after the errors are corrected.

−l *ldif-file*
Read LDIF from the specified file instead of standard input.

−n *dbnum*
Add entries to the *dbnum*−th database listed in the configuration file. The **−n** cannot be used in conjunction with the **−b** option.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−q enable quick (fewer integrity checks) mode. Does fewer consistency checks on the input data, and no consistency checks when writing the database. Improves the load time but if any errors or interruptions occur the resulting database will be unusable.

-s disable schema checking. This option is intended to be used when loading databases containing special objects, such as fractional objects on a partial replica. Loading normal objects which do not conform to schema may result in unexpected and ill behavior.

−S SID
Server ID to use in generated entryCSN. Also used for contextCSN if '−w' is set as well. Defaults to 0.

−u enable dry-run (don't write to backend) mode.

−v enable verbose mode.

−w write syncrepl context information. After all entries are added, the contextCSN will be updated with the greatest CSN in the database.

LIMITATIONS

Your **slapd**(8) should not be running when you do this to ensure consistency of the database.

slapadd may not provide naming or schema checks. It is advisable to use **ldapadd**(1) when adding new entries into an existing directory.

EXAMPLES

To import the entries specified in file **ldif** into your **slapd**(8) database give the command:

```
/usr/sbin/slapadd -l ldif
```

SEE ALSO

ldap(3), **ldif**(5), **slapcat**(8), **ldapadd**(1), **slapd**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slapauth − Check a list of string-represented IDs for LDAP authc/authz

SYNOPSIS

SBINDIR/slapauth [−d level] [−f slapd.conf] [−F confdir] [−M mech] [−o name[=value]] [−R realm] [−U authcID] [−v] [−X authzID] ID [...]

DESCRIPTION

Slapauth is used to check the behavior of the slapd in mapping identities for authentication and authorization purposes, as specified in **slapd.conf**(5). It opens the **slapd.conf**(5) configuration file, reads in the **authz-policy** and **authz-regexp** directives, and then parses the **ID** list given on the command-line.

OPTIONS

−d *level*
: enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
: specify an alternative **slapd.conf**(5) file.

−F *confdir*
: specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−M *mech*
: specify a mechanism.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see '−s' in slapd(8))
syslog-level=<level> (see '−S' in slapd(8))
syslog-user=<user>   (see '−l' in slapd(8))
```

−R *realm*
: specify a realm.

−U *authcID*
: specify an ID to be used as *authcID* throughout the test session. If present, and if no **authzID** is given, the IDs in the ID list are treated as **authzID**.

−X *authzID*
: specify an ID to be used as *authzID* throughout the test session. If present, and if no **authcID** is given, the IDs in the ID list are treated as **authcID**. If both *authcID* and *authzID* are given via command line switch, the ID list cannot be present.

−v enable verbose mode.

EXAMPLES

The command

```
    SBINDIR/slapauth -f /ETCDIR/slapd.conf -v \
  -U bjorn -X u:bjensen
```

tests whether the user *bjorn* can assume the identity of the user *bjensen* provided the directives

```
authz-policy from
authz-regexp "^uid=([^,]+).*,cn=auth$"
        "ldap:///dc=example,dc=net??sub?uid=$1"
```

are defined in **slapd.conf**(5).

SEE ALSO

ldap(3), **slapd**(8) **slaptest**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slapauth − Check a list of string-represented IDs for LDAP authc/authz

SYNOPSIS

/usr/sbin/slapauth [−d level] [−f slapd.conf] [−F confdir] [−M mech] [−o name[=value]] [−R realm] [−U authcID] [−v] [−X authzID] ID [...]

DESCRIPTION

Slapauth is used to check the behavior of the slapd in mapping identities for authentication and authorization purposes, as specified in **slapd.conf**(5). It opens the **slapd.conf**(5) configuration file, reads in the **authz-policy** and **authz-regexp** directives, and then parses the **ID** list given on the command-line.

OPTIONS

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−M *mech*
specify a mechanism.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−R *realm*
specify a realm.

−U *authcID*
specify an ID to be used as *authcID* throughout the test session. If present, and if no **authzID** is given, the IDs in the ID list are treated as **authzID**.

−X *authzID*
specify an ID to be used as *authzID* throughout the test session. If present, and if no **authcID** is given, the IDs in the ID list are treated as **authcID**. If both *authcID* and *authzID* are given via command line switch, the ID list cannot be present.

−v enable verbose mode.

EXAMPLES

The command

```
/usr/sbin/slapauth -f //etc/openldap/slapd.conf -v \
-U bjorn -X u:bjensen
```

tests whether the user *bjorn* can assume the identity of the user *bjensen* provided the directives

```
authz-policy from
authz-regexp "^uid=([^,]+).*,cn=auth$"
        "ldap:///dc=example,dc=net??sub?uid=$1"
```

are defined in **slapd.conf**(5).

SEE ALSO

ldap(3), **slapd**(8) **slaptest**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slapcat − SLAPD database to LDIF utility

SYNOPSIS

SBINDIR/slapcat [−a filter] [−b suffix] [−c] [−d level] [−f slapd.conf] [−F confdir] [−g] [−l ldif-file] [−n dbnum] [−o name[=value]] [−s subtree-dn] [−v]

DESCRIPTION

Slapcat is used to generate an LDAP Directory Interchange Format (LDIF) output based upon the contents of a **slapd**(8) database. It opens the given database determined by the database number or suffix and writes the corresponding LDIF to standard output or the specified file. Databases configured as **subordinate** of this one are also output, unless **-g** is specified.

The entry records are presented in database order, not superior first order. The entry records will include all (user and operational) attributes stored in the database. The entry records will not include dynamically generated attributes (such as subschemaSubentry).

The output of slapcat is intended to be used as input to **slapadd**(8). The output of slapcat cannot generally be used as input to **ldapadd**(1) or other LDAP clients without first editing the output. This editing would normally include reordering the records into superior first order and removing no-user-modification operational attributes.

OPTIONS

−a *filter*
Only dump entries matching the asserted filter. For example

```
slapcat -a \
   "(!(entryDN:dnSubtreeMatch:=ou=People,dc=example,dc=com))"
```

will dump all but the "ou=People,dc=example,dc=com" subtree of the "dc=example,dc=com" database.

−b *suffix*
Use the specified *suffix* to determine which database to generate output for. The −b cannot be used in conjunction with the **−n** option.

−c Enable continue (ignore errors) mode.

−d *level*
Enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
Specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−g disable subordinate gluing. Only the specified database will be processed, and not its glued subordinates (if any).

−l *ldif-file*
Write LDIF to specified file instead of standard output.

−n *dbnum*
Generate output for the *dbnum*−th database listed in the configuration file. The **−n** cannot be used in conjunction with the **−b** option.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see '−s' in slapd(8))
syslog-level=<level> (see '−S' in slapd(8))
syslog-user=<user>   (see '−l' in slapd(8))
```

−s *subtree-dn*
: Only dump entries in the subtree specified by this DN. Implies '-b subtree-dn' if no **−b** or **−n** option is given.

−v
: Enable verbose mode.

LIMITATIONS

For some backend types, your **slapd**(8) should not be running (at least, not in read-write mode) when you do this to ensure consistency of the database. It is always safe to run **slapcat** with the **slapd-bdb**(5), **slapd-hdb**(5), and **slapd-null**(5) backends.

EXAMPLES

To make a text backup of your SLAPD database and put it in a file called **ldif**, give the command:

```
SBINDIR/slapcat -l ldif
```

SEE ALSO

ldap(3), **ldif**(5), **slapadd**(8), **ldapadd**(1), **slapd**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slapcat − SLAPD database to LDIF utility

SYNOPSIS

/usr/sbin/slapcat [−a filter] [−b suffix] [−c] [−d level] [−f slapd.conf] [−F confdir] [−g] [−l ldif-file] [−n dbnum] [−o name[=value]] [−s subtree-dn] [−v]

DESCRIPTION

Slapcat is used to generate an LDAP Directory Interchange Format (LDIF) output based upon the contents of a **slapd**(8) database. It opens the given database determined by the database number or suffix and writes the corresponding LDIF to standard output or the specified file. Databases configured as **subordinate** of this one are also output, unless **-g** is specified.

The entry records are presented in database order, not superior first order. The entry records will include all (user and operational) attributes stored in the database. The entry records will not include dynamically generated attributes (such as subschemaSubentry).

The output of slapcat is intended to be used as input to **slapadd**(8). The output of slapcat cannot generally be used as input to **ldapadd**(1) or other LDAP clients without first editing the output. This editing would normally include reordering the records into superior first order and removing no-user-modification operational attributes.

OPTIONS

−a *filter*

Only dump entries matching the asserted filter. For example

```
slapcat -a \
   "(!(entryDN:dnSubtreeMatch:=ou=People,dc=example,dc=com))"
```

will dump all but the "ou=People,dc=example,dc=com" subtree of the "dc=example,dc=com" database.

−b *suffix*

Use the specified *suffix* to determine which database to generate output for. The −b cannot be used in conjunction with the **−n** option.

−c Enable continue (ignore errors) mode.

−d *level*

Enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*

Specify an alternative **slapd.conf**(5) file.

−F *confdir*

specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−g disable subordinate gluing. Only the specified database will be processed, and not its glued subordinates (if any).

−l *ldif-file*

Write LDIF to specified file instead of standard output.

−n *dbnum*

Generate output for the *dbnum*−th database listed in the configuration file. The **−n** cannot be used in conjunction with the **−b** option.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−s *subtree-dn*
: Only dump entries in the subtree specified by this DN. Implies ‘-b subtree-dn’ if no **−b** or **−n** option is given.

−v
: Enable verbose mode.

LIMITATIONS

For some backend types, your **slapd**(8) should not be running (at least, not in read-write mode) when you do this to ensure consistency of the database. It is always safe to run **slapcat** with the **slapd-bdb**(5), **slapd-hdb**(5), and **slapd-null**(5) backends.

EXAMPLES

To make a text backup of your SLAPD database and put it in a file called **ldif**, give the command:

```
/usr/sbin/slapcat -l ldif
```

SEE ALSO

ldap(3), **ldif**(5), **slapadd**(8), **ldapadd**(1), **slapd**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slapd − Stand-alone LDAP Daemon

SYNOPSIS

LIBEXECDIR/slapd [−[4|6]] [−T {acl|add|auth|cat|dn|index|passwd|test}] [−d debug−level] [−f slapd−config−file] [−F slapd−config−directory] [−h URLs] [−n service−name] [−s syslog−level] [−l syslog−local−user] [−o option[=value]] [−r directory] [−u user] [−g group] [−c cookie]

DESCRIPTION

Slapd is the stand-alone LDAP daemon. It listens for LDAP connections on any number of ports (default 389), responding to the LDAP operations it receives over these connections. **slapd** is typically invoked at boot time, usually out of **/etc/rc.local**. Upon startup, **slapd** normally forks and disassociates itself from the invoking tty. If configured in the config file (or config directory), the **slapd** process will print its process ID (see **getpid**(2)) to a **.pid** file, as well as the command line options during invocation to an **.args** file (see **slapd.conf**(5)). If the **−d** flag is given, even with a zero argument, **slapd** will not fork and disassociate from the invoking tty.

See the "OpenLDAP Administrator's Guide" for more details on **slapd**.

OPTIONS

−4 Listen on IPv4 addresses only.

−6 Listen on IPv6 addresses only.

−T {a|c|d|i|p|t|acl|auth}
Run in Tool mode. The additional argument selects whether to run as slapadd, slapcat, slapdn, slapindex, slappasswd, or slaptest (slapacl and slapauth need the entire "*acl*" and "*auth*" option value to be spelled out, as "*a*" is reserved to **slapadd**). This option should be the first option specified when it is used; any remaining options will be interpreted by the corresponding slap tool program, according to the respective man pages. Note that these tool programs will usually be symbolic links to slapd. This option is provided for situations where symbolic links are not provided or not usable.

−d *debug−level*
Turn on debugging as defined by *debug−level*. If this option is specified, even with a zero argument, **slapd** will not fork or disassociate from the invoking terminal. Some general operation and status messages are printed for any value of *debug−level*. *debug−level* is taken as a bit string, with each bit corresponding to a different kind of debugging information. See <ldap_log.h> for details. Comma-separated arrays of friendly names can be specified to select debugging output of the corresponding debugging information. All the names recognized by the *loglevel* directive described in **slapd.conf**(5) are supported. If *debug−level* is **?**, a list of installed levels is printed, and slapd exits.

Remember that if you turn on packet logging, packets containing bind passwords will be output, so if you redirect the log to a logfile, that file should be read-protected.

−s *syslog−level*
This option tells **slapd** at what level debugging statements should be logged to the **syslog**(8) facility. The value "syslog−level" can be set to any value or combination allowed by the "-d" switch. Slapd logs all messages selected by "syslog−level" at the syslog(3) severity level "DEBUG", on the unit specified with "-l".

−n *service−name*
Specifies the service name for logging and other purposes. Defaults to basename of argv[0], i.e.: "slapd".

−l *syslog−local−user*
Selects the local user of the **syslog**(8) facility. Value can be **LOCAL0**, through **LOCAL7**, as well as **USER** and **DAEMON**. The default is **LOCAL4**. However, this option is only permitted on systems that support local users with the **syslog**(8) facility. Logging to syslog(8) occurs at the

"DEBUG" severity level.

−f *slapd−config−file*
Specifies the slapd configuration file. The default is **ETCDIR/slapd.conf**.

−F *slapd−config−directory*
Specifies the slapd configuration directory. The default is **ETCDIR/slapd.d**. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, slapd will attempt to read the default config directory before trying to use the default config file. If a valid config directory exists then the default config file is ignored. All of the slap tools that use the config options observe this same behavior.

−h *URLlist*
slapd will by default serve **ldap:///** (LDAP over TCP on all interfaces on default LDAP port). That is, it will bind using INADDR_ANY and port 389. The **−h** option may be used to specify LDAP (and other scheme) URLs to serve. For example, if slapd is given **−h "ldap://127.0.0.1:9009/ ldaps:/// ldapi:///"**, it will listen on 127.0.0.1:9009 for LDAP, 0.0.0.0:636 for LDAP over TLS, and LDAP over IPC (Unix domain sockets). Host 0.0.0.0 represents INADDR_ANY (any interface). A space separated list of URLs is expected. The URLs should be of the LDAP, LDAPS, or LDAPI schemes, and generally without a DN or other optional parameters (excepting as discussed below). Support for the latter two schemes depends on selected configuration options. Hosts may be specified by name or IPv4 and IPv6 address formats. Ports, if specified, must be numeric. The default ldap:// port is 389 and the default ldaps:// port is 636.

The listener permissions are indicated by "x-mod=-rwxrwxrwx", "x-mod=0777" or "x-mod=777", where any of the "rwx" can be "-" to suppress the related permission, while any of the "7" can be any legal octal digit, according to chmod(1). The listeners can take advantage of the "x-mod" extension to apply rough limitations to operations, e.g. allow read operations ("r", which applies to search and compare), write operations ("w", which applies to add, delete, modify and modrdn), and execute operations ("x", which means bind is required). "User" permissions apply to authenticated users, while "other" apply to anonymous users; "group" permissions are ignored. For example, "ldap:///????x-mod=-rw-------" means that read and write is only allowed for authenticated connections, and bind is required for all operations. This feature is experimental, and requires to be manually enabled at configure time.

−r *directory*
Specifies a directory to become the root directory. slapd will change the current working directory to this directory and then **chroot**(2) to this directory. This is done after opening listeners but before reading any configuration file or initializing any backend. When used as a security mechanism, it should be used in conjunction with **-u** and **-g** options.

−u *user* **slapd** will run slapd with the specified user name or id, and that user's supplementary group access list as set with initgroups(3). The group ID is also changed to this user's gid, unless the -g option is used to override. Note when used with **-r**, slapd will use the user database in the change root environment.

Note that on some systems, running as a non-privileged user will prevent passwd back-ends from accessing the encrypted passwords. Note also that any shell back-ends will run as the specified non-privileged user.

−g *group*
slapd will run with the specified group name or id. Note when used with **-r**, slapd will use the group database in the change root environment.

−c *cookie*
This option provides a cookie for the syncrepl replication consumer. The cookie is a comma separated list of name=value pairs. Currently supported syncrepl cookie fields are **rid** and **csn. rid** identifies a replication thread within the consumer server and is used to find the syncrepl

specification in **slapd.conf**(5) having the matching replication identifier in its definition. The **rid** must be provided in order for any other specified values to be used. **csn** is the commit sequence number received by a previous synchronization and represents the state of the consumer replica content which the syncrepl engine will synchronize to the current provider content.

−o *option[=value]*

This option provides a generic means to specify options without the need to reserve a separate letter for them.

It supports the following options:

slp={**on**|**off**|*slp−attrs*}

When SLP support is compiled into slapd, disable it (**off**), enable it by registering at SLP DAs without specific SLP attributes (**on**), or with specific SLP attributes *slp−attrs* that must be an SLP attribute list definition according to the SLP standard.

For example, "-o slp=(tree=production),(server-type=OpenLDAP),(server-version=2.3.20)" registers at SLP DAs with the three SLP attributes tree, server-type and server-version that have the values given above. This allows to specifically query the SLP DAs for LDAP servers holding the *production* tree in case multiple trees are available.

EXAMPLES

To start *slapd* and have it fork and detach from the terminal and start serving the LDAP databases defined in the default config file, just type:

```
LIBEXECDIR/slapd
```

To start **slapd** with an alternate configuration file, and turn on voluminous debugging which will be printed on standard error, type:

```
LIBEXECDIR/slapd -f /var/tmp/slapd.conf -d 255
```

To test whether the configuration file is correct or not, type:

```
LIBEXECDIR/slapd -Tt
```

SEE ALSO

ldap(3), **slapd.conf**(5), **slapd.access**(5), **slapacl**(8), **slapadd**(8), **slapauth**(8), **slapcat**(8), **slapdn**(8), **slapindex**(8), **slappasswd**(8), **slaptest**(8).

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

BUGS

See http://www.openldap.org/its/

ACKNOWLEDGEMENTS

NAME

slapd − Stand-alone LDAP Daemon

SYNOPSIS

/usr/libexec/slapd [−[4|6]] [−T {acl|add|auth|cat|dn|index|passwd|test}] [−d debug−level] [−f slapd−config−file] [−F slapd−config−directory] [−h URLs] [−n service−name] [−s syslog−level] [−l syslog−local−user] [−o option[=value]] [−r directory] [−u user] [−g group] [−c cookie]

DESCRIPTION

Slapd is the stand-alone LDAP daemon. It listens for LDAP connections on any number of ports (default 389), responding to the LDAP operations it receives over these connections. **slapd** is typically invoked at boot time, usually out of **/etc/rc.local**. Upon startup, **slapd** normally forks and disassociates itself from the invoking tty. If configured in the config file (or config directory), the **slapd** process will print its process ID (see **getpid**(2)) to a **.pid** file, as well as the command line options during invocation to an **.args** file (see **slapd.conf**(5)). If the **−d** flag is given, even with a zero argument, **slapd** will not fork and disassociate from the invoking tty.

See the "OpenLDAP Administrator's Guide" for more details on **slapd**.

OPTIONS

−4 Listen on IPv4 addresses only.

−6 Listen on IPv6 addresses only.

−T {a|c|d|i|p|t|acl|auth}

Run in Tool mode. The additional argument selects whether to run as slapadd, slapcat, slapdn, slapindex, slappasswd, or slaptest (slapacl and slapauth need the entire "*acl*" and "*auth*" option value to be spelled out, as "*a*" is reserved to **slapadd**). This option should be the first option specified when it is used; any remaining options will be interpreted by the corresponding slap tool program, according to the respective man pages. Note that these tool programs will usually be symbolic links to slapd. This option is provided for situations where symbolic links are not provided or not usable.

−d *debug−level*

Turn on debugging as defined by *debug−level*. If this option is specified, even with a zero argument, **slapd** will not fork or disassociate from the invoking terminal. Some general operation and status messages are printed for any value of *debug−level*. *debug−level* is taken as a bit string, with each bit corresponding to a different kind of debugging information. See <ldap_log.h> for details. Comma-separated arrays of friendly names can be specified to select debugging output of the corresponding debugging information. All the names recognized by the *loglevel* directive described in **slapd.conf**(5) are supported. If *debug−level* is **?**, a list of installed levels is printed, and slapd exits.

Remember that if you turn on packet logging, packets containing bind passwords will be output, so if you redirect the log to a logfile, that file should be read-protected.

−s *syslog−level*

This option tells **slapd** at what level debugging statements should be logged to the **syslog**(8) facility. The value "syslog−level" can be set to any value or combination allowed by the "-d" switch. Slapd logs all messages selected by "syslog−level" at the syslog(3) severity level "DEBUG", on the unit specified with "-l".

−n *service−name*

Specifies the service name for logging and other purposes. Defaults to basename of argv[0], i.e.: "slapd".

−l *syslog−local−user*

Selects the local user of the **syslog**(8) facility. Value can be **LOCAL0**, through **LOCAL7**, as well as **USER** and **DAEMON**. The default is **LOCAL4**. However, this option is only permitted on systems that support local users with the **syslog**(8) facility. Logging to syslog(8) occurs at the

"DEBUG" severity level.

−f *slapd−config−file*
Specifies the slapd configuration file. The default is **/etc/openldap/slapd.conf**.

−F *slapd−config−directory*
Specifies the slapd configuration directory. The default is **/etc/openldap/slapd.d**. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, slapd will attempt to read the default config directory before trying to use the default config file. If a valid config directory exists then the default config file is ignored. All of the slap tools that use the config options observe this same behavior.

−h *URLlist*
slapd will by default serve **ldap:///** (LDAP over TCP on all interfaces on default LDAP port). That is, it will bind using INADDR_ANY and port 389. The **−h** option may be used to specify LDAP (and other scheme) URLs to serve. For example, if slapd is given **−h "ldap://127.0.0.1:9009/ ldaps:/// ldapi:///"**, it will listen on 127.0.0.1:9009 for LDAP, 0.0.0.0:636 for LDAP over TLS, and LDAP over IPC (Unix domain sockets). Host 0.0.0.0 represents INADDR_ANY (any interface). A space separated list of URLs is expected. The URLs should be of the LDAP, LDAPS, or LDAPI schemes, and generally without a DN or other optional parameters (excepting as discussed below). Support for the latter two schemes depends on selected configuration options. Hosts may be specified by name or IPv4 and IPv6 address formats. Ports, if specified, must be numeric. The default ldap:// port is 389 and the default ldaps:// port is 636.

The listener permissions are indicated by "x-mod=-rwxrwxrwx", "x-mod=0777" or "x-mod=777", where any of the "rwx" can be "-" to suppress the related permission, while any of the "7" can be any legal octal digit, according to chmod(1). The listeners can take advantage of the "x-mod" extension to apply rough limitations to operations, e.g. allow read operations ("r", which applies to search and compare), write operations ("w", which applies to add, delete, modify and modrdn), and execute operations ("x", which means bind is required). "User" permissions apply to authenticated users, while "other" apply to anonymous users; "group" permissions are ignored. For example, "ldap:///????x-mod=-rw-------" means that read and write is only allowed for authenticated connections, and bind is required for all operations. This feature is experimental, and requires to be manually enabled at configure time.

−r *directory*
Specifies a directory to become the root directory. slapd will change the current working directory to this directory and then **chroot**(2) to this directory. This is done after opening listeners but before reading any configuration file or initializing any backend. When used as a security mechanism, it should be used in conjunction with **-u** and **-g** options.

−u *user* **slapd** will run slapd with the specified user name or id, and that user's supplementary group access list as set with initgroups(3). The group ID is also changed to this user's gid, unless the -g option is used to override. Note when used with **-r**, slapd will use the user database in the change root environment.

Note that on some systems, running as a non-privileged user will prevent passwd back-ends from accessing the encrypted passwords. Note also that any shell back-ends will run as the specified non-privileged user.

−g *group*
slapd will run with the specified group name or id. Note when used with **-r**, slapd will use the group database in the change root environment.

−c *cookie*
This option provides a cookie for the syncrepl replication consumer. The cookie is a comma separated list of name=value pairs. Currently supported syncrepl cookie fields are **rid** and **csn. rid** identifies a replication thread within the consumer server and is used to find the syncrepl

specification in **slapd.conf**(5) having the matching replication identifier in its definition. The **rid** must be provided in order for any other specified values to be used. **csn** is the commit sequence number received by a previous synchronization and represents the state of the consumer replica content which the syncrepl engine will synchronize to the current provider content.

−o *option[=value]*

This option provides a generic means to specify options without the need to reserve a separate letter for them.

It supports the following options:

slp={**on**|**off**|*slp−attrs*}

When SLP support is compiled into slapd, disable it (**off**), enable it by registering at SLP DAs without specific SLP attributes (**on**), or with specific SLP attributes *slp−attrs* that must be an SLP attribute list definition according to the SLP standard.

For example, "-o slp=(tree=production),(server-type=OpenLDAP),(server-version=2.3.20)" registers at SLP DAs with the three SLP attributes tree, server-type and server-version that have the values given above. This allows to specifically query the SLP DAs for LDAP servers holding the *production* tree in case multiple trees are available.

EXAMPLES

To start *slapd* and have it fork and detach from the terminal and start serving the LDAP databases defined in the default config file, just type:

```
/usr/libexec/slapd
```

To start **slapd** with an alternate configuration file, and turn on voluminous debugging which will be printed on standard error, type:

```
/usr/libexec/slapd -f /var/tmp/slapd.conf -d 255
```

To test whether the configuration file is correct or not, type:

```
/usr/libexec/slapd -Tt
```

SEE ALSO

ldap(3), **slapd.conf**(5), **slapd.access**(5), **slapacl**(8), **slapadd**(8), **slapauth**(8), **slapcat**(8), **slapdn**(8), **slapindex**(8), **slappasswd**(8), **slaptest**(8).

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

BUGS

See http://www.openldap.org/its/

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

slapdn − Check a list of string-represented LDAP DNs based on schema syntax

SYNOPSIS

SBINDIR/slapdn [−d level] [−f slapd.conf] [−F confdir] [−N | −P] [−o name[=value]] [−v] DN [...]

DESCRIPTION

Slapdn is used to check the conformance of a DN based on the schema defined in **slapd**(8) and that loaded via **slapd.conf**(5). It opens the **slapd.conf**(5) configuration file, reads in the schema definitions, and then parses the **DN** list given on the command-line.

OPTIONS

−d *level*
: enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
: specify an alternative **slapd.conf**(5) file.

−F *confdir*
: specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−N
: only output a normalized form of the DN, suitable to be used in a normalization tool; incompatible with **−P**.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see '−s' in slapd(8))
syslog-level=<level> (see '−S' in slapd(8))
syslog-user=<user>   (see '−l' in slapd(8))
```

−P
: only output a prettified form of the DN, suitable to be used in a check and beautification tool; incompatible with **−N**.

−v
: enable verbose mode.

EXAMPLES

To check a **DN** give the command:

```
SBINDIR/slapdn -f /ETCDIR/slapd.conf -v DN
```

SEE ALSO

ldap(3), **slapd**(8) **slaptest**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slapdn − Check a list of string-represented LDAP DNs based on schema syntax

SYNOPSIS

/usr/sbin/slapdn [−d level] [−f slapd.conf] [−F confdir] [−N | −P] [−o name[=value]] [−v] DN [...]

DESCRIPTION

Slapdn is used to check the conformance of a DN based on the schema defined in **slapd**(8) and that loaded via **slapd.conf**(5). It opens the **slapd.conf**(5) configuration file, reads in the schema definitions, and then parses the **DN** list given on the command-line.

OPTIONS

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−N only output a normalized form of the DN, suitable to be used in a normalization tool; incompatible with **−P**.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−P only output a prettified form of the DN, suitable to be used in a check and beautification tool; incompatible with **−N**.

−v enable verbose mode.

EXAMPLES

To check a **DN** give the command:

```
/usr/sbin/slapdn -f //etc/openldap/slapd.conf -v DN
```

SEE ALSO

ldap(3), **slapd**(8) **slaptest**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slapindex − Reindex entries in a SLAPD database

SYNOPSIS

SBINDIR/slapindex [−b suffix] [−c] [−d level] [−f slapd.conf] [−F confdir] [−g] [−n dbnum] [−o name[=value]] [−q] [−t] [−v] [attr] [...]

DESCRIPTION

Slapindex is used to regenerate **slapd**(8) indices based upon the current contents of a database. It opens the given database determined by the database number or suffix and updates the indices for all values of all attributes of all entries. If a list of specific attributes is provided on the command line, only the indices for those attributes will be regenerated. Databases configured as **subordinate** of this one are also re-indexed, unless **-g** is specified.

All files eventually created by **slapindex** will belong to the identity **slapindex** is run as, so make sure you either run **slapindex** with the same identity **slapd**(8) will be run as (see option **−u** in **slapd**(8)), or change file ownership before running **slapd**(8).

OPTIONS

−b *suffix*
Use the specified *suffix* to determine which database to generate output for. The −b cannot be used in conjunction with the **−n** option.

−c enable continue (ignore errors) mode.

−d *level*
enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
specify an alternative **slapd.conf**(5) file.

−F *confdir*
specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−g disable subordinate gluing. Only the specified database will be processed, and not its glued subordinates (if any).

−n *dbnum*
Generate output for the *dbnum*−th database listed in the configuration file. The **−n** cannot be used in conjunction with the **−b** option.

−o *option[=value]*
Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−q enable quick (fewer integrity checks) mode. Performs no consistency checks when writing the database. Improves indexing time, **however** the database will most likely be unusable if any errors or interruptions occur.

−t enable truncate mode. Truncates (empties) an index database before indexing any entries. May only be used with Quick mode.

−v enable verbose mode.

LIMITATIONS

Your **slapd**(8) should not be running (at least, not in read-write mode) when you do this to ensure consistency of the database.

This command provides ample opportunity for the user to obtain and drink their favorite beverage.

EXAMPLES

To reindex your SLAPD database, give the command:

```
SBINDIR/slapindex
```

To regenerate the index for only a specific attribute, e.g. "uid", give the command:

```
SBINDIR/slapindex uid
```

SEE ALSO

ldap(3), **ldif**(5), **slapadd**(8), **ldapadd**(1), **slapd**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slapindex − Reindex entries in a SLAPD database

SYNOPSIS

/usr/sbin/slapindex [−b suffix] [−c] [−d level] [−f slapd.conf] [−F confdir] [−g] [−n dbnum] [−o name[=value]] [−q] [−t] [−v] [attr] [...]

DESCRIPTION

Slapindex is used to regenerate **slapd**(8) indices based upon the current contents of a database. It opens the given database determined by the database number or suffix and updates the indices for all values of all attributes of all entries. If a list of specific attributes is provided on the command line, only the indices for those attributes will be regenerated. Databases configured as **subordinate** of this one are also re-indexed, unless **-g** is specified.

All files eventually created by **slapindex** will belong to the identity **slapindex** is run as, so make sure you either run **slapindex** with the same identity **slapd**(8) will be run as (see option **−u** in **slapd**(8)), or change file ownership before running **slapd**(8).

OPTIONS

−b *suffix*
: Use the specified *suffix* to determine which database to generate output for. The −b cannot be used in conjunction with the **−n** option.

−c
: enable continue (ignore errors) mode.

−d *level*
: enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
: specify an alternative **slapd.conf**(5) file.

−F *confdir*
: specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, an attempt to read the default config directory will be made before trying to use the default config file. If a valid config directory exists then the default config file is ignored.

−g
: disable subordinate gluing. Only the specified database will be processed, and not its glued subordinates (if any).

−n *dbnum*
: Generate output for the *dbnum*−th database listed in the configuration file. The **−n** cannot be used in conjunction with the **−b** option.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−q
: enable quick (fewer integrity checks) mode. Performs no consistency checks when writing the database. Improves indexing time, **however** the database will most likely be unusable if any errors or interruptions occur.

−t
: enable truncate mode. Truncates (empties) an index database before indexing any entries. May only be used with Quick mode.

−v
: enable verbose mode.

LIMITATIONS

Your **slapd**(8) should not be running (at least, not in read-write mode) when you do this to ensure consistency of the database.

This command provides ample opportunity for the user to obtain and drink their favorite beverage.

EXAMPLES

To reindex your SLAPD database, give the command:

```
/usr/sbin/slapindex
```

To regenerate the index for only a specific attribute, e.g. "uid", give the command:

```
/usr/sbin/slapindex uid
```

SEE ALSO

ldap(3), **ldif**(5), **slapadd**(8), **ldapadd**(1), **slapd**(8)

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slappasswd − OpenLDAP password utility

SYNOPSIS

SBINDIR/slappasswd [−v] [−u] [−g|−s secret|−T file] [−h hash] [−c salt-format] [−n]

DESCRIPTION

Slappasswd is used to generate an userPassword value suitable for use with **ldapmodify**(1) or **slapd.conf**(5) *rootpw* configuration directive.

OPTIONS

−v enable verbose mode.

−u Generate RFC 2307 userPassword values (the default). Future versions of this program may generate alternative syntaxes by default. This option is provided for forward compatibility.

−s *secret*

The secret to hash. If this, **−g** and **−T** are absent, the user will be prompted for the secret to hash. **−s**, **−g** and **−T** and mutually exclusive flags.

−g Generate the secret. If this, **−s** and **−T** are absent, the user will be prompted for the secret to hash. **−s**, **−g** and **−T** and mutually exclusive flags. If this is present, *{CLEARTEXT}* is used as scheme. **−g** and **−h** are mutually exclusive flags.

−T *file* Hash the contents of the file. If this, **−g** and **−s** are absent, the user will be prompted for the secret to hash. **−s**, **−g** and **−T** and mutually exclusive flags.

−h *scheme*

If -h is specified, one of the following RFC 2307 schemes may be specified: *{CRYPT}*, *{MD5}*, *{SMD5}*, *{SSHA}*, and *{SHA}*. The default is *{SSHA}*.

Note that scheme names may need to be protected, due to **{** and **}**, from expansion by the user’s command interpreter.

{SHA} and **{SSHA}** use the SHA-1 algorithm (FIPS 160-1), the latter with a seed.

{MD5} and **{SMD5}** use the MD5 algorithm (RFC 1321), the latter with a seed.

{CRYPT} uses the **crypt**(3).

{CLEARTEXT} indicates that the new password should be added to userPassword as clear text. Unless *{CLEARTEXT}* is used, this flag is incompatible with **−g**.

−c *crypt-salt-format*

Specify the format of the salt passed to **crypt**(3) when generating {CRYPT} passwords. This string needs to be in **sprintf**(3) format and may include one (and only one) %s conversion. This conversion will be substituted with a string random characters from [A−Za−z0−9./]. For example, ’%.2s’ provides a two character salt and ’1%.8s’ tells some versions of crypt(3) to use an MD5 algorithm and provides 8 random characters of salt. The default is ’%s’, which provides 31 characters of salt.

−n Omit the trailing newline; useful to pipe the credentials into a command.

LIMITATIONS

The practice of storing hashed passwords in userPassword violates Standard Track (RFC 4519) schema specifications and may hinder interoperability. A new attribute type, authPassword, to hold hashed passwords has been defined (RFC 3112), but is not yet implemented in **slapd**(8).

It should also be noted that the behavior of **crypt**(3) is platform specific.

SECURITY CONSIDERATIONS

Use of hashed passwords does not protect passwords during protocol transfer. TLS or other eavesdropping protections should be in−place before using LDAP simple bind.

The hashed password values should be protected as if they were clear text passwords.

SEE ALSO

ldappasswd(1), **ldapmodify**(1), **slapd**(8) **slapd.conf**(5) **RFC 2307 RFC 4519 RFC 3112**

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slappasswd − OpenLDAP password utility

SYNOPSIS

/usr/sbin/slappasswd [−v] [−u] [−g|−s secret|−T file] [−h hash] [−c salt-format] [−n]

DESCRIPTION

Slappasswd is used to generate an userPassword value suitable for use with **ldapmodify**(1) or **slapd.conf**(5) *rootpw* configuration directive.

OPTIONS

−v enable verbose mode.

−u Generate RFC 2307 userPassword values (the default). Future versions of this program may generate alternative syntaxes by default. This option is provided for forward compatibility.

−s *secret*

The secret to hash. If this, **−g** and **−T** are absent, the user will be prompted for the secret to hash. **−s**, **−g** and **−T** and mutually exclusive flags.

−g Generate the secret. If this, **−s** and **−T** are absent, the user will be prompted for the secret to hash. **−s**, **−g** and **−T** and mutually exclusive flags. If this is present, *{CLEARTEXT}* is used as scheme. **−g** and **−h** are mutually exclusive flags.

−T *file* Hash the contents of the file. If this, **−g** and **−s** are absent, the user will be prompted for the secret to hash. **−s**, **−g** and **−T** and mutually exclusive flags.

−h *scheme*

If -h is specified, one of the following RFC 2307 schemes may be specified: *{CRYPT}*, *{MD5}*, *{SMD5}*, *{SSHA}*, and *{SHA}*. The default is *{SSHA}*.

Note that scheme names may need to be protected, due to **{** and **}**, from expansion by the user’s command interpreter.

{SHA} and **{SSHA}** use the SHA-1 algorithm (FIPS 160-1), the latter with a seed.

{MD5} and **{SMD5}** use the MD5 algorithm (RFC 1321), the latter with a seed.

{CRYPT} uses the **crypt**(3).

{CLEARTEXT} indicates that the new password should be added to userPassword as clear text. Unless *{CLEARTEXT}* is used, this flag is incompatible with **−g**.

−c *crypt-salt-format*

Specify the format of the salt passed to **crypt**(3) when generating {CRYPT} passwords. This string needs to be in **sprintf**(3) format and may include one (and only one) %s conversion. This conversion will be substituted with a string random characters from [A−Za−z0−9./]. For example, ’%.2s’ provides a two character salt and ’1%.8s’ tells some versions of crypt(3) to use an MD5 algorithm and provides 8 random characters of salt. The default is ’%s’, which provides 31 characters of salt.

−n Omit the trailing newline; useful to pipe the credentials into a command.

LIMITATIONS

The practice of storing hashed passwords in userPassword violates Standard Track (RFC 4519) schema specifications and may hinder interoperability. A new attribute type, authPassword, to hold hashed passwords has been defined (RFC 3112), but is not yet implemented in **slapd**(8).

It should also be noted that the behavior of **crypt**(3) is platform specific.

SECURITY CONSIDERATIONS

Use of hashed passwords does not protect passwords during protocol transfer. TLS or other eavesdropping protections should be in−place before using LDAP simple bind.

The hashed password values should be protected as if they were clear text passwords.

SEE ALSO

ldappasswd(1), **ldapmodify**(1), **slapd**(8) **slapd.conf**(5) **RFC 2307 RFC 4519 RFC 3112**

"OpenLDAP Administrator's Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slaptest − Check the suitability of the OpenLDAP slapd.conf file

SYNOPSIS

SBINDIR/slaptest [−d level] [−f slapd.conf] [−F confdir] [−o name[=value]] [−Q] [−u] [−v]

DESCRIPTION

Slaptest is used to check the conformance of the **slapd.conf**(5) configuration file. It opens the **slapd.conf**(5) configuration file, and parses it according to the general and the backend-specific rules, checking its sanity.

OPTIONS

−d *level*
: enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
: specify an alternative **slapd.conf**(5) file.

−F *confdir*
: specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, slaptest will attempt to read the default config directory before trying to use the default config file. If a valid config directory exists then the default config file is ignored. If dryrun mode is also specified, no conversion will occur.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−Q Be extremely quiet: only the exit code indicates success (0) or not (any other value).

−u enable dryrun mode (i.e. don’t fail if databases cannot be opened, but config is fine).

−v enable verbose mode.

EXAMPLES

To check a **slapd.conf**(5) give the command:

```
SBINDIR/slaptest -f /ETCDIR/slapd.conf -v
```

SEE ALSO

ldap(3), **slapd**(8) **slapdn**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

ACKNOWLEDGEMENTS

NAME

slaptest − Check the suitability of the OpenLDAP slapd.conf file

SYNOPSIS

/usr/sbin/slaptest [−d level] [−f slapd.conf] [−F confdir] [−o name[=value]] [−Q] [−u] [−v]

DESCRIPTION

Slaptest is used to check the conformance of the **slapd.conf**(5) configuration file. It opens the **slapd.conf**(5) configuration file, and parses it according to the general and the backend-specific rules, checking its sanity.

OPTIONS

−d *level*
: enable debugging messages as defined by the specified *level*; see **slapd**(8) for details.

−f *slapd.conf*
: specify an alternative **slapd.conf**(5) file.

−F *confdir*
: specify a config directory. If both **-f** and **-F** are specified, the config file will be read and converted to config directory format and written to the specified directory. If neither option is specified, slaptest will attempt to read the default config directory before trying to use the default config file. If a valid config directory exists then the default config file is ignored. If dryrun mode is also specified, no conversion will occur.

−o *option[=value]*
: Specify an **option** with a(n optional) **value**. Possible generic options/values are:

```
syslog=<subsystems>  (see ‘−s’ in slapd(8))
syslog-level=<level> (see ‘−S’ in slapd(8))
syslog-user=<user>   (see ‘−l’ in slapd(8))
```

−Q Be extremely quiet: only the exit code indicates success (0) or not (any other value).

−u enable dryrun mode (i.e. don’t fail if databases cannot be opened, but config is fine).

−v enable verbose mode.

EXAMPLES

To check a **slapd.conf**(5) give the command:

```
/usr/sbin/slaptest -f //etc/openldap/slapd.conf -v
```

SEE ALSO

ldap(3), **slapd**(8) **slapdn**(8)

"OpenLDAP Administrator’s Guide" (http://www.OpenLDAP.org/doc/admin/)

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

slattach — attach serial lines as network interfaces

SYNOPSIS

slattach [**-Hhlmn**] [**-s** *baudrate*] [**-t** *ldisc*] *ttyname*

DESCRIPTION

slattach is used to assign a tty line to a network interface which uses asynchronous serial lines.

Currently the **slattach** command is used to attach sl(4) or strip(4) interfaces. These interfaces have to be created using the ifconfig(8) **create** subcommand before the **slattach** command. The network source and destination addresses and other interface parameters are configured via ifconfig(8).

The following operands are supported by **slattach**:

-H Turn on DTR/CTS flow control. By default, no flow control is done.

-h Turn on RTS/CTS flow control. By default, no flow control is done.

-l Turn on the CLOCAL flag, making it possible to run SLIP on a cable without modem control signals (e.g. DTR, DSR, DCD).

-m Maintain modem control signals after closing the line. Specifically, this disables HUPCL.

-n Don't detach from invoking tty.

-s *baudrate*
Specifies the speed of the connection. If not specified, the default of 9600 is used.

-t *ldisc* Specifies the line discipline to use for the tty. Supported line disciplines are "slip" (creates a sl(4) instance) and "strip" (creates a strip(4) instance). If this option is not specified, the default is "slip".

ttyname Specifies the name of the tty device. *ttyname* should be a string of the form ttyXX, or /dev/ttyXX.

Only the super-user may attach a network interface.

To detach the interface, use "ifconfig interface-name down" after killing off the **slattach** process. *Interface-name* is the name that is shown by netstat(1).

EXAMPLES

```
slattach ttyh8
slattach −s 4800 /dev/tty01
```

DIAGNOSTICS

Messages indicating that the specified interface is not configured or created, the requested address is unknown, or that the user is not privileged but tried to alter an interface's configuration.

SEE ALSO

netstat(1), daemon(3), netintro(4), sl(4), strip(4), ifconfig(8), rc(8), sliplogin(8), slstats(8)

HISTORY

The **slattach** command appeared in 4.3BSD.

BUGS

There is no way to specify the interface name (sl%d etc.) to be attached by the **slattach** command. There is no way to see which interface is assigned to the specified tty by the **slattach** command, either.

It would be better if the network interfaces were created by the **slattach** command rather than by using the `ifconfig`(8) **create** subcommand before the **slattach** command.

NAME

`sliplogin` — attach a serial line network interface

SYNOPSIS

`sliplogin` [*`loginname`*]

DESCRIPTION

`sliplogin` is used to turn the terminal line on standard input into a Serial Line IP (SLIP) link to a remote host. To do this, the program searches the file `/etc/sliphome/slip.hosts` for an entry matching *`loginname`* (which defaults to the current login name if omitted). If a matching entry is found, the line is configured appropriately for slip (8-bit transparent i/o) and converted to SLIP line discipline. Then a shell script is invoked to initialize the slip interface with the appropriate local and remote IP address, netmask, etc.

The usual initialization script is `/etc/sliphome/slip.login` but, if particular hosts need special initialization, the file `/etc/sliphome/slip.login.`*`loginname`* will be executed instead if it exists. The script is invoked with the parameters

slipunit The unit number of the slip interface assigned to this line. E.g., **0** for **sl0**.

speed The speed of the line.

args The arguments from the `/etc/sliphome/slip.hosts` entry, in order starting with *`loginname`*.

Only the super-user may attach a network interface. The interface is automatically detached when the other end hangs up or the **`sliplogin`** process dies. If the kernel slip module has been configured for it, all routes through that interface will also disappear at the same time. If there is other processing a site would like done on hangup, the file `/etc/sliphome/slip.logout` or `/etc/sliphome/slip.logout.`*`loginname`* is executed if it exists. It is given the same arguments as the login script.

Format of /etc/sliphome/slip.hosts

Comments (lines starting with a '#') and blank lines are ignored. Other lines must start with a *`loginname`* but the remaining arguments can be whatever is appropriate for the `slip.login` file that will be executed for that name. Arguments are separated by white space and follow normal `sh`(1) quoting conventions (however, *`loginname`* cannot be quoted). Usually, lines have the form

```
loginname local-address remote-address netmask opt-args
```

where *local-address* and *remote-address* are the IP host names or addresses of the local and remote ends of the slip line and *netmask* is the appropriate IP netmask. These arguments are passed directly to `ifconfig`(8). *opt-args* are optional arguments used to configure the line.

EXAMPLES

The normal use of **`sliplogin`** is to create a `/etc/passwd` entry for each legal, remote slip site with **`sliplogin`** as the shell for that entry. E.g.,

```
Sfoo:ikhuy6:2010:1:slip line to foo:/tmp:/usr/sbin/sliplogin
```

(Our convention is to name the account used by remote host *`hostname`* as *Shostname*.) Then an entry is added to `slip.hosts` that looks like:

```
Sfoo   ‘hostname‘    foo    netmask
```

where *'hostname'* will be evaluated by `sh`(1) to the local host name and *netmask* is the local host IP netmask.

Note that **sliplogin** must be setuid to root and, while not a security hole, moral defectives can use it to place terminal lines in an unusable state and/or deny access to legitimate users of a remote slip line. To prevent this, a site can create a group, say *slip*, that only the slip login accounts are put in then make sure that `/usr/sbin/sliplogin` is in group *slip* and mode 4550 (setuid root, only group *slip* can execute binary).

DIAGNOSTICS

sliplogin logs various information to the system log daemon, `syslogd`(8), with a facility code of *daemon*. The messages are listed here, grouped by severity level.

Error Severity

ioctl (TCGETS): *reason*
: A `TCGETS` **ioctl**() to get the line parameters failed.

ioctl (TCSETS): *reason*
: A `TCSETS` **ioctl**() to set the line parameters failed.

/etc/sliphome/slip.hosts: *reason*
: The `/etc/sliphome/slip.hosts` file could not be opened.

access denied for *user*
: No entry for *user* was found in `/etc/sliphome/slip.hosts`.

Notice Severity

attaching slip unit *unit* **for** *`loginname`*
: SLIP unit *unit* was successfully attached.

SEE ALSO

`sl`(4), `slattach`(8), `syslogd`(8)

HISTORY

The **sliplogin** command is currently in beta test.

NAME

smtp − Postfix SMTP+LMTP client

SYNOPSIS

smtp [generic Postfix daemon options]

DESCRIPTION

The Postfix SMTP+LMTP client implements the SMTP and LMTP mail delivery protocols. It processes message delivery requests from the queue manager. Each request specifies a queue file, a sender address, a domain or host to deliver to, and recipient information. This program expects to be run from the **master**(8) process manager.

The SMTP+LMTP client updates the queue file and marks recipients as finished, or it informs the queue manager that delivery should be tried again at a later time. Delivery status reports are sent to the **bounce**(8), **defer**(8) or **trace**(8) daemon as appropriate.

The SMTP+LMTP client looks up a list of mail exchanger addresses for the destination host, sorts the list by preference, and connects to each listed address until it finds a server that responds.

When a server is not reachable, or when mail delivery fails due to a recoverable error condition, the SMTP+LMTP client will try to deliver the mail to an alternate host.

After a successful mail transaction, a connection may be saved to the **scache**(8) connection cache server, so that it may be used by any SMTP+LMTP client for a subsequent transaction.

By default, connection caching is enabled temporarily for destinations that have a high volume of mail in the active queue. Connection caching can be enabled permanently for specific destinations.

SMTP DESTINATION SYNTAX

SMTP destinations have the following form:

domainname

domainname:*port*

Look up the mail exchangers for the specified domain, and connect to the specified port (default: **smtp**).

[*hostname*]

[*hostname*]:*port*

Look up the address(es) of the specified host, and connect to the specified port (default: **smtp**).

[*address*]

[*address*]:*port*

Connect to the host at the specified address, and connect to the specified port (default: **smtp**). An IPv6 address must be formatted as [**ipv6**:*address*].

LMTP DESTINATION SYNTAX

LMTP destinations have the following form:

unix:*pathname*

Connect to the local UNIX-domain server that is bound to the specified *pathname*. If the process runs chrooted, an absolute pathname is interpreted relative to the Postfix queue directory.

inet:*hostname*

inet:*hostname*:*port*

inet:[*address*]

inet:[*address*]:*port*

Connect to the specified TCP port on the specified local or remote host. If no port is specified, connect to the port defined as **lmtp** in **services**(4). If no such service is found, the **lmtp_tcp_port** configuration parameter (default value of 24) will be used. An IPv6 address must be formatted as [**ipv6**:*address*].

SECURITY

The SMTP+LMTP client is moderately security-sensitive. It talks to SMTP or LMTP servers and to DNS servers on the network. The SMTP+LMTP client can be run chrooted at fixed low privilege.

STANDARDS

RFC 821 (SMTP protocol)
RFC 822 (ARPA Internet Text Messages)
RFC 1651 (SMTP service extensions)
RFC 1652 (8bit-MIME transport)
RFC 1870 (Message Size Declaration)
RFC 2033 (LMTP protocol)
RFC 2034 (SMTP Enhanced Error Codes)
RFC 2045 (MIME: Format of Internet Message Bodies)
RFC 2046 (MIME: Media Types)
RFC 2554 (AUTH command)
RFC 2821 (SMTP protocol)
RFC 2920 (SMTP Pipelining)
RFC 3207 (STARTTLS command)
RFC 3461 (SMTP DSN Extension)
RFC 3463 (Enhanced Status Codes)

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8). Corrupted message files are marked so that the queue manager can move them to the **corrupt** queue for further inspection.

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces, protocol problems, and of other trouble.

BUGS

SMTP and LMTP connection caching does not work with TLS. The necessary support for TLS object passivation and re-activation does not exist without closing the session, which defeats the purpose.

SMTP and LMTP connection caching assumes that SASL credentials are valid for all destinations that map onto the same IP address and TCP port.

CONFIGURATION PARAMETERS

Before Postfix version 2.3, the LMTP client is a separate program that implements only a subset of the functionality available with SMTP: there is no support for TLS, and connections are cached in-process, making it ineffective when the client is used for multiple domains.

Most smtp_*xxx* configuration parameters have an lmtp_*xxx* "mirror" parameter for the equivalent LMTP feature. This document describes only those LMTP-related parameters that aren't simply "mirror" parameters.

Changes to **main.cf** are picked up automatically, as **smtp**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

COMPATIBILITY CONTROLS

ignore_mx_lookup_error (no)
Ignore DNS MX lookups that produce no response.

smtp_always_send_ehlo (yes)
Always send EHLO at the start of an SMTP session.

smtp_never_send_ehlo (no)
Never send EHLO at the start of an SMTP session.

smtp_defer_if_no_mx_address_found (no)
Defer mail delivery when no MX record resolves to an IP address.

smtp_line_length_limit (990)
The maximal length of message header and body lines that Postfix will send via SMTP.

smtp_pix_workaround_delay_time (10s)
How long the Postfix SMTP client pauses before sending ".<CR><LF>" in order to work around the PIX firewall "<CR><LF>.<CR><LF>" bug.

smtp_pix_workaround_threshold_time (500s)
How long a message must be queued before the Postfix SMTP client turns on the PIX firewall "<CR><LF>.<CR><LF>" bug workaround for delivery through firewalls with "smtp fixup" mode turned on.

smtp_pix_workarounds (disable_esmtp, delay_dotcrlf)
A list that specifies zero or more workarounds for CISCO PIX firewall bugs.

smtp_pix_workaround_maps (empty)
Lookup tables, indexed by the remote SMTP server address, with per-destination workarounds for CISCO PIX firewall bugs.

smtp_quote_rfc821_envelope (yes)
Quote addresses in SMTP MAIL FROM and RCPT TO commands as required by RFC 821.

smtp_skip_5xx_greeting (yes)
Skip SMTP servers that greet with a 5XX status code (go away, do not try again later).

smtp_skip_quit_response (yes)
Do not wait for the response to the SMTP QUIT command.

Available in Postfix version 2.0 and earlier:

smtp_skip_4xx_greeting (yes)
Skip SMTP servers that greet with a 4XX status code (go away, try again later).

Available in Postfix version 2.2 and later:

smtp_discard_ehlo_keyword_address_maps (empty)
Lookup tables, indexed by the remote SMTP server address, with case insensitive lists of EHLO keywords (pipelining, starttls, auth, etc.) that the Postfix SMTP client will ignore in the EHLO response from a remote SMTP server.

smtp_discard_ehlo_keywords (empty)
A case insensitive list of EHLO keywords (pipelining, starttls, auth, etc.) that the Postfix SMTP client will ignore in the EHLO response from a remote SMTP server.

smtp_generic_maps (empty)
Optional lookup tables that perform address rewriting in the SMTP client, typically to transform a locally valid address into a globally valid address when sending mail across the Internet.

Available in Postfix version 2.2.9 and later:

smtp_cname_overrides_servername (version dependent)
Allow DNS CNAME records to override the servername that the Postfix SMTP client uses for logging, SASL password lookup, TLS policy decisions, or TLS certificate verification.

Available in Postfix version 2.3 and later:

lmtp_discard_lhlo_keyword_address_maps (empty)
Lookup tables, indexed by the remote LMTP server address, with case insensitive lists of LHLO keywords (pipelining, starttls, auth, etc.) that the LMTP client will ignore in the LHLO response from a remote LMTP server.

lmtp_discard_lhlo_keywords (empty)
A case insensitive list of LHLO keywords (pipelining, starttls, auth, etc.) that the LMTP client will ignore in the LHLO response from a remote LMTP server.

Available in Postfix version 2.4.4 and later:

send_cyrus_sasl_authzid (no)
When authenticating to a remote SMTP or LMTP server with the default setting "no", send no SASL authoriZation ID (authzid); send only the SASL authentiCation ID (authcid) plus the authcid's password.

MIME PROCESSING CONTROLS

Available in Postfix version 2.0 and later:

disable_mime_output_conversion (no)
Disable the conversion of 8BITMIME format to 7BIT format.

mime_boundary_length_limit (2048)
The maximal length of MIME multipart boundary strings.

mime_nesting_limit (100)
The maximal recursion level that the MIME processor will handle.

EXTERNAL CONTENT INSPECTION CONTROLS

Available in Postfix version 2.1 and later:

smtp_send_xforward_command (no)
Send the non-standard XFORWARD command when the Postfix SMTP server EHLO response announces XFORWARD support.

SASL AUTHENTICATION CONTROLS

smtp_sasl_auth_enable (no)
Enable SASL authentication in the Postfix SMTP client.

smtp_sasl_password_maps (empty)
Optional SMTP client lookup tables with one username:password entry per remote hostname or domain, or sender address when sender-dependent authentication is enabled.

smtp_sasl_security_options (noplaintext, noanonymous)
SASL security options; as of Postfix 2.3 the list of available features depends on the SASL client implementation that is selected with **smtp_sasl_type**.

Available in Postfix version 2.2 and later:

smtp_sasl_mechanism_filter (empty)
If non-empty, a Postfix SMTP client filter for the remote SMTP server's list of offered SASL mechanisms.

Available in Postfix version 2.3 and later:

smtp_sender_dependent_authentication (no)
Enable sender-dependent authentication in the Postfix SMTP client; this is available only with SASL authentication, and disables SMTP connection caching to ensure that mail from different senders will use the appropriate credentials.

smtp_sasl_path (empty)
Implementation-specific information that is passed through to the SASL plug-in implementation that is selected with **smtp_sasl_type**.

smtp_sasl_type (cyrus)
The SASL plug-in type that the Postfix SMTP client should use for authentication.

STARTTLS SUPPORT CONTROLS

Detailed information about STARTTLS configuration may be found in the TLS_README document.

smtp_tls_security_level (empty)
The default SMTP TLS security level for the Postfix SMTP client; when a non-empty value is specified, this overrides the obsolete parameters smtp_use_tls, smtp_enforce_tls, and smtp_tls_enforce_peername.

smtp_sasl_tls_security_options ($smtp_sasl_security_options)
The SASL authentication security options that the Postfix SMTP client uses for TLS encrypted SMTP sessions.

smtp_starttls_timeout (300s)
Time limit for Postfix SMTP client write and read operations during TLS startup and shutdown handshake procedures.

smtp_tls_CAfile (empty)
The file with the certificate of the certification authority (CA) that issued the Postfix SMTP client certificate.

smtp_tls_CApath (empty)
Directory with PEM format certificate authority certificates that the Postfix SMTP client uses to verify a remote SMTP server certificate.

smtp_tls_cert_file (empty)
File with the Postfix SMTP client RSA certificate in PEM format.

smtp_tls_mandatory_ciphers (medium)
The minimum TLS cipher grade that the Postfix SMTP client will use with mandatory TLS encryption.

smtp_tls_exclude_ciphers (empty)
List of ciphers or cipher types to exclude from the Postfix SMTP client cipher list at all TLS security levels.

smtp_tls_mandatory_exclude_ciphers (empty)
Additional list of ciphers or cipher types to exclude from the SMTP client cipher list at mandatory TLS security levels.

smtp_tls_dcert_file (empty)
File with the Postfix SMTP client DSA certificate in PEM format.

smtp_tls_dkey_file ($smtp_tls_dcert_file)
File with the Postfix SMTP client DSA private key in PEM format.

smtp_tls_key_file ($smtp_tls_cert_file)
File with the Postfix SMTP client RSA private key in PEM format.

smtp_tls_loglevel (0)
Enable additional Postfix SMTP client logging of TLS activity.

smtp_tls_note_starttls_offer (no)
Log the hostname of a remote SMTP server that offers STARTTLS, when TLS is not already enabled for that server.

smtp_tls_policy_maps (empty)
Optional lookup tables with the Postfix SMTP client TLS security policy by next-hop destination; when a non-empty value is specified, this overrides the obsolete smtp_tls_per_site parameter.

smtp_tls_mandatory_protocols (SSLv3, TLSv1)
List of TLS protocols that the Postfix SMTP client will use with mandatory TLS encryption.

smtp_tls_scert_verifydepth (5)
The verification depth for remote SMTP server certificates.

smtp_tls_secure_cert_match (nexthop, dot-nexthop)
The server certificate peername verification method for the "secure" TLS security level.

smtp_tls_session_cache_database (empty)
Name of the file containing the optional Postfix SMTP client TLS session cache.

smtp_tls_session_cache_timeout (3600s)
The expiration time of Postfix SMTP client TLS session cache information.

smtp_tls_verify_cert_match (hostname)
The server certificate peername verification method for the "verify" TLS security level.

tls_daemon_random_bytes (32)
The number of pseudo-random bytes that an **smtp**(8) or **smtpd**(8) process requests from the **tlsmgr**(8) server in order to seed its internal pseudo random number generator (PRNG).

tls_high_cipherlist (ALL:!EXPORT:!LOW:!MEDIUM:+RC4:@STRENGTH)
The OpenSSL cipherlist for "HIGH" grade ciphers.

tls_medium_cipherlist (ALL:!EXPORT:!LOW:+RC4:@STRENGTH)
The OpenSSL cipherlist for "MEDIUM" or higher grade ciphers.

tls_low_cipherlist (ALL:!EXPORT:+RC4:@STRENGTH)
The OpenSSL cipherlist for "LOW" or higher grade ciphers.

tls_export_cipherlist (ALL:+RC4:@STRENGTH)
The OpenSSL cipherlist for "EXPORT" or higher grade ciphers.

tls_null_cipherlist (eNULL:!aNULL)
The OpenSSL cipherlist for "NULL" grade ciphers that provide authentication without encryption.

Available in Postfix version 2.4 and later:

smtp_sasl_tls_verified_security_options ($smtp_sasl_tls_security_options)
The SASL authentication security options that the Postfix SMTP client uses for TLS encrypted SMTP sessions with a verified server certificate.

OBSOLETE STARTTLS CONTROLS

The following configuration parameters exist for compatibility with Postfix versions before 2.3. Support for these will be removed in a future release.

smtp_use_tls (no)
Opportunistic mode: use TLS when a remote SMTP server announces STARTTLS support, otherwise send the mail in the clear.

smtp_enforce_tls (no)
Enforcement mode: require that remote SMTP servers use TLS encryption, and never send mail in the clear.

smtp_tls_enforce_peername (yes)
With mandatory TLS encryption, require that the remote SMTP server hostname matches the information in the remote SMTP server certificate.

smtp_tls_per_site (empty)
Optional lookup tables with the Postfix SMTP client TLS usage policy by next-hop destination and by remote SMTP server hostname.

smtp_tls_cipherlist (empty)
Obsolete Postfix < 2.3 control for the Postfix SMTP client TLS cipher list.

RESOURCE AND RATE CONTROLS

smtp_destination_concurrency_limit ($default_destination_concurrency_limit)
The maximal number of parallel deliveries to the same destination via the smtp message delivery transport.

smtp_destination_recipient_limit ($default_destination_recipient_limit)
The maximal number of recipients per delivery via the smtp message delivery transport.

smtp_connect_timeout (30s)
The SMTP client time limit for completing a TCP connection, or zero (use the operating system built-in time limit).

smtp_helo_timeout (300s)
The SMTP client time limit for sending the HELO or EHLO command, and for receiving the initial server response.

lmtp_lhlo_timeout (300s)
The LMTP client time limit for sending the LHLO command, and for receiving the initial server response.

smtp_xforward_timeout (300s)
The SMTP client time limit for sending the XFORWARD command, and for receiving the server response.

smtp_mail_timeout (300s)
The SMTP client time limit for sending the MAIL FROM command, and for receiving the server response.

smtp_rcpt_timeout (300s)
The SMTP client time limit for sending the SMTP RCPT TO command, and for receiving the server response.

smtp_data_init_timeout (120s)
The SMTP client time limit for sending the SMTP DATA command, and for receiving the server response.

smtp_data_xfer_timeout (180s)
The SMTP client time limit for sending the SMTP message content.

smtp_data_done_timeout (600s)
The SMTP client time limit for sending the SMTP ".", and for receiving the server response.

smtp_quit_timeout (300s)
The SMTP client time limit for sending the QUIT command, and for receiving the server response.

Available in Postfix version 2.1 and later:

smtp_mx_address_limit (5)
The maximal number of MX (mail exchanger) IP addresses that can result from mail exchanger lookups, or zero (no limit).

smtp_mx_session_limit (2)
The maximal number of SMTP sessions per delivery request before giving up or delivering to a fall-back relay host, or zero (no limit).

smtp_rset_timeout (20s)
The SMTP client time limit for sending the RSET command, and for receiving the server response.

Available in Postfix version 2.2 and earlier:

lmtp_cache_connection (yes)
Keep Postfix LMTP client connections open for up to $max_idle seconds.

Available in Postfix version 2.2 and later:

smtp_connection_cache_destinations (empty)
Permanently enable SMTP connection caching for the specified destinations.

smtp_connection_cache_on_demand (yes)
Temporarily enable SMTP connection caching while a destination has a high volume of mail in the active queue.

smtp_connection_reuse_time_limit (300s)
The amount of time during which Postfix will use an SMTP connection repeatedly.

smtp_connection_cache_time_limit (2s)
When SMTP connection caching is enabled, the amount of time that an unused SMTP client socket is kept open before it is closed.

Available in Postfix version 2.3 and later:

connection_cache_protocol_timeout (5s)
Time limit for connection cache connect, send or receive operations.

TROUBLE SHOOTING CONTROLS

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.

error_notice_recipient (postmaster)
The recipient of postmaster notifications about mail delivery problems that are caused by policy, resource, software or protocol errors.

internal_mail_filter_classes (empty)
What categories of Postfix-generated mail are subject to before-queue content inspection by non_smtpd_milters, header_checks and body_checks.

notify_classes (resource, software)
The list of error classes that are reported to the postmaster.

MISCELLANEOUS CONTROLS

best_mx_transport (empty)
Where the Postfix SMTP client should deliver mail when it detects a "mail loops back to myself" error condition.

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

disable_dns_lookups (no)
Disable DNS lookups in the Postfix SMTP and LMTP clients.

inet_interfaces (all)
The network interface addresses that this mail system receives mail on.

inet_protocols (ipv4)
The Internet protocols Postfix will attempt to use when making or accepting connections.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

lmtp_tcp_port (24)
The default TCP port that the Postfix LMTP client connects to.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

proxy_interfaces (empty)
The network interface addresses that this mail system receives mail on by way of a proxy or network address translation unit.

smtp_bind_address (empty)
An optional numerical network address that the Postfix SMTP client should bind to when making an IPv4 connection.

smtp_bind_address6 (empty)
An optional numerical network address that the Postfix SMTP client should bind to when making an IPv6 connection.

smtp_helo_name ($myhostname)
The hostname to send in the SMTP EHLO or HELO command.

lmtp_lhlo_name ($myhostname)
The hostname to send in the LMTP LHLO command.

smtp_host_lookup (dns)
What mechanisms when the Postfix SMTP client uses to look up a host's IP address.

smtp_randomize_addresses (yes)
Randomize the order of equal-preference MX host 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".

Available with Postfix 2.2 and earlier:

fallback_relay (empty)
Optional list of relay hosts for SMTP destinations that can't be found or that are unreachable.

Available with Postfix 2.3 and later:

smtp_fallback_relay ($fallback_relay)
Optional list of relay hosts for SMTP destinations that can't be found or that are unreachable.

SEE ALSO

qmgr(8), queue manager
bounce(8), delivery status reports
scache(8), connection cache server
postconf(5), configuration parameters

master(5), generic daemon options
master(8), process manager
tlsmgr(8), TLS session and PRNG management
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
SASL_README, Postfix SASL howto
TLS_README, Postfix STARTTLS 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

Command pipelining in cooperation with:
Jon Ribbens
Oaktree Internet Solutions Ltd.,
Internet House,
Canal Basin,
Coventry,
CV1 4LY, United Kingdom.

SASL support originally by:
Till Franke
SuSE Rhein/Main AG
65760 Eschborn, Germany

Connection caching in cooperation with:
Victor Duchovni
Morgan Stanley

TLS support originally by:
Lutz Jaenicke
BTU Cottbus
Allgemeine Elektrotechnik
Universitaetsplatz 3-4
D-03044 Cottbus, Germany

NAME

smtpd − Postfix SMTP server

SYNOPSIS

smtpd [generic Postfix daemon options]

DESCRIPTION

The SMTP server accepts network connection requests and performs zero or more SMTP transactions per connection. Each received message is piped through the **cleanup**(8) daemon, and is placed into the **incoming** queue as one single queue file. For this mode of operation, the program expects to be run from the **master**(8) process manager.

Alternatively, the SMTP server be can run in stand-alone mode; this is traditionally obtained with "**sendmail -bs**". When the SMTP server runs stand-alone with non $**mail_owner** privileges, it receives mail even while the mail system is not running, deposits messages directly into the **maildrop** queue, and disables the SMTP server’s access policies. As of Postfix version 2.3, the SMTP server refuses to receive mail from the network when it runs with non $**mail_owner** privileges.

The SMTP server implements a variety of policies for connection requests, and for parameters given to **HELO, ETRN, MAIL FROM, VRFY** and **RCPT TO** commands. They are detailed below and in the **main.cf** configuration file.

SECURITY

The SMTP server is moderately security-sensitive. It talks to SMTP clients and to DNS servers on the network. The SMTP server can be run chrooted at fixed low privilege.

STANDARDS

RFC 821 (SMTP protocol)
RFC 1123 (Host requirements)
RFC 1652 (8bit-MIME transport)
RFC 1869 (SMTP service extensions)
RFC 1870 (Message Size Declaration)
RFC 1985 (ETRN command)
RFC 2034 (SMTP Enhanced Error Codes)
RFC 2554 (AUTH command)
RFC 2821 (SMTP protocol)
RFC 2920 (SMTP Pipelining)
RFC 3207 (STARTTLS command)
RFC 3461 (SMTP DSN Extension)
RFC 3463 (Enhanced Status Codes)

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces, protocol problems, policy violations, and of other trouble.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically, as **smtpd**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

COMPATIBILITY CONTROLS

The following parameters work around implementation errors in other software, and/or allow you to override standards in order to prevent undesirable use.

broken_sasl_auth_clients (no)
Enable inter-operability with SMTP clients that implement an obsolete version of the AUTH command (RFC 2554).

disable_vrfy_command (no)
Disable the SMTP VRFY command.

smtpd_noop_commands (empty)
List of commands that the Postfix SMTP server replies to with "250 Ok", without doing any syntax checks and without changing state.

strict_rfc821_envelopes (no)
Require that addresses received in SMTP MAIL FROM and RCPT TO commands are enclosed with <>, and that those addresses do not contain RFC 822 style comments or phrases.

Available in Postfix version 2.1 and later:

resolve_null_domain (no)
Resolve an address that ends in the "@" null domain as if the local hostname were specified, instead of rejecting the address as invalid.

smtpd_reject_unlisted_sender (no)
Request that the Postfix SMTP server rejects mail from unknown sender addresses, even when no explicit reject_unlisted_sender access restriction is specified.

smtpd_sasl_exceptions_networks (empty)
What SMTP clients Postfix will not offer AUTH support to.

Available in Postfix version 2.2 and later:

smtpd_discard_ehlo_keyword_address_maps (empty)
Lookup tables, indexed by the remote SMTP client address, with case insensitive lists of EHLO keywords (pipelining, starttls, auth, etc.) that the SMTP server will not send in the EHLO response to a remote SMTP client.

smtpd_discard_ehlo_keywords (empty)
A case insensitive list of EHLO keywords (pipelining, starttls, auth, etc.) that the SMTP server will not send in the EHLO response to a remote SMTP client.

smtpd_delay_open_until_valid_rcpt (yes)
Postpone the start of an SMTP mail transaction until a valid RCPT TO command is received.

Available in Postfix version 2.3 and later:

smtpd_tls_always_issue_session_ids (yes)
Force the Postfix SMTP server to issue a TLS session id, even when TLS session caching is turned off (smtpd_tls_session_cache_database is empty).

ADDRESS REWRITING CONTROLS

See the ADDRESS_REWRITING_README document for a detailed discussion of Postfix address rewriting.

receive_override_options (empty)
Enable or disable recipient validation, built-in content filtering, or address mapping.

Available in Postfix version 2.2 and later:

local_header_rewrite_clients (permit_inet_interfaces)
Rewrite message header addresses in mail from these clients and update incomplete addresses with the domain name in $myorigin or $mydomain; either don’t rewrite message headers from other clients at all, or rewrite message headers and update incomplete addresses with the domain specified in the remote_header_rewrite_domain parameter.

AFTER QUEUE EXTERNAL CONTENT INSPECTION CONTROLS

As of version 1.0, Postfix can be configured to send new mail to an external content filter AFTER the mail is queued. This content filter is expected to inject mail back into a (Postfix or other) MTA for further delivery. See the FILTER_README document for details.

content_filter (empty)
The name of a mail delivery transport that filters mail after it is queued.

BEFORE QUEUE EXTERNAL CONTENT INSPECTION CONTROLS

As of version 2.1, the Postfix SMTP server can be configured to send incoming mail to a real-time SMTP-based content filter BEFORE mail is queued. This content filter is expected to inject mail back into Postfix. See the SMTPD_PROXY_README document for details on how to configure and operate this feature.

smtpd_proxy_filter (empty)
The hostname and TCP port of the mail filtering proxy server.

smtpd_proxy_ehlo ($myhostname)
How the Postfix SMTP server announces itself to the proxy filter.

smtpd_proxy_timeout (100s)
The time limit for connecting to a proxy filter and for sending or receiving information.

BEFORE QUEUE MILTER CONTROLS

As of version 2.3, Postfix supports the Sendmail version 8 Milter (mail filter) protocol. These content filters run outside Postfix. They can inspect the SMTP command stream and the message content, and can request modifications before mail is queued. For details see the MILTER_README document.

smtpd_milters (empty)
A list of Milter (mail filter) applications for new mail that arrives via the Postfix **smtpd**(8) server.

milter_protocol (2)
The mail filter protocol version and optional protocol extensions for communication with a Milter (mail filter) application.

milter_default_action (tempfail)
The default action when a Milter (mail filter) application is unavailable or mis-configured.

milter_macro_daemon_name ($myhostname)
The {daemon_name} macro value for Milter (mail filter) applications.

milter_macro_v ($mail_name $mail_version)
The {v} macro value for Milter (mail filter) applications.

milter_connect_timeout (30s)
The time limit for connecting to a Milter (mail filter) application, and for negotiating protocol options.

milter_command_timeout (30s)
The time limit for sending an SMTP command to a Milter (mail filter) application, and for receiving the response.

milter_content_timeout (300s)
The time limit for sending message content to a Milter (mail filter) application, and for receiving the response.

milter_connect_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after completion of an SMTP connection.

milter_helo_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the SMTP HELO or EHLO command.

milter_mail_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the SMTP MAIL FROM command.

milter_rcpt_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the SMTP RCPT TO command.

milter_data_macros (see postconf -n output)
The macros that are sent to version 4 or higher Milter (mail filter) applications after the SMTP DATA command.

milter_unknown_command_macros (see postconf -n output)
The macros that are sent to version 3 or higher Milter (mail filter) applications after an unknown SMTP command.

milter_end_of_data_macros (see postconf -n output)
The macros that are sent to Milter (mail filter) applications after the message end-of-data.

GENERAL CONTENT INSPECTION CONTROLS

The following parameters are applicable for both built-in and external content filters.

Available in Postfix version 2.1 and later:

receive_override_options (empty)
Enable or disable recipient validation, built-in content filtering, or address mapping.

EXTERNAL CONTENT INSPECTION CONTROLS

The following parameters are applicable for both before-queue and after-queue content filtering.

Available in Postfix version 2.1 and later:

smtpd_authorized_xforward_hosts (empty)
What SMTP clients are allowed to use the XFORWARD feature.

SASL AUTHENTICATION CONTROLS

Postfix SASL support (RFC 2554) can be used to authenticate remote SMTP clients to the Postfix SMTP server, and to authenticate the Postfix SMTP client to a remote SMTP server. See the SASL_README document for details.

broken_sasl_auth_clients (no)
Enable inter-operability with SMTP clients that implement an obsolete version of the AUTH command (RFC 2554).

smtpd_sasl_auth_enable (no)
Enable SASL authentication in the Postfix SMTP server.

smtpd_sasl_local_domain (empty)
The name of the local SASL authentication realm.

smtpd_sasl_security_options (noanonymous)
SASL security options; as of Postfix 2.3 the list of available features depends on the SASL server implementation that is selected with **smtpd_sasl_type**.

smtpd_sender_login_maps (empty)
Optional lookup table with the SASL login names that own sender (MAIL FROM) addresses.

Available in Postfix version 2.1 and later:

smtpd_sasl_exceptions_networks (empty)
What SMTP clients Postfix will not offer AUTH support to.

Available in Postfix version 2.3 and later:

smtpd_sasl_authenticated_header (no)
Report the SASL authenticated user name in the **smtpd**(8) Received message header.

smtpd_sasl_path (smtpd)
Implementation-specific information that is passed through to the SASL plug-in implementation that is selected with **smtpd_sasl_type**.

smtpd_sasl_type (cyrus)
The SASL plug-in type that the Postfix SMTP server should use for authentication.

STARTTLS SUPPORT CONTROLS

Detailed information about STARTTLS configuration may be found in the TLS_README document.

smtpd_tls_security_level (empty)
The SMTP TLS security level for the Postfix SMTP server; when a non-empty value is specified, this overrides the obsolete parameters smtpd_use_tls and smtpd_enforce_tls.

smtpd_sasl_tls_security_options ($smtpd_sasl_security_options)
The SASL authentication security options that the Postfix SMTP server uses for TLS encrypted SMTP sessions.

smtpd_starttls_timeout (300s)
The time limit for Postfix SMTP server write and read operations during TLS startup and shutdown handshake procedures.

smtpd_tls_CAfile (empty)
The file with the certificate of the certification authority (CA) that issued the Postfix SMTP server certificate.

smtpd_tls_CAfile (empty)
The file with the certificate of the certification authority (CA) that issued the Postfix SMTP server certificate.

smtpd_tls_always_issue_session_ids (yes)
Force the Postfix SMTP server to issue a TLS session id, even when TLS session caching is turned off (smtpd_tls_session_cache_database is empty).

smtpd_tls_ask_ccert (no)
Ask a remote SMTP client for a client certificate.

smtpd_tls_auth_only (no)
When TLS encryption is optional in the Postfix SMTP server, do not announce or accept SASL authentication over unencrypted connections.

smtpd_tls_ccert_verifydepth (5)
The verification depth for remote SMTP client certificates.

smtpd_tls_cert_file (empty)
File with the Postfix SMTP server RSA certificate in PEM format.

smtpd_tls_exclude_ciphers (empty)
List of ciphers or cipher types to exclude from the SMTP server cipher list at all TLS security levels.

smtpd_tls_dcert_file (empty)
File with the Postfix SMTP server DSA certificate in PEM format.

smtpd_tls_dh1024_param_file (empty)
File with DH parameters that the Postfix SMTP server should use with EDH ciphers.

smtpd_tls_dh512_param_file (empty)
File with DH parameters that the Postfix SMTP server should use with EDH ciphers.

smtpd_tls_dkey_file ($smtpd_tls_dcert_file)
File with the Postfix SMTP server DSA private key in PEM format.

smtpd_tls_key_file ($smtpd_tls_cert_file)
File with the Postfix SMTP server RSA private key in PEM format.

smtpd_tls_loglevel (0)
Enable additional Postfix SMTP server logging of TLS activity.

smtpd_tls_mandatory_ciphers (medium)
The minimum TLS cipher grade that the Postfix SMTP server will use with mandatory TLS encryption.

smtpd_tls_mandatory_exclude_ciphers (empty)
Additional list of ciphers or cipher types to exclude from the SMTP server cipher list at mandatory TLS security levels.

smtpd_tls_mandatory_protocols (SSLv3, TLSv1)
The TLS protocols accepted by the Postfix SMTP server with mandatory TLS encryption.

smtpd_tls_received_header (no)
Request that the Postfix SMTP server produces Received: message headers that include information about the protocol and cipher used, as well as the client CommonName and client certificate issuer CommonName.

smtpd_tls_req_ccert (no)
With mandatory TLS encryption, require a remote SMTP client certificate in order to allow TLS connections to proceed.

smtpd_tls_session_cache_database (empty)
Name of the file containing the optional Postfix SMTP server TLS session cache.

smtpd_tls_session_cache_timeout (3600s)
The expiration time of Postfix SMTP server TLS session cache information.

smtpd_tls_wrappermode (no)
Run the Postfix SMTP server in the non-standard "wrapper" mode, instead of using the STARTTLS command.

tls_daemon_random_bytes (32)
The number of pseudo-random bytes that an **smtp**(8) or **smtpd**(8) process requests from the **tlsmgr**(8) server in order to seed its internal pseudo random number generator (PRNG).

tls_high_cipherlist (ALL:!EXPORT:!LOW:!MEDIUM:+RC4:@STRENGTH)
The OpenSSL cipherlist for "HIGH" grade ciphers.

tls_medium_cipherlist (ALL:!EXPORT:!LOW:+RC4:@STRENGTH)
The OpenSSL cipherlist for "MEDIUM" or higher grade ciphers.

tls_low_cipherlist (ALL:!EXPORT:+RC4:@STRENGTH)
The OpenSSL cipherlist for "LOW" or higher grade ciphers.

tls_export_cipherlist (ALL:+RC4:@STRENGTH)
The OpenSSL cipherlist for "EXPORT" or higher grade ciphers.

tls_null_cipherlist (eNULL:!aNULL)
The OpenSSL cipherlist for "NULL" grade ciphers that provide authentication without encryption.

OBSOLETE STARTTLS CONTROLS

The following configuration parameters exist for compatibility with Postfix versions before 2.3. Support for these will be removed in a future release.

smtpd_use_tls (no)
Opportunistic TLS: announce STARTTLS support to SMTP clients, but do not require that clients use TLS encryption.

smtpd_enforce_tls (no)
Mandatory TLS: announce STARTTLS support to SMTP clients, and require that clients use TLS encryption.

smtpd_tls_cipherlist (empty)
Obsolete Postfix < 2.3 control for the Postfix SMTP server TLS cipher list.

VERP SUPPORT CONTROLS

With VERP style delivery, each recipient of a message receives a customized copy of the message with his/her own recipient address encoded in the envelope sender address. The VERP_README file describes configuration and operation details of Postfix support for variable envelope return path addresses. VERP style delivery is requested with the SMTP XVERP command or with the "sendmail -V" command-line option and is available in Postfix version 1.1 and later.

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.

Available in Postfix version 1.1 and 2.0:

authorized_verp_clients ($mynetworks)
What SMTP clients are allowed to specify the XVERP command.

Available in Postfix version 2.1 and later:

smtpd_authorized_verp_clients ($authorized_verp_clients)
What SMTP clients are allowed to specify the XVERP command.

TROUBLE SHOOTING CONTROLS

The DEBUG_README document describes how to debug parts of the Postfix mail system. The methods vary from making the software log a lot of detail, to running some daemon processes under control of a call tracer or debugger.

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.

error_notice_recipient (postmaster)
The recipient of postmaster notifications about mail delivery problems that are caused by policy, resource, software or protocol errors.

internal_mail_filter_classes (empty)
What categories of Postfix-generated mail are subject to before-queue content inspection by non_smtpd_milters, header_checks and body_checks.

notify_classes (resource, software)
The list of error classes that are reported to the postmaster.

soft_bounce (no)
Safety net to keep mail queued that would otherwise be returned to the sender.

Available in Postfix version 2.1 and later:

smtpd_authorized_xclient_hosts (empty)
What SMTP clients are allowed to use the XCLIENT feature.

KNOWN VERSUS UNKNOWN RECIPIENT CONTROLS

As of Postfix version 2.0, the SMTP server rejects mail for unknown recipients. This prevents the mail queue from clogging up with undeliverable MAILER-DAEMON messages. Additional information on this topic is in the LOCAL_RECIPIENT_README and ADDRESS_CLASS_README documents.

show_user_unknown_table_name (yes)
Display the name of the recipient table in the "User unknown" responses.

canonical_maps (empty)
Optional address mapping lookup tables for message headers and envelopes.

recipient_canonical_maps (empty)
Optional address mapping lookup tables for envelope and header recipient addresses.

Parameters concerning known/unknown local recipients:

mydestination ($myhostname, localhost.$mydomain, localhost)
The list of domains that are delivered via the $local_transport mail delivery transport.

inet_interfaces (all)
The network interface addresses that this mail system receives mail on.

proxy_interfaces (empty)
The network interface addresses that this mail system receives mail on by way of a proxy or network address translation unit.

inet_protocols (ipv4)
The Internet protocols Postfix will attempt to use when making or accepting connections.

local_recipient_maps (proxy:unix:passwd.byname $alias_maps)
Lookup tables with all names or addresses of local recipients: a recipient address is local when its domain matches $mydestination, $inet_interfaces or $proxy_interfaces.

unknown_local_recipient_reject_code (550)
The numerical Postfix SMTP server response code when a recipient address is local, and $local_recipient_maps specifies a list of lookup tables that does not match the recipient.

Parameters concerning known/unknown recipients of relay destinations:

relay_domains ($mydestination)
What destination domains (and subdomains thereof) this system will relay mail to.

relay_recipient_maps (empty)
Optional lookup tables with all valid addresses in the domains that match $relay_domains.

unknown_relay_recipient_reject_code (550)
The numerical Postfix SMTP server reply code when a recipient address matches $relay_domains, and relay_recipient_maps specifies a list of lookup tables that does not match the recipient address.

Parameters concerning known/unknown recipients in virtual alias domains:

virtual_alias_domains ($virtual_alias_maps)
Postfix is final destination for the specified list of virtual alias domains, that is, domains for which all addresses are aliased to addresses in other local or remote domains.

virtual_alias_maps ($virtual_maps)
Optional lookup tables that alias specific mail addresses or domains to other local or remote address.

unknown_virtual_alias_reject_code (550)
The SMTP server reply code when a recipient address matches $virtual_alias_domains, and $virtual_alias_maps specifies a list of lookup tables that does not match the recipient address.

Parameters concerning known/unknown recipients in virtual mailbox domains:

virtual_mailbox_domains ($virtual_mailbox_maps)
Postfix is final destination for the specified list of domains; mail is delivered via the $virtual_transport mail delivery transport.

virtual_mailbox_maps (empty)
Optional lookup tables with all valid addresses in the domains that match $virtual_mailbox_domains.

unknown_virtual_mailbox_reject_code (550)
The SMTP server reply code when a recipient address matches $virtual_mailbox_domains, and $virtual_mailbox_maps specifies a list of lookup tables that does not match the recipient address.

RESOURCE AND RATE CONTROLS

The following parameters limit resource usage by the SMTP server and/or control client request rates.

line_length_limit (2048)
Upon input, long lines are chopped up into pieces of at most this length; upon delivery, long lines are reconstructed.

queue_minfree (0)
The minimal amount of free space in bytes in the queue file system that is needed to receive mail.

message_size_limit (10240000)
The maximal size in bytes of a message, including envelope information.

smtpd_recipient_limit (1000)
The maximal number of recipients that the Postfix SMTP server accepts per message delivery request.

smtpd_timeout (300s)
The time limit for sending a Postfix SMTP server response and for receiving a remote SMTP client request.

smtpd_history_flush_threshold (100)
The maximal number of lines in the Postfix SMTP server command history before it is flushed upon receipt of EHLO, RSET, or end of DATA.

Available in Postfix version 2.3 and later:

smtpd_peername_lookup (yes)
Attempt to look up the remote SMTP client hostname, and verify that the name matches the client IP address.

The per SMTP client connection count and request rate limits are implemented in co-operation with the **anvil**(8) service, and are available in Postfix version 2.2 and later.

smtpd_client_connection_count_limit (50)
How many simultaneous connections any client is allowed to make to this service.

smtpd_client_connection_rate_limit (0)
The maximal number of connection attempts any client is allowed to make to this service per time unit.

smtpd_client_message_rate_limit (0)
The maximal number of message delivery requests that any client is allowed to make to this service per time unit, regardless of whether or not Postfix actually accepts those messages.

smtpd_client_recipient_rate_limit (0)
The maximal number of recipient addresses that any client is allowed to send to this service per time unit, regardless of whether or not Postfix actually accepts those recipients.

smtpd_client_event_limit_exceptions ($mynetworks)
Clients that are excluded from connection count, connection rate, or SMTP request rate restrictions.

Available in Postfix version 2.3 and later:

smtpd_client_new_tls_session_rate_limit (0)
The maximal number of new (i.e., uncached) TLS sessions that a remote SMTP client is allowed to negotiate with this service per time unit.

TARPIT CONTROLS

When a remote SMTP client makes errors, the Postfix SMTP server can insert delays before responding. This can help to slow down run-away software. The behavior is controlled by an error counter that counts the number of errors within an SMTP session that a client makes without delivering mail.

smtpd_error_sleep_time (1s)
With Postfix version 2.1 and later: the SMTP server response delay after a client has made more than $smtpd_soft_error_limit errors, and fewer than $smtpd_hard_error_limit errors, without delivering mail.

smtpd_soft_error_limit (10)
The number of errors a remote SMTP client is allowed to make without delivering mail before the Postfix SMTP server slows down all its responses.

smtpd_hard_error_limit (20)
The maximal number of errors a remote SMTP client is allowed to make without delivering mail.

smtpd_junk_command_limit (100)
The number of junk commands (NOOP, VRFY, ETRN or RSET) that a remote SMTP client can send before the Postfix SMTP server starts to increment the error counter with each junk command.

Available in Postfix version 2.1 and later:

smtpd_recipient_overshoot_limit (1000)
The number of recipients that a remote SMTP client can send in excess of the limit specified with $smtpd_recipient_limit, before the Postfix SMTP server increments the per-session error count for each excess recipient.

ACCESS POLICY DELEGATION CONTROLS

As of version 2.1, Postfix can be configured to delegate access policy decisions to an external server that runs outside Postfix. See the file SMTPD_POLICY_README for more information.

smtpd_policy_service_max_idle (300s)
The time after which an idle SMTPD policy service connection is closed.

smtpd_policy_service_max_ttl (1000s)
The time after which an active SMTPD policy service connection is closed.

smtpd_policy_service_timeout (100s)
The time limit for connecting to, writing to or receiving from a delegated SMTPD policy server.

ACCESS CONTROLS

The SMTPD_ACCESS_README document gives an introduction to all the SMTP server access control features.

smtpd_delay_reject (yes)
Wait until the RCPT TO command before evaluating $smtpd_client_restrictions, $smtpd_helo_restrictions and $smtpd_sender_restrictions, or wait until the ETRN command before evaluating $smtpd_client_restrictions and $smtpd_helo_restrictions.

parent_domain_matches_subdomains (see 'postconf -d' output)
What Postfix features match subdomains of "domain.tld" automatically, instead of requiring an explicit ".domain.tld" pattern.

smtpd_client_restrictions (empty)
Optional SMTP server access restrictions in the context of a client SMTP connection request.

smtpd_helo_required (no)
Require that a remote SMTP client introduces itself at the beginning of an SMTP session with the HELO or EHLO command.

smtpd_helo_restrictions (empty)
Optional restrictions that the Postfix SMTP server applies in the context of the SMTP HELO command.

smtpd_sender_restrictions (empty)
Optional restrictions that the Postfix SMTP server applies in the context of the MAIL FROM command.

smtpd_recipient_restrictions (permit_mynetworks, reject_unauth_destination)
The access restrictions that the Postfix SMTP server applies in the context of the RCPT TO command.

smtpd_etrn_restrictions (empty)
Optional SMTP server access restrictions in the context of a client ETRN request.

allow_untrusted_routing (no)
Forward mail with sender-specified routing (user[@%!]remote[@%!]site) from untrusted clients to destinations matching $relay_domains.

smtpd_restriction_classes (empty)
User-defined aliases for groups of access restrictions.

smtpd_null_access_lookup_key (<>)
The lookup key to be used in SMTP **access**(5) tables instead of the null sender address.

permit_mx_backup_networks (empty)
Restrict the use of the permit_mx_backup SMTP access feature to only domains whose primary MX hosts match the listed networks.

Available in Postfix version 2.0 and later:

smtpd_data_restrictions (empty)
Optional access restrictions that the Postfix SMTP server applies in the context of the SMTP DATA command.

smtpd_expansion_filter (see 'postconf -d' output)
What characters are allowed in $name expansions of RBL reply templates.

Available in Postfix version 2.1 and later:

smtpd_reject_unlisted_sender (no)
Request that the Postfix SMTP server rejects mail from unknown sender addresses, even when no explicit reject_unlisted_sender access restriction is specified.

smtpd_reject_unlisted_recipient (yes)
Request that the Postfix SMTP server rejects mail for unknown recipient addresses, even when no explicit reject_unlisted_recipient access restriction is specified.

Available in Postfix version 2.2 and later:

smtpd_end_of_data_restrictions (empty)
Optional access restrictions that the Postfix SMTP server applies in the context of the SMTP END-OF-DATA command.

SENDER AND RECIPIENT ADDRESS VERIFICATION CONTROLS

Postfix version 2.1 introduces sender and recipient address verification. This feature is implemented by sending probe email messages that are not actually delivered. This feature is requested via the reject_unverified_sender and reject_unverified_recipient access restrictions. The status of verification probes is maintained by the **verify**(8) server. See the file ADDRESS_VERIFICATION_README for information about how to configure and operate the Postfix sender/recipient address verification service.

address_verify_poll_count (3)
How many times to query the **verify**(8) service for the completion of an address verification request in progress.

address_verify_poll_delay (3s)
The delay between queries for the completion of an address verification request in progress.

address_verify_sender (postmaster)
The sender address to use in address verification probes.

unverified_sender_reject_code (450)
The numerical Postfix SMTP server response code when a recipient address is rejected by the reject_unverified_sender restriction.

unverified_recipient_reject_code (450)
The numerical Postfix SMTP server response when a recipient address is rejected by the reject_unverified_recipient restriction.

ACCESS CONTROL RESPONSES

The following parameters control numerical SMTP reply codes and/or text responses.

access_map_reject_code (554)
The numerical Postfix SMTP server response code when a client is rejected by an **access**(5) map restriction.

defer_code (450)
The numerical Postfix SMTP server response code when a remote SMTP client request is rejected by the "defer" restriction.

invalid_hostname_reject_code (501)
The numerical Postfix SMTP server response code when the client HELO or EHLO command parameter is rejected by the reject_invalid_helo_hostname restriction.

maps_rbl_reject_code (554)
The numerical Postfix SMTP server response code when a remote SMTP client request is blocked by the reject_rbl_client, reject_rhsbl_client, reject_rhsbl_sender or reject_rhsbl_recipient restriction.

non_fqdn_reject_code (504)
The numerical Postfix SMTP server reply code when a client request is rejected by the reject_non_fqdn_helo_hostname, reject_non_fqdn_sender or reject_non_fqdn_recipient restriction.

plaintext_reject_code (450)
The numerical Postfix SMTP server response code when a request is rejected by the **reject_plaintext_session** restriction.

reject_code (554)
The numerical Postfix SMTP server response code when a remote SMTP client request is rejected by the "reject" restriction.

relay_domains_reject_code (554)
The numerical Postfix SMTP server response code when a client request is rejected by the reject_unauth_destination recipient restriction.

unknown_address_reject_code (450)
The numerical Postfix SMTP server response code when a sender or recipient address is rejected by the reject_unknown_sender_domain or reject_unknown_recipient_domain restriction.

unknown_client_reject_code (450)
The numerical Postfix SMTP server response code when a client without valid address <=> name mapping is rejected by the reject_unknown_client_hostname restriction.

unknown_hostname_reject_code (450)
The numerical Postfix SMTP server response code when the hostname specified with the HELO or EHLO command is rejected by the reject_unknown_helo_hostname restriction.

Available in Postfix version 2.0 and later:

default_rbl_reply (see 'postconf -d' output)
The default SMTP server response template for a request that is rejected by an RBL-based restriction.

multi_recipient_bounce_reject_code (550)
The numerical Postfix SMTP server response code when a remote SMTP client request is blocked by the reject_multi_recipient_bounce restriction.

rbl_reply_maps (empty)
Optional lookup tables with RBL response templates.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

command_directory (see 'postconf -d' output)
The location of all postfix administrative commands.

double_bounce_sender (double-bounce)
The sender address of postmaster notifications that are generated by the mail system.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

mail_name (Postfix)
The mail system name that is displayed in Received: headers, in the SMTP greeting banner, and in bounced mail.

mail_owner (postfix)
The UNIX system account that owns the Postfix queue and most Postfix daemon processes.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

myhostname (see 'postconf -d' output)
The internet hostname of this mail system.

mynetworks (see 'postconf -d' output)
The list of "trusted" SMTP clients that have more privileges than "strangers".

myorigin ($myhostname)
The domain name that locally-posted mail appears to come from, and that locally posted mail is delivered to.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

queue_directory (see 'postconf -d' output)
The location of the Postfix top-level queue directory.

recipient_delimiter (empty)
The separator between user names and address extensions (user+foo).

smtpd_banner ($myhostname ESMTP $mail_name)
The text that follows the 220 status code in the SMTP greeting banner.

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".

Available in Postfix version 2.2 and later:

smtpd_forbidden_commands (CONNECT, GET, POST)
List of commands that causes the Postfix SMTP server to immediately terminate the session with a 221 code.

SEE ALSO

anvil(8), connection/rate limiting
cleanup(8), message canonicalization
tlsmgr(8), TLS session and PRNG management
trivial-rewrite(8), address resolver
verify(8), address verification service
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
ADDRESS_CLASS_README, blocking unknown hosted or relay recipients
ADDRESS_REWRITING_README Postfix address manipulation
FILTER_README, external after-queue content filter
LOCAL_RECIPIENT_README, blocking unknown local recipients
MILTER_README, before-queue mail filter applications
SMTPD_ACCESS_README, built-in access policies
SMTPD_POLICY_README, external policy server
SMTPD_PROXY_README, external before-queue content filter
SASL_README, Postfix SASL howto
TLS_README, Postfix STARTTLS howto
VERP_README, Postfix XVERP extension
XCLIENT_README, Postfix XCLIENT extension
XFORWARD_README, Postfix XFORWARD extension

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

SASL support originally by:
Till Franke

SuSE Rhein/Main AG
65760 Eschborn, Germany

TLS support originally by:
Lutz Jaenicke
BTU Cottbus
Allgemeine Elektrotechnik
Universitaetsplatz 3-4
D-03044 Cottbus, Germany

NAME

spawn − Postfix external command spawner

SYNOPSIS

spawn [generic Postfix daemon options] command_attributes...

DESCRIPTION

The **spawn**(8) daemon provides the Postfix equivalent of **inetd**. It listens on a port as specified in the Postfix **master.cf** file and spawns an external command whenever a connection is established. The connection can be made over local IPC (such as UNIX-domain sockets) or over non-local IPC (such as TCP sockets). The command´s standard input, output and error streams are connected directly to the communication endpoint.

This daemon expects to be run from the **master**(8) process manager.

COMMAND ATTRIBUTE SYNTAX

The external command attributes are given in the **master.cf** file at the end of a service definition. The syntax is as follows:

user=*username* (required)

user=*username*:*groupname*

The external command is executed with the rights of the specified *username*. The software refuses to execute commands with root privileges, or with the privileges of the mail system owner. If *groupname* is specified, the corresponding group ID is used instead of the group ID of *username*.

argv=*command...* (required)

The command to be executed. This must be specified as the last command attribute. The command is executed directly, i.e. without interpretation of shell meta characters by a shell command interpreter.

BUGS

In order to enforce standard Postfix process resource controls, the **spawn**(8) daemon runs only one external command at a time. As such, it presents a noticeable overhead by wasting precious process resources. The **spawn**(8) daemon is expected to be replaced by a more structural solution.

DIAGNOSTICS

The **spawn**(8) daemon reports abnormal child exits. Problems are logged to **syslogd**(8).

SECURITY

This program needs root privilege in order to execute external commands as the specified user. It is therefore security sensitive. However the **spawn**(8) daemon does not talk to the external command and thus is not vulnerable to data-driven attacks.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically as **spawn**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

In the text below, *transport* is the first field of the entry in the **master.cf** file.

RESOURCE AND RATE CONTROL

*transport***_time_limit ($command_time_limit)**

The amount of time the command is allowed to run before it is terminated.

Postfix 2.4 and later support a suffix that specifies the time unit: s (seconds), m (minutes), h (hours), d (days), w (weeks). The default time unit is seconds.

MISCELLANEOUS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

export_environment (see 'postconf -d' output)
The list of environment variables that a Postfix process will export to non-Postfix processes.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

mail_owner (postfix)
The UNIX system account that owns the Postfix queue and most Postfix daemon processes.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

postconf(5), configuration parameters
master(8), process manager
syslogd(8), system logging

LICENSE

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

spray — send many packets to host

SYNOPSIS

spray [**-c** *count*] [**-d** *delay*] [**-l** *length*] *host ...*

DESCRIPTION

spray sends multiple RPC packets to *host* and records how many of them were correctly received and how long it took.

The options are as follows:

-c *count*
Send *count* packets.

-d *delay*
Pause *delay* microseconds between sending each packet.

-l *length*
Set the length of the packet that holds the RPC call message to *length* bytes. Not all values of *length* are possible because RPC data is encoded using XDR. **spray** rounds up to the nearest possible value.

spray is intended for use in network testing, measurement, and management. This command *can be very hard on a network and should be used with caution.*

SEE ALSO

netstat(1), ifconfig(8), ping(8), rpc.sprayd(8)

NAME

ssh-keysign — ssh helper program for host-based authentication

SYNOPSIS

ssh-keysign

DESCRIPTION

ssh-keysign is used by `ssh`(1) to access the local host keys and generate the digital signature required during host-based authentication with SSH protocol version 2.

ssh-keysign is disabled by default and can only be enabled in the global client configuration file `/etc/ssh/ssh_config` by setting **EnableSSHKeysign** to "yes".

ssh-keysign is not intended to be invoked by the user, but from `ssh`(1). See `ssh`(1) and `sshd`(8) for more information about host-based authentication.

FILES

`/etc/ssh/ssh_config`
Controls whether **ssh-keysign** is enabled.

`/etc/ssh/ssh_host_dsa_key, /etc/ssh/ssh_host_rsa_key`
These files contain the private parts of the host keys used to generate the digital signature. They should be owned by root, readable only by root, and not accessible to others. Since they are readable only by root, **ssh-keysign** must be set-uid root if host-based authentication is used.

SEE ALSO

`ssh`(1), `ssh-keygen`(1), `ssh_config`(5), `sshd`(8)

HISTORY

ssh-keysign first appeared in OpenBSD 3.2.

AUTHORS

Markus Friedl ⟨markus@openbsd.org⟩

NAME

sshd — OpenSSH SSH daemon

SYNOPSIS

sshd [**-46Ddeiqt**] [**-b** *bits*] [**-f** *config_file*] [**-g** *login_grace_time*] [**-h** *host_key_file*] [**-k** *key_gen_time*] [**-o** *option*] [**-p** *port*] [**-u** *len*]

DESCRIPTION

sshd (OpenSSH Daemon) is the daemon program for ssh(1). Together these programs replace rlogin(1) and rsh(1), and provide secure encrypted communications between two untrusted hosts over an insecure network.

sshd listens for connections from clients. It is normally started at boot from /etc/rc.d/sshd. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

sshd can be configured using command-line options or a configuration file (by default sshd_config(5)); command-line options override values specified in the configuration file. **sshd** rereads its configuration file when it receives a hangup signal, SIGHUP, by executing itself with the name and options it was started with, e.g. /usr/sbin/sshd.

The options are as follows:

-4 Forces **sshd** to use IPv4 addresses only.

-6 Forces **sshd** to use IPv6 addresses only.

-b *bits*
Specifies the number of bits in the ephemeral protocol version 1 server key (default 768).

-D When this option is specified, **sshd** will not detach and does not become a daemon. This allows easy monitoring of **sshd**.

-d Debug mode. The server sends verbose debug output to the system log, and does not put itself in the background. The server also will not fork and will only process one connection. This option is only intended for debugging for the server. Multiple **-d** options increase the debugging level. Maximum is 3.

-e When this option is specified, **sshd** will send the output to the standard error instead of the system log.

-f *config_file*
Specifies the name of the configuration file. The default is /etc/ssh/sshd_config. **sshd** refuses to start if there is no configuration file.

-g *login_grace_time*
Gives the grace time for clients to authenticate themselves (default 120 seconds). If the client fails to authenticate the user within this many seconds, the server disconnects and exits. A value of zero indicates no limit.

-h *host_key_file*
Specifies a file from which a host key is read. This option must be given if **sshd** is not run as root (as the normal host key files are normally not readable by anyone but root). The default is /etc/ssh/ssh_host_key for protocol version 1, and /etc/ssh/ssh_host_rsa_key and /etc/ssh/ssh_host_dsa_key for protocol version 2. It is possible to have multiple host key files for the different protocol versions and host key algorithms.

-i Specifies that **sshd** is being run from `inetd`(8). **sshd** is normally not run from inetd because it needs to generate the server key before it can respond to the client, and this may take tens of seconds. Clients would have to wait too long if the key was regenerated every time. However, with small key sizes (e.g. 512) using **sshd** from inetd may be feasible.

-k *key_gen_time*
Specifies how often the ephemeral protocol version 1 server key is regenerated (default 3600 seconds, or one hour). The motivation for regenerating the key fairly often is that the key is not stored anywhere, and after about an hour it becomes impossible to recover the key for decrypting intercepted communications even if the machine is cracked into or physically seized. A value of zero indicates that the key will never be regenerated.

-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, and their values, see `sshd_config`(5).

-p *port*
Specifies the port on which the server listens for connections (default 22). Multiple port options are permitted. Ports specified in the configuration file with the **Port** option are ignored when a command-line port is specified. Ports specified using the **ListenAddress** option override command-line ports.

-q Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged.

-t Test mode. Only check the validity of the configuration file and sanity of the keys. This is useful for updating **sshd** reliably as configuration options may change.

-u *len*
This option is used to specify the size of the field in the `utmp` structure that holds the remote host name. If the resolved host name is longer than *len*, the dotted decimal value will be used instead. This allows hosts with very long host names that overflow this field to still be uniquely identified. Specifying **-u0** indicates that only dotted decimal addresses should be put into the `utmp` file. **-u0** may also be used to prevent **sshd** from making DNS requests unless the authentication mechanism or configuration requires it. Authentication mechanisms that may require DNS include **RhostsRSAAuthentication**, **HostbasedAuthentication**, and using a **from="pattern-list"** option in a key file. Configuration options that require DNS include using a USER@HOST pattern in **AllowUsers** or **DenyUsers**.

AUTHENTICATION

The OpenSSH SSH daemon supports SSH protocols 1 and 2. Both protocols are supported by default, though this can be changed via the **Protocol** option in `sshd_config`(5). Protocol 2 supports both RSA and DSA keys; protocol 1 only supports RSA keys. For both protocols, each host has a host-specific key, normally 2048 bits, used to identify the host.

Forward security for protocol 1 is provided through an additional server key, normally 768 bits, generated when the server starts. This key is normally regenerated every hour if it has been used, and is never stored on disk. Whenever a client connects, the daemon responds with its public host and server keys. The client compares the RSA host key against its own database to verify that it has not changed. The client then generates a 256-bit random number. It encrypts this random number using both the host key and the server key, and sends the encrypted number to the server. Both sides then use this random number as a session key which is used to encrypt all further communications in the session. The rest of the session is encrypted using a conventional cipher, currently Blowfish or 3DES, with 3DES being used by default. The client selects the encryption algorithm to use from those offered by the server.

For protocol 2, forward security is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key. The rest of the session is encrypted using a symmetric cipher, currently 128-bit AES, Blowfish, 3DES, CAST128, Arcfour, 192-bit AES, or 256-bit AES. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (hmac-md5, hmac-sha1, umac-64 or hmac-ripemd160).

Finally, the server and the client enter an authentication dialog. The client tries to authenticate itself using host-based authentication, public key authentication, challenge-response authentication, or password authentication.

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time the client may request things like allocating a pseudo-tty, forwarding X11 connections, forwarding TCP connections, or forwarding the authentication agent connection over the secure channel.

After this, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side may send data at any time, and such data is forwarded to/from the shell or command on the server side, and the user terminal in the client side.

When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.

LOGIN PROCESS

When a user successfully logs in, **sshd** does the following:

1. If the login is on a tty, and no command has been specified, prints last login time and `/etc/motd` (unless prevented in the configuration file or by `~/.hushlogin`; see the **FILES** section).
2. If the login is on a tty, records login time.
3. Checks `/etc/nologin`; if it exists, prints contents and quits (unless root).
4. Changes to run with normal user privileges.
5. Sets up basic environment.
6. Reads the file `~/.ssh/environment`, if it exists, and users are allowed to change their environment. See the **PermitUserEnvironment** option in `sshd_config`(5).
7. Changes to user's home directory.
8. If `~/.ssh/rc` exists, runs it; else if `/etc/ssh/sshrc` exists, runs it; otherwise runs xauth. The "rc" files are given the X11 authentication protocol and cookie in standard input. See **SSHRC**, below.
9. Runs user's shell or command.

SSHRC

If the file `~/.ssh/rc` exists, `sh`(1) runs it after reading the environment files but before starting the user's shell or command. It must not produce any output on stdout; stderr must be used instead. If X11 forwarding is in use, it will receive the "proto cookie" pair in its standard input (and `DISPLAY` in its environment). The script must call `xauth`(1) because **sshd** will not run xauth automatically to add X11 cookies.

The primary purpose of this file is to run any initialization routines which may be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment.

This file will probably contain some initialization code followed by something similar to:

```
    if read proto cookie && [ -n "$DISPLAY" ]; then
            if [ ‘echo $DISPLAY | cut -c1-10‘ = ’localhost:’ ]; then
                    # X11UseLocalhost=yes
                    echo add unix:‘echo $DISPLAY |
                        cut -c11-‘ $proto $cookie
            else
                    # X11UseLocalhost=no
                    echo add $DISPLAY $proto $cookie
            fi | xauth -q -
    fi
```

If this file does not exist, `/etc/ssh/sshrc` is run, and if that does not exist either, xauth is used to add the cookie.

AUTHORIZED_KEYS FILE FORMAT

AuthorizedKeysFile specifies the file containing public keys for public key authentication; if none is specified, the default is `~/.ssh/authorized_keys`. Each line of the file contains one key (empty lines and lines starting with a ‘#’ are ignored as comments). Protocol 1 public keys consist of the following space-separated fields: options, bits, exponent, modulus, comment. Protocol 2 public key consist of: options, keytype, base64-encoded key, comment. The options field is optional; its presence is determined by whether the line starts with a number or not (the options field never starts with a number). The bits, exponent, modulus, and comment fields give the RSA key for protocol version 1; the comment field is not used for anything (but may be convenient for the user to identify the key). For protocol version 2 the keytype is “ssh-dss” or “ssh-rsa”.

Note that lines in this file are usually several hundred bytes long (because of the size of the public key encoding) up to a limit of 8 kilobytes, which permits DSA keys up to 8 kilobits and RSA keys up to 16 kilobits. You don’t want to type them in; instead, copy the `identity.pub`, `id_dsa.pub`, or the `id_rsa.pub` file and edit it.

sshd enforces a minimum RSA key modulus size for protocol 1 and protocol 2 keys of 768 bits.

The options (if present) consist of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported (note that option keywords are case-insensitive):

command="command"
: Specifies that the command is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is run on a pty if the client requests a pty; otherwise it is run without a tty. If an 8-bit clean channel is required, one must not request a pty or should specify **no-pty**. A quote may be included in the command by quoting it with a backslash. This option might be useful to restrict certain public keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Note that the client may specify TCP and/or X11 forwarding unless they are explicitly prohibited. The command originally supplied by the client is available in the `SSH_ORIGINAL_COMMAND` environment variable. Note that this option applies to shell, command or subsystem execution.

environment="NAME=value"
: Specifies that the string is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted. Environment processing is disabled by default and is controlled via the **PermitUserEnvironment** option. This option is automatically disabled if **UseLogin** is enabled.

from="pattern-list"
Specifies that in addition to public key authentication, the canonical name of the remote host must be present in the comma-separated list of patterns. The purpose of this option is to optionally increase security: public key authentication by itself does not trust the network or name servers or anything (but the key); however, if somebody somehow steals the key, the key permits an intruder to log in from anywhere in the world. This additional option makes using a stolen key more difficult (name servers and/or routers would have to be compromised in addition to just the key).

See **PATTERNS** in `ssh_config`(5) for more information on patterns.

no-agent-forwarding
Forbids authentication agent forwarding when this key is used for authentication.

no-port-forwarding
Forbids TCP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, e.g. in connection with the **command** option.

no-pty
Prevents tty allocation (a request to allocate a pty will fail).

no-user-rc
Disables execution of `~/.ssh/rc`.

no-X11-forwarding
Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

permitopen="host:port"
Limit local ''`ssh -L`'' port forwarding such that it may only connect to the specified host and port. IPv6 addresses can be specified with an alternative syntax: *host*/*port*. Multiple **permitopen** options may be applied separated by commas. No pattern matching is performed on the specified hostnames, they must be literal domains or addresses.

tunnel="n"
Force a `tun`(4) device on the server. Without this option, the next available device will be used if the client requests a tunnel.

An example authorized_keys file:

```
# Comments allowed at start of line
ssh-rsa AAAAB3Nza...LiPk== user@example.net
from="*.sales.example.net,!pc.sales.example.net" ssh-rsa
AAAAB2...19Q== john@example.net
command="dump /home",no-pty,no-port-forwarding ssh-dss
AAAAC3...51R== example.net
permitopen="192.0.2.1:80",permitopen="192.0.2.2:25" ssh-dss
AAAAB5...21S==
tunnel="0",command="sh /etc/netstart tun0" ssh-rsa AAAA...==
jane@example.net
```

SSH_KNOWN_HOSTS FILE FORMAT

The `/etc/ssh/ssh_known_hosts` and `~/.ssh/known_hosts` files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional), and the per-user file is maintained automatically: whenever the user connects from an unknown host, its key is added to the per-user file.

Each line in these files contains the following fields: hostnames, bits, exponent, modulus, comment. The fields are separated by spaces.

Hostnames is a comma-separated list of patterns ('*' and '?' act as wildcards); each pattern in turn is matched against the canonical host name (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern may also be preceded by '!' to indicate negation: if the host name matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line. A hostname or address may optionally be enclosed within '[' and ']' brackets then followed by ':' and a non-standard port number.

Alternately, hostnames may be stored in a hashed form which hides host names and addresses should the file's contents be disclosed. Hashed hostnames start with a '|' character. Only one hashed hostname may appear on a single line and none of the above negation or wildcard operators may be applied.

Bits, exponent, and modulus are taken directly from the RSA host key; they can be obtained, for example, from `/etc/ssh/ssh_host_key.pub`. The optional comment field continues to the end of the line, and is not used.

Lines starting with '#' and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key. It is thus permissible (but not recommended) to have several lines or different host keys for the same names. This will inevitably happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information; authentication is accepted if valid information can be found from either file.

Note that the lines in these files are typically hundreds of characters long, and you definitely don't want to type in the host keys by hand. Rather, generate them by a script or by taking `/etc/ssh/ssh_host_key.pub` and adding the host names at the front.

An example ssh_known_hosts file:

```
# Comments allowed at start of line
closenet,...,192.0.2.53 1024 37 159...93 closenet.example.net
cvs.example.net,192.0.2.10 ssh-rsa AAAA1234.....=
# A hashed hostname
|1|JfKTdBh7rNbXkVAQCRp4OQoPfmI=|USECr3SWf1JUPsms5AqfD5QfxkM= ssh-rsa
AAAA1234.....=
```

FILES

~/.hushlogin
: This file is used to suppress printing the last login time and `/etc/motd`, if **PrintLastLog** and **PrintMotd**, respectively, are enabled. It does not suppress printing of the banner specified by **Banner**.

~/.rhosts
: This file is used for host-based authentication (see `ssh`(1) for more information). On some machines this file may need to be world-readable if the user's home directory is on an NFS partition, because **sshd** 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 above. The content of the file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

  If this file, the `~/.ssh` directory, or the user's home directory are writable by other users, then the file could be modified or replaced by unauthorized users. In this case, **sshd** will not allow it to be used unless the **StrictModes** option has been set to "no". The recommended permissions can be set by executing "chmod go-w ~/ ~/.ssh ~/.ssh/authorized_keys".

~/.ssh/environment
: This file is read into the environment at login (if it exists). It can only contain empty lines, comment lines (that start with '#'), and assignment lines of the form name=value. The file should be writable only by the user; it need not be readable by anyone else. Environment processing is disabled by default and is controlled via the **PermitUserEnvironment** option.

~/.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. The format of this file is described above. This file should be writable only by root/the owner and can, but need not be, world-readable.

~/.ssh/rc
: Contains initialization routines to be run before the user's home directory becomes accessible. This file should be writable only by the user, and need not be readable by anyone else.

/etc/hosts.allow
/etc/hosts.deny
: Access controls that should be enforced by tcp-wrappers are defined here. Further details are described in `hosts_access`(5).

/etc/hosts.equiv
: This file is for host-based authentication (see `ssh`(1)). It should only be writable by root.

/etc/moduli
: Contains Diffie-Hellman groups used for the "Diffie-Hellman Group Exchange". The file format is described in `moduli`(5).

/etc/motd
: See `motd`(5).

/etc/nologin
: If this file exists, **sshd** refuses to let anyone except root log in. The contents of the file are displayed to anyone trying to log in, and non-root connections are refused. The file should be world-readable.

/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_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. These files should only be owned by root, readable only by root, and not accessible to others. Note that **sshd** does not start if these files are group/world-accessible.

/etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_rsa_key.pub
: These three files contain the public parts of the host keys. These files should be world-readable but writable only by root. Their contents should match the respective private parts. These files are not really used for anything; they are provided for the convenience of the user so their contents can be copied to known hosts files. These files are created using `ssh-keygen`(1).

/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. The format of this file is described above. This file should be writable only by root/the owner and should be world-readable.

/etc/ssh/sshd_config
: Contains configuration data for **sshd**. The file format and configuration options are described in `sshd_config`(5).

/etc/ssh/sshrc
: Similar to `~/.ssh/rc`, it can be used to specify machine-specific login-time initializations globally. This file should be writable only by root, and should be world-readable.

/var/empty
: `chroot`(2) directory used by **sshd** during privilege separation in the pre-authentication phase. The directory should not contain any files and must be owned by root and not group or world-writable.

/var/run/sshd.pid
: Contains the process ID of the **sshd** listening for connections (if there are several daemons running concurrently for different ports, this contains the process ID of the one started last). The content of this file is not sensitive; it can be world-readable.

SEE ALSO

`scp`(1), `sftp`(1), `ssh`(1), `ssh-add`(1), `ssh-agent`(1), `ssh-keygen`(1), `ssh-keyscan`(1), `chroot`(2), `hosts_access`(5), `login.conf`(5), `moduli`(5), `sshd_config`(5), `inetd`(8), `sftp-server`(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. Niels Provos and Markus Friedl contributed support for privilege separation.

CAVEATS

System security is not improved unless **rshd**, **rlogind**, and **rexecd** are disabled (thus completely disabling `rlogin` and `rsh` into the machine).

NAME

stdethers — a NIS filter program

SYNOPSIS

stdethers [*file*]

DESCRIPTION

stdethers parses the ethers(5) style input stream (stdin, or *file* if given), and outputs lines containing only the MAC address and host name.

stdethers is used by other NIS programs when creating some of the NIS maps.

SEE ALSO

nis(8), ypserv(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

stdhosts — a NIS filter program

SYNOPSIS

stdhosts [**-n**] [*file*]

DESCRIPTION

stdhosts parses the `hosts`(5) style input stream (stdin, or *file* if given), and outputs lines containing only the IPv4 address and host name.

stdhosts is used by other NIS programs when creating some of the NIS maps.

-n allows other address types in the output, including IPv6 addresses.

SEE ALSO

`nis`(8), `ypserv`(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

sti — Simulate Terminal Input: send characters to a tty device

SYNOPSIS

sti *tty* [*string ...*]

DESCRIPTION

sti will send the provided *string* to the *tty* specified in the command line using `ioctl`(2) `TIOCSTI`, or send the standard input if a *string* is not supplied. This `ioctl`(2) is limited to the superuser.

The *string* is interpreted using `unvis`(3).

SEE ALSO

`ioctl`(2), `unvis`(3)

HISTORY

The **sti** first appeared at NetBSD 4.0.

AUTHORS

The **sti** command was written by Christos Zoulas ⟨christos@NetBSD.org⟩.

NAME

strfile, **unstr** — create a random access file for storing strings

SYNOPSIS

strfile [**-iorsx**] [**-c** *char*] *source_file* [*output_file*]
unstr *source_file*

DESCRIPTION

strfile reads a file containing groups of lines separated by a line containing a single percent '%' sign and creates a data file which contains a header structure and a table of file offsets for each group of lines. This allows random access of the strings.

The output file, if not specified on the command line, is named *source_file*.**out**.

The options are as follows:

-c *char*
Change the delimiting character from the percent sign to *char*.

-i Ignore case when ordering the strings.

-o Order the strings in alphabetical order. The offset table will be sorted in the alphabetical order of the groups of lines referenced. Any initial non-alphanumeric characters are ignored. This option causes the STR_ORDERED bit in the header *str_flags* field to be set.

-r Randomize access to the strings. Entries in the offset table will be randomly ordered. This option causes the STR_RANDOM bit in the header *str_flags* field to be set.

-s Run silently; don't give a summary message when finished.

-x Note that each alphabetic character in the groups of lines is rotated 13 positions in a simple caesar cipher. This option causes the STR_ROTATED bit in the header *str_flags* field to be set.

The format of the header is:

```
#define VERSION 1
unsigned long  str_version;   /* version number */
unsigned long  str_numstr;    /* # of strings in the file */
unsigned long  str_longlen;   /* length of longest string */
unsigned long  str_shortlen;  /* length of shortest string */
#define STR_RANDOM     0x1    /* randomized pointers */
#define STR_ORDERED    0x2    /* ordered pointers */
#define STR_ROTATED    0x4    /* rot-13'd text */
unsigned long  str_flags;     /* bit field for flags */
char str_delim;                      /* delimiting character */
```

All fields are written in big-endian byte order.

The purpose of **unstr** is to undo the work of **strfile**. It prints out the strings contained in the file *source_file* in the order that they are listed in the header file *source_file*.**dat** to standard output. It is possible to create sorted versions of input files by using **-o** when **strfile** is run and then using **unstr** to dump them out in the table order.

FILES

strfile.out default output file.

SEE ALSO

`byteorder`(3), `fortune`(6)

HISTORY

The **`strfile`** utility first appeared in 4.4BSD.

NAME

string2key — map a password into a key

SYNOPSIS

string2key [**−5** | **−-version5**] [**−4** | **−-version4**] [**−a** | **−-afs**] [**−c** *cell* | **−-cell=***cell*] [**−w** *password* | **−-password=***password*] [**−p** *principal* | **−-principal=***principal*] [**−k** *string* | **−-keytype=***string*] *password*

DESCRIPTION

string2key performs the string-to-key function. This is useful when you want to handle the raw key instead of the password. Supported options:

−5, **−-version5**
: Output Kerberos v5 string-to-key

−4, **−-version4**
: Output Kerberos v4 string-to-key

−a, **−-afs**
: Output AFS string-to-key

−c *cell*, **−-cell=***cell*
: AFS cell to use

−w *password*, **−-password=***password*
: Password to use

−p *principal*, **−-principal=***principal*
: Kerberos v5 principal to use

−k *string*, **−-keytype=***string*
: Keytype

−-version
: print version

−-help

NAME

sunlabel — read or modify a SunOS disk label

SYNOPSIS

sunlabel [**-mnqs**] *device*

DESCRIPTION

sunlabel reads or modifies a SunOS disk label on *device*, which is used by the PROM on NetBSD/sparc hardware to find partitions to boot from. **sunlabel** only reads/writes the first 512 bytes of *device*.

The supported options are:

-m	Ignore an incorrect magic number in the disk label.
-n	Synthesize a new label rather than reading what is there.
-q	Quiet mode - don't print unnecessary babble (currently this suppresses the "sunlabel>" prompt).
-s	Ignore checksum errors when reading the label.

Note that **-m** is dangerous, especially when combined with **-s**, since it will then happily believe whatever garbage it may find in the label. When using these flags, all values should be checked carefully, both those printed by **L** and the partition table printed by **P**.

sunlabel prints a prompt "sunlabel>" and expects commands. The following commands are understood:

?	Show a short help message.
[abcdefghijklmnop] *<cylno> <size>*	Change partition (see below).
L	Print label, except for the partition table.
P	Print the partition table.
Q	Quit program (error if no write since last change).
Q!	Quit program (unconditionally) [EOF also quits].
S	Set label in the kernel (orthogonal to **W**).
V *<name> <value>*	Change a non-partition label value.
W	Write (possibly modified) label out.

The **a** through **p** commands will accept, for the *<size>* parameter, the *nnn/nnn/nnn* syntax used by SunOS 4.x **format**. (For those not familiar with this syntax, *a/b/c* means *a* cylinders + *b* tracks + *c* sectors. For example, if the disk has 16 tracks of 32 sectors, *3/4/5* means (3∗16∗32)+(4∗32)+5=1669. This calculation always uses the *nsect* and *ntrack* values as printed by the **L** command; in particular, if they are zero (which they will initially be if **-n** is used), this syntax is not very useful. Some additional strings are accepted. For the *<cylno>* parameter, "end-X" (where *X* is a partition letter) indicates that the partition should start with the first free cylinder after partition *X*; "start-X" indicates that the partition should start at the same place as partition *X*. For the *<size>* parameter, "end-X" indicates that the partition should end at the same place as partition *X* (even if partition *X* ends partway through a cylinder); "start-X" indicates that the partition should end with the last cylinder before partition *X*; and "size-X" means that the partition's size should exactly match partition *X*'s size.

Note that **sunlabel** supports 16 partitions. SunOS supports only 8. Labels written by **sunlabel**, when partitions *i* through *p* are all set *offset=0 size=0*, are identical to Sun labels. If any of the “extended” partitions are nontrivial, information about them is tucked into some otherwise unused space in the Sun label format.

The **V** command changes fields printed by the **L** command. For example, if the **L** command prints

```
ascii: ST15230N cyl 5657 alt 2 hd 19 sec 78
rpm: 0          pcyl: 0         apc: 0          obs1: 0
obs2: 0         intrlv: 1       ncyl: 5657      acyl: 0
nhead: 19       nsect: 78       obs3: 0         obs4: 0
```

then **V** *ncyl 6204* would set the *ncyl* value to 6204, or **V** *ascii Seagate ST15230N cyl 5657 hd 19 sec varying* would set the ascii-label string to that string. **sunlabel** performs very few consistency checks on the values you supply, and the ones it does perform never generate errors, only warnings.

AUTHORS

der Mouse ⟨mouse@rodents.montreal.qc.ca⟩

BUGS

It may be that the space in the label where the information for the extended partitions is saved is used by SunOS.

Not very many consistency checks are done on the **V** arguments, and those only produce warnings.

NetBSD doesn't support 16 partitions in a Sun disk label yet.

NAME

supfilesrv, supscan − sup server processes

SYNOPSIS

supfilesrv [*-4*] [*-6*] [*-d*] [*-l*] [*-q*] [*-N*] [*-P*] [*-C MaxChildren*]
supscan [*-v*] [*-s*] [*collection*] [*basedir*]

DESCRIPTION

Supfilesrv is the server processes used to interact with *sup* client processes via the IP/TCP network protocol. This server normally is expected to be running on server machines at all times. Each machine with files of interest to users on other machines is expected to be a file server and should run *supfilesrv.*

A file server machine will service requests for both "private" and "system" file collections. No special action is necessary to support private collections, as the client user is expected to supply all necessary information. For system collections, if the base directory is not the default (see FILES below), an entry must be put into the directory list file; this entry is a single text line containing the name of the collection, one or more spaces, and the name of the base directory for that collection.

Each collection should have an entry in the host list file; this entry is a single text line containing the name of the collection, one or more spaces, and the name of the host machine acting as file server for that collection.

Details of setting up a file collection for the file server are described in the manual entry for *sup(1).*

Supfilesrv generally runs as a network server process that listens for connections, and for each connection (double-)forks a process to handle the interaction with the client. However, with the -d flag, no forking will take place: the server will listen for a network connection, handle it, and exit. This is useful for debugging the servers in "live" mode rather than as daemons.

For debugging purposes, the -P "debugging ports" flag can be used. It will cause the selection of an alternate, non-privileged set of TCP ports instead of the usual ports, which are reserved for the active server processes. The -N "network debugging" flag can be used to produce voluminous messages describing the network communication progress and status. The more -N switches that you use the more output you get. Use 3 (separated by spaces: -N -N -N) to get a complete record of all network messages. Log messages are printed by *syslog* on *daemon.log* . To suppress log messages, the -q "quiet" flag can be used.

supfilesrv uses libwrap style access control (the /etc/hosts.allow and /etc/hosts.deny files) with service name "supfilesrv". The -l "log" flag turn on loggin of accepted connections (denied connections are always logged).

Normally the *supfilesrv* will only respond to 3 requests simultaneously, forking a child process for each client. If it gets additional requests it will respond with the error FSSETUPBUSY. The -C MaxChildren switch can be used to increase (or decrease) this number.

supfilesrv listens to IPv4 listening socket by default. With the -6 flag, it will listen to IPv6 listening socket. For dual stack support you will want to run two instances of *supfilesrv.*

SUPSCAN

It is possible to pre-compile a list of the files in a collection to make *supfilesrv* service that collection much faster. This can be done by running *supscan* on the desired collection on the repository machine. This produces a list of all the files in the collection at the time of the *supscan;* subsequent upgrades will be based on this list of files rather than actually scanning the disk at the time of the upgrade. Of course, the upgrade will consequently bring the client machine up to the status of the repository machine as of the time of the *supscan* rather than as of the time of the upgrade; hence, if *supscan* is used, it should be run periodically on the collection. This facility is useful for extremely large file collections that are upgraded many times per day, such as the CMU UNIX system software. The "verbose" flag *-v* will cause *supscan* to produce output messages as it scans the files in the collection. The "system" flag *-s* will cause *supscan* to scan all system collections residing on the current host. The *basedir* parameter must be specified if the collection is a private collection whose base directory is not the default.

FILES

/usr default base directory for a collection

/etc/supfiles/coll.dir
base directory list for system collections

/etc/supfiles/coll.host
host name list for system collections

<base-directory>/sup/<collection>/*
files used by file server (see *sup(1))*

<base-directory>/sup/<collection>/list
list file used by *supscan* to create file list

<base-directory>/sup/<collection>/scan
file list created by *supscan* from list file

SEE ALSO

sup(1) hosts_access(5) hosts_options(5)
The SUP Software Upgrade Protocol, S. A. Shafer, CMU Computer Science Dept., 1985.

DIAGNOSTICS

The file server places log messages on the standard and diagnostic output files. The process name and process id number generally accompany each message for diagnostic purposes.

HISTORY

31-July-92 Mary Thompson (mrt) at Carnegie Mellon University
Removed references to supnameserver which has not existed for a long time. Update a few file names. Added -C switch.

21-May-87 Glenn Marcy (gm0w) at Carnegie-Mellon University
Updated documentation for 4.3; changed /usr/cmu to /usr/cs.

15-Jan-86 Glenn Marcy (gm0w) at Carnegie-Mellon University
Updated documentation; -s switch to supscan.

23-May-85 Steven Shafer (sas) at Carnegie-Mellon University
Supscan created and documented; also -N flag.

04-Apr-85 Steven Shafer (sas) at Carnegie-Mellon University
Created.

NAME

svhlabel — update disk label from SGI Volume Header

SYNOPSIS

svhlabel [**-fqrw**] *device*

DESCRIPTION

svhlabel is used to update a NetBSD disk label from the Silicon Graphics Volume Header on disks that were previously used on IRIX systems.

svhlabel scans the Volume Header contained in the first blocks of the disk and generates additional partition entries for the disk from the entries found.

Each Volume Header entry which does not have an equivalent partition in the disk label (equivalent in having the same size and offset) is added to the first free partition slot in the disk label. A free partition slot is defined as one with an `fstype` of ‘`unused`’ and a `size` of zero (‘0’). If there are not enough free slots in the disk label, a warning will be issued.

The raw partition (typically partition *c*, but *d* on i386 and some other platforms) is left alone during this process.

By default, the proposed changed disk label will be displayed and no disk label update will occur.

Available options:

-f
Force an update, even if there has been no change.

-q
Performs operations in a quiet fashion.

-w
Update the in-core label if it has been changed.

-r
In conjunction with **-w**, also update the on-disk label. You probably do not want to do this.

SEE ALSO

`disklabel`(8), `dkctl`(8), `mount_efs`(8), `sgivol`(8)

HISTORY

The **svhlabel** command appeared in NetBSD 5.0.

NAME

swapctl, **swapon** — system swap management tool

SYNOPSIS

swapctl −A [**−f** | **−o**] [**−n**] [**−p** *priority*] [**−t** *blk*|*noblk*|*auto*]
swapctl −D *dumpdev*|*none*
swapctl −U [**−n**] [**−t** *blk*|*noblk*|*auto*]
swapctl −a [**−p** *priority*] *path*
swapctl −c −p *priority path*
swapctl −d *path*
swapctl −l | **−s** [**−k** | **−m** | **−g** | **−h**]
swapctl −q
swapctl −z
swapon −a [**−t** *blk*|*noblk*]
swapon *path*

DESCRIPTION

The **swapctl** program adds, removes, lists and prioritizes swap devices and files for the system. The **swapon** program acts the same as the **swapctl** program, as if called with the **−a** option, except if **swapon** itself is called with **−a** in which case, **swapon** acts as **swapctl** with the **−A** option.

The following options are available:

−A This option causes **swapctl** to read the /etc/fstab file for devices and files with a "sw" or "dp" type, and adds all "sw" type entries as swap devices and sets the last "dp" type entry as the dump device. If no swap devices are configured, **swapctl** will exit with an error code. If used together with **−t** *auto* this option will not read /etc/fstab but query the kernel for all swap partitions on local hard disks.

−a The **−a** option requires that a *path* also be in the argument list. The *path* is added to the kernel's list of swap devices using the swapctl(2) system call. When using the **swapon** form of this command, the **−a** option is treated the same as the **−A** option, for backwards compatibility.

−c The **−c** option changes the priority of the listed swap device or file.

−D The **−D** option requires that a *dumpdev* also be in the argument list. The kernel dump device is set to *dumpdev*. The word "none" can be used instead of a *dumpdev* to disable the currently set dump device. This change is made via the swapctl(2) system call. The dump device is used when the system crashes to write a current snapshot of real memory, to be saved later with savecore(8) at system reboot, and analyzed to determine the problem.

−d The **−d** option removes the listed *path* from the kernel's list of swap devices or files.

−f Used in combination with the **−A** command and **−t** *auto* flag this option makes **swapctl** use the first discovered swap device to also become the dump device. The **−f** option is mutually exclusive with the **−o** option.

−g The **−g** option uses (1024 ∗ 1024 ∗ 1024) byte blocks instead of the default 512 byte.

−h The **−h** option uses humanize_number(3) to display the sizes.

−k The **−k** option uses 1024 byte blocks instead of the default 512 byte.

−l The **−l** option lists the current swap devices and files, and their usage statistics.

−m The **−m** option uses (1024 ∗ 1024) byte blocks instead of the default 512 byte.

-n Used with the **-A** or **-U** command, the **-n** option makes **swapctl** print the action it would take, but not actually change any swap or dump devices.

-o Similar to the **-f** flag, this "Dump Only" option makes **swapctl** find the first swap device and configure it as dump device. No swap device is changed. This option needs to be used in combination with **-A -t** *auto* and is mutually exclusive with **-f**.

-p The **-p** option sets the priority of swap devices or files to the *priority* argument. This works with the **-a**, **-c**, and **-l** options.

-q Query /etc/fstab, checking for any defined swap or dump devices. If any are found, **swapctl** returns with an exit status of 0, if none are found the exit status will be 1.

-s The **-s** option displays a single line summary of current swap statistics.

-t This flag modifies the function of the **-A** and **-U** options. The **-t** option allows the type of device to add to be specified. An argument of *blk* causes all block devices in /etc/fstab to be added. An argument of *noblk* causes all non-block devices in /etc/fstab to be added. An argument of *auto* causes all swap partitions on local hard disks to be used. This option is useful in early system startup, where swapping may be needed before all file systems are available, such as during disk checks of large file systems.

-U This option causes **swapctl** to read the /etc/fstab file for devices and files with a "sw" type, and remove all these entries as swap devices. If no swap devices are unconfigured, **swapctl** will exit with an error code. If used together with **-t** *auto* this option will not read /etc/fstab but unconfigure all local swap partitions.

-z The **-z** option displays the current dump device.

SWAP PRIORITY

The NetBSD swap system allows different swap devices and files to be assigned different priorities, to allow the faster resources to be used first. Swap devices at the same priority are used in a round-robin fashion until there is no more space available at this priority, when the next priority level will be used. The default priority is 0, the highest. This value can be any valid integer, with higher values receiving less priority.

SWAP OPTIONS

When parsing the /etc/fstab file for swap devices, the following options are recognized:

priority=N	This option sets the priority of the specified swap device to N.
nfsmntpt=/path	This option is useful for swapping to NFS files. It specifies the local mount point to mount an NFS filesystem. The mount point must exist as a directory. Typically, once this mount has succeeded, the file to be used for swapping on will be available under this point mount. For example:

```
server:/export/swap/client none swap sw,nfsmntpt=/swap
```

EXIT STATUS

If the requested operation was sucessful, the **swapctl** utility exits with status 0. If an error occurred, the exit status is 1.

For easy scriptability, the **-z** operation (query dump device) and **-l** (list swap partitions) return an exit status of 1 if no dump device or swap partition has been configured. If any swap partition is available or a dump device is set, the respective query returns 0.

SEE ALSO

`swapctl`(2), `fstab`(5), `mount_nfs`(8)

HISTORY

The **swapctl** program was first made available in NetBSD 1.3. The original **swapon** program, provided for backwards compatibility, appeared in 4.0BSD.

AUTHORS

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

CAVEATS

Using the automatic swap partition detection done by the **-A -t** *auto* option may be dangereous. Depending on the on-disk partitioning scheme used, the type of a partition may not be accurately recognizable as a swap partition. The autodetection might recognize and use partitions on removable media like USB sticks. An easy way to test the autoconfiguration is to use **swapctl** with the **-n** option.

BUGS

If no swap information is specified in `/etc/fstab`, the system startup scripts (see `rc`(8)) will configure no swap space and your machine will behave very badly if (more likely when) it runs out of real memory.

Local and remote swap files cannot be configured until after the file systems they reside on are mounted read/write. The system startup scripts need to `fsck`(8) all local file systems before this can happen. This process requires substantial amounts of memory on some systems. If you configure no local block swap devices on a machine that has local file systems to check and rely only on swap files, the machine will have no swap space at all during system `fsck`(8) and may run out of real memory, causing fsck to abnormally exit and startup scripts to fail.

NAME

sync — force completion of pending disk writes (flush cache)

SYNOPSIS

sync

DESCRIPTION

The **sync** program can be called to ensure that all disk writes have been completed before the processor is halted in a way not suitably done by `reboot`(8) or `halt`(8). Generally, it is preferable to use `reboot`(8) or `halt`(8) to shut down the system, as they may perform additional actions such as resynchronizing the hardware clock and flushing internal caches before performing a final **sync**.

The **sync** program simply invokes the `sync`(2) system call.

SEE ALSO

`fsync`(2), `sync`(2), `halt`(8), `reboot`(8)

HISTORY

A **sync** command appeared in Version 6 AT&T UNIX.

NAME

sysctl — get or set kernel state

SYNOPSIS

sysctl [**-AdeMn**] [**-r** | **-x**] [*name ...*]
sysctl [**-nq**] [**-r** | **-x**] **-w** *name*=*value ...*
sysctl [**-en**] [**-r** | **-x**] **-a**
sysctl [**-nq**] [**-r** | **-x**] **-f** *file*

DESCRIPTION

The **sysctl** utility retrieves kernel state and allows processes with appropriate privilege to set kernel state. The state to be retrieved or set is described using a "Management Information Base" ("MIB") style name, described as a dotted set of components. The '/' character may also be used as a separator and a leading separator character is accepted. If *name* specifies a non-leaf node in the MIB, all the nodes underneath *name* will be printed.

The following options are available:

-A List all the known MIB names including tables, unless any MIB arguments or **-f** *file* are given. Those with string or integer values will be printed as with the **-a** flag; for table or structure values that **sysctl** is not able to print, the name of the utility to retrieve them is given. Errors in retrieving or setting values will be directed to stdout instead of stderr.

-a List all the currently available string or integer values. The use of a solitary separator character (either '.' or '/') by itself has the same effect. Any given *name* arguments are ignored if this option is specified.

-d Descriptions of each of the nodes selected will be printed instead of their values.

-e Separate the name and the value of the variable(s) with '='. This is useful for producing output which can be fed back to the **sysctl** utility. This option is ignored if **-n** is specified or a variable is being set.

-f Specifies the name of a file to read and process. Blank lines and comments (beginning with '#') are ignored. Line continuations with '\' are permitted. Remaining lines are processed similarly to command line arguments of the form *name* or *name*=*value*. The **-w** flag is implied by **-f**. Any *name* arguments are ignored.

-M Makes **sysctl** print the MIB instead of any of the actual values contained in the MIB. This causes the entire MIB to be printed unless specific MIB arguments or **-f** *file* are also given.

-n Specifies that the printing of the field name should be suppressed and that only its value should be output. This flag is useful for setting shell variables. For example, to save the pagesize in variable psize, use:

```
      set psize=`sysctl -n hw.pagesize`
```

-q Used to indicate that nothing should be printed for writes unless an error is detected.

-r Raw output form. Values printed are in their raw binary forms as retrieved directly from the kernel. Some additional nodes that **sysctl** cannot print directly can be retrieved with this flag. This option conflicts with the **-x** option.

-w Sets the MIB style name given to the value given. The MIB style name and value must be separated by '=' with no whitespace. Only integral and string values can be set via this method.

-x Makes **sysctl** print the requested value in a hexadecimal representation instead of its regular form. If specified more than once, the output for each value resembles that of hexdump(1) when given the **-C** flag. This option conflicts with the **-r** option.

The `proc` top-level MIB has a special semantic: it represent per-process values and as such may differ from one process to another. The second-level name is the pid of the process (in decimal form), or the special word `curproc`. For variables below `proc.`⟨pid⟩.rlimit, the integer value may be replaced with the string `unlimited` if it matches the magic value used to disable a limit.

The information available from **sysctl** consists of integers, strings, and tables. The tabular information can only be retrieved by special purpose programs such as **ps**, **systat**, and **netstat**. See `sysctl`(7) for description of available MIBs.

CREATION AND DELETION

New nodes are allowed to be created by the superuser when the kernel is running at security level 0. These new nodes may refer to existing kernel data or to new data that is only instrumented by `sysctl`(3) itself.

The syntax for creating new nodes is "//create=new.node.path" followed by one or more of the following attributes separated by commas. The use of a double separator (both '/' and '.' can be used as separators) as the prefix tells sysctl that the first series of tokens is not a MIB name, but a command. It is recommended that the double separator preceding the command not be the same as the separator used in naming the MIB entry so as to avoid possible parse conflicts. The "value" assigned, if one is given, must be last.

- *type*=⟨*T*⟩ where *T* must be one of "node", "int", "string", "quad", or "struct". If the type is omitted, the "node" type is assumed.
- *size*=⟨*S*⟩ here, *S* asserts the size of the new node. Nodes of type "node" should not have a size set. The size may be omitted for nodes of types "int" or "quad". If the size is omitted for a node of type "string", the size will be determined by the length of the given value, or by the kernel for kernel strings. Nodes of type "struct" must have their size explicitly set.
- *addr*=⟨*A*⟩ or *symbol*=⟨*A*⟩ The kernel address of the data being instrumented. If "symbol" is used, the symbol must be globally visible to the in-kernel `ksyms`(4) driver.
- *n*=⟨*N*⟩ The MIB number to be assigned to the new node. If no number is specified, the kernel will assign a value.
- *flags*=⟨*F*⟩ A concatenated string of single letters that govern the behavior of the node. Flags currently available are:

a	Allow anyone to write to the node, if it is writable.
h	"Hidden". **sysctl** must be invoked with **−A** or the hidden node must be specifically requested in order to see it
i	"Immediate". Makes the node store data in itself, rather than allocating new space for it. This is the default for nodes of type "int" and "quad". This is the opposite of owning data.
o	"Own". When the node is created, separate space will be allocated to store the data to be instrumented. This is the default for nodes of type "string" and "struct" where it is not possible to guarantee sufficient space to store the data in the node itself.
p	"Private". Nodes that are marked private, and children of nodes so marked, are only viewable by the superuser. Be aware that the immediate data that some nodes may store is not necessarily protected by this.
x	"Hexadecimal". Make **sysctl** default to hexadecimal display of the retrieved value
r	"Read-only". The data instrumented by the given node is read-only. Note that other mechanisms may still exist for changing the data. This is the default for nodes that instrument data.
w	"Writable". The data instrumented by the given node is writable at any time. This is the default for nodes that can have children.

1 "Read-only at securelevel 1". The data instrumented by this node is writable until the securelevel reaches or passes securelevel 1. Examples of this include some network tunables.

2 "Read-only at securelevel 2". The data instrumented by this node is writable until the securelevel reaches or passes securelevel 2. An example of this is the per-process core filename setting.

- `value`=⟨*V*⟩ An initial starting value for a new node that does not reference existing kernel data. Initial values can only be assigned for nodes of the "int", "quad", and "string" types.

New nodes must fit the following set of criteria:

- If the new node is to address an existing kernel object, only one of the "symbol" or "addr" arguments may be given.
- The size for a "struct" type node must be specified; no initial value is expected or permitted.
- Either the size or the initial value for a "string" node must be given.
- The node which will be the parent of the new node must be writable.

If any of the given parameters describes an invalid configuration, **sysctl** will emit a diagnostic message to the standard error and exit.

Descriptions can be added by the super-user to any node that does not have one, provided that the node is not marked with the "PERMANENT" flag. The syntax is similar to the syntax for creating new nodes with the exception of the keyword that follows the double separator at the start of the command: "//describe=new.node.path=new node description". Once a description has been added, it cannot be changed or removed.

When destroying nodes, only the path to the node is necessary, i.e., "//destroy=old.node.path". No other parameters are expected or permitted. Nodes being destroyed must have no children, and their parent must be writable. Nodes that are marked with the "`PERMANENT`" flag (as assigned by the kernel) may not be deleted.

In all cases, the initial '=' that follows the command (eg, "create", "destroy", or "describe") may be replaced with another instance of the separator character, provided that the same separator character is used for the length of the name specification.

FILES

`/etc/sysctl.conf` **sysctl** variables set at boot time

EXAMPLES

For example, to retrieve the maximum number of processes allowed in the system, one would use the following request:

```
sysctl kern.maxproc
```

To set the maximum number of processes allowed in the system to 1000, one would use the following request:

```
sysctl -w kern.maxproc=1000
```

Information about the system clock rate may be obtained with:

```
sysctl kern.clockrate
```

Information about the load average history may be obtained with:

```
sysctl vm.loadavg
```

To view the values of the per-process variables of the current shell, the request:

```
sysctl proc.$$
```

can be used if the shell interpreter replaces $$ with its pid (this is true for most shells).

To redirect core dumps to the `/var/tmp/`⟨username⟩ directory,

```
sysctl -w proc.$$.corename=/var/tmp/%u/%n.core
```

should be used.

```
sysctl -w proc.curproc.corename=/var/tmp/%u/%n.core
```

changes the value for the sysctl process itself, and will not have the desired effect.

To create the root of a new sub-tree called “local” add some children to the new node, and some descriptions:

```
sysctl -w //create=local
sysctl -w //describe=local=my local sysctl tree
sysctl -w //create=local.esm_debug,type=int,symbol=esm_debug,flags=w
sysctl -w //describe=local.esm_debug=esm driver debug knob
sysctl -w //create=local.audiodebug,type=int,symbol=audiodebug,flags=w
sysctl -w //describe=local.audiodebug=generic audio debug knob
```

Note that the children are made writable so that the two debug settings in question can be tuned arbitrarily.

To destroy that same subtree:

```
sysctl -w //destroy=local.esm_debug
sysctl -w //destroy=local.audiodebug
sysctl -w //destroy=local
```

SEE ALSO

`sysctl`(3), `ksyms`(4), `sysctl`(7)

HISTORY

`sysctl` first appeared in 4.4BSD.

NAME

sysinst — install or upgrade a NetBSD system

SYNOPSIS

sysinst

DESCRIPTION

sysinst is a menu-based program that may be used to install or upgrade a NetBSD system. It is usually invoked automatically when the system is booted from appropriate installation media.

sysinst is usually not present on a NetBSD system that has been fully installed.

SEE ALSO

release(7), afterboot(8), boot(8), diskless(8), <*machine*>/INSTALL.* files on CD-ROM installation media, .../NetBSD-<*rel*>/<*machine*>/INSTALL.* files in NetBSD releases or snapshots.

HISTORY

A **sysinst** command appeared in NetBSD 1.3.

NAME

syslogd — log systems messages

SYNOPSIS

syslogd [**-dnrSsTUv**] [**-b** *bind_address*] [**-f** *config_file*] [**-g** *group*]
[**-m** *mark_interval*] [**-P** *file_list*] [**-p** *log_socket*
[**-p** *log_socket2 ...*]] [**-t** *chroot_dir*] [**-u** *user*]

DESCRIPTION

syslogd reads and logs messages to the system console, log files, other machines and/or users as specified by its configuration file. The options are as follows:

-b *bind_address*
: Specify one specific IP address or hostname to bind to. If a hostname is specified, the IPv4 or IPv6 address which corresponds to it is used.

-d Enable debugging to the standard output, and do not disassociate from the controlling terminal.

-f Specify the pathname of an alternative configuration file; the default is `/etc/syslog.conf`.

-g *group* Set GID to *group* after the sockets and log files have been opened.

-m Select the number of minutes between ‘‘mark’’ messages; the default is 20 minutes.

-n Do not perform hostname lookups; report only numeric addresses.

-P Specify the pathname of a file containing a list of sockets to be created. The format of the file is simply one socket per line.

-p Specify the pathname of a log socket. Multiple **-p** options create multiple log sockets. If no **-p** arguments are created, the default socket of `/var/run/log` is used.

-r Disable the compression of repeated instances of the same line into a single line of the form “last message repeated N times”.

-S Sync kernel messages to disk immediately.

-s Select “secure” mode, in which **syslogd** does not listen on a UDP socket but only communicates over a UNIX domain socket. This is valuable when the machine on which **syslogd** runs is subject to attack over the network and it is desired that the machine be protected from attempts to remotely fill logs and similar attacks.

-t *chroot_dir*
: `chroot`(2) to *chroot_dir* after the sockets and log files have been opened.

-T Always use the local time and date for messages received from the network, instead of the timestamp field supplied in the message by the remote host. This is useful if some of the originating hosts can't keep time properly or are unable to generate a correct timestamp.

-u *user* Set UID to *user* after the sockets and log files have been opened.

-U Unique priority logging. Only log messages at the priority specified by the selector in the configuration file. Without this option, messages at the specified priority or higher are logged. This option changes the default priority comparison from ‘>=’ to ‘=’.

-v Verbose logging. If specified once, the numeric facility and priority are logged with each locally-written message. If specified more than once, the names of the facility and priority are logged with each locally-written message.

syslogd reads its configuration file when it starts up and whenever it receives a hangup signal. For information on the format of the configuration file, see `syslog.conf`(5).

syslogd reads messages from the UNIX domain socket `/var/run/log`, from an Internet domain socket specified in `/etc/services`, and from the special device `/dev/klog` (to read kernel messages).

syslogd creates the file `/var/run/syslogd.pid`, and stores its process id there. This can be used to kill or reconfigure **syslogd**.

By using multiple **-p** options, one can set up many chroot environments by passing the pathname to the log socket (`/var/run/log`) in each chroot area to **syslogd**. For example:

```
      syslogd -p /var/run/log -p /web/var/run/log -p /ftp/var/run/log
```

Note: the normal log socket must now also be passed to **syslogd**.

The logged message includes the date, time, and hostname (or pathname of the log socket). Commonly, the program name and the process id is included.

The date and time are taken from the received message. If the format of the timestamp field is incorrect, time obtained from the local host is used instead. This can be overridden by the **-T** flag.

Accesses from UDP socket can be filtered by libwrap configuration files, like `/etc/hosts.deny`. Specify "`syslogd`" in *daemon_list* portion of the configuration files. Refer to `hosts_access`(5) for details.

SYSLOG PROTOCOL NOTES

The message sent to **syslogd** should consist of a single line. The message can contain a priority code, which should be a preceding decimal number in angle braces, for example, '⟨5⟩'. This priority code should map into the priorities defined in the include file ⟨`sys/syslog.h`⟩. See RFC 3164 for detailed description of the message format.

Messages from the local kernel that are not tagged with a priority code receive the default facility `LOG_KERN` and priority `LOG_NOTICE`. All other untagged messages receive the default facility `LOG_USER` and priority `LOT_NOTICE`.

FILES

`/etc/syslog.conf`	The configuration file.
`/var/run/syslogd.pid`	The process id of current **syslogd**.
`/var/run/log`	Name of the UNIX domain datagram log socket.
`/dev/klog`	The kernel log device.

SEE ALSO

`logger`(1), `syslog`(3), `services`(5), `syslog.conf`(5), `newsyslog`(8)

The BSD syslog Protocol, RFC, 3164, August 2001.

HISTORY

The **syslogd** command appeared in 4.3BSD. Support for multiple log sockets appeared in NetBSD 1.4. libwrap support appeared in NetBSD 1.6.

NAME

tadpolectl — get or set tadpole microcontroller state

SYNOPSIS

tadpolectl [**-n**] *name ...*
tadpolectl [**-n**] **-w** *name=value ...*
tadpolectl [**-n**] **-a**

DESCRIPTION

The **tadpolectl** utility retrieves values from the ts102 microcontroller
and allows processes with appropriate privilege to set some values. The state to be retrieved or set is described using a ``Management Information Base'' (``MIB'') style name, described as a dotted set of components. The **-a** flag can be used to list all the currently available string or integer values.

The **-n** flag specifies that the printing of the field name should be suppressed and that only its value should be output. This flag is useful for setting shell variables. For example, to save the mains power status in variable mains, use:

```
set mains='tadpolectl -n hw.power.mains'
```

If just a MIB style name is given, the corresponding value is retrieved. If a value is to be set, the **-w** flag must be specified and the MIB name followed by an equal sign and the new value to be used.

The information available from **tadpolectl** consists of only integers. Some registers can be modified, but have no way of reading what the current value is. Those registers will always display "0".

The changeable column indicates whether a process with appropriate privilege can change the value, and if a displayed value is valid.

Name	**Changeable**	**Valid**
hw.microcontroller.version	no	yes
hw.version	no	yes
hw.poweroncycles	no	yes
hw.poweronseconds	no	yes
hw.power.mains	no	yes
hw.power.battery.int	no	yes
hw.power.battery.ext	no	yes
hw.power.battery.chargedisabled	yes	yes
hw.power.battery.int.chargerate	yes	yes
hw.power.battery.ext.chargerate	yes	yes
hw.power.battery.int.chargelevel	no	yes
hw.power.battery.ext.chargelevel	no	yes
hw.video.external	no	yes
hw.video.lid	no	yes
hw.video.syncinva	yes	yes
hw.video.syncinvb	yes	yes
hw.video.compsync	yes	yes
hw.video.tft.brightness	yes	yes
hw.speaker.freq	yes	no
hw.speaker.volume	yes	yes
hw.kbd.repeat.delay	yes	yes
hw.kbd.repeat.speed	yes	yes
hw.kbd.click	yes	yes

hw.mouse.recalibrate	yes	no
hw.mouse.disable	yes	yes
hw.mouse.intclick	yes	yes
hw.mouse.extclick	yes	yes
hw.mouse.sensitivity	yes	yes
hw.serial.power	yes	yes

EXAMPLES

For example, to retrieve the current internal battery charge level, one would use the following request:

```
tadpolectl hw.power.battery.int.chargelevel
```

To set the speaker beep frequency of the system to 1000, one would use the following request:

```
tadpolectl -w hw.speaker.freq=1000
```

SEE ALSO

`sysctl`(8)

HISTORY

`tadpolectl` first appeared in NetBSD 1.5.

NAME

ntalkd, **talkd** — remote user communication server

SYNOPSIS

ntalkd [**-dl**]

DESCRIPTION

ntalkd is the server that notifies a user that someone else wants to initiate a conversation. It acts as a repository of invitations, responding to requests by clients wishing to rendezvous to hold a conversation.

In normal operation, a client, the caller, initiates a rendezvous by sending a CTL_MSG to the server of type LOOK_UP (see ⟨protocols/talkd.h⟩). This causes the server to search its invitation tables to check if an invitation currently exists for the caller (to speak to the callee specified in the message). If the lookup fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the callee's login ports requesting contact.

When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

OPTIONS

ntalkd supports the following options:

-d The **-d** option turns on debugging logging.

-l The **-l** option turns on accounting logging for **ntalkd** via the syslogd(8) service.

FILES

/usr/libexec/ntalkd

SEE ALSO

talk(1), write(1), syslog(3), syslogd(8)

HISTORY

The **ntalkd** command appeared in 4.3BSD.

The original talkd program was coded improperly, in a machine and byte-order dependent fashion. When this was corrected, it required a protocol change, which necessitated a different daemon to handle it, thus **ntalkd** or "new" talk daemon. The old daemon has long since been removed, but the detritus remain.

NAME

tbrconfig — configure a token bucket regulator for an output queue

SYNOPSIS

tbrconfig *interface* [*tokenrate* [*bucketsize*]]
tbrconfig −d *interface*
tbrconfig −a

DESCRIPTION

tbrconfig configures a token bucket regulator for the output network interface queue. A token bucket regulator limits both the average amount and instantaneous amount of packets that the underlying driver can dequeue from the network interface within the kernel.

Conceptually, tokens accumulate in a bucket at the average *tokenrate*, up to the *bucketsize*. The driver can dequeue packets as long as there are positive amount of tokens, and the length of the dequeued packet is subtracted from the remaining tokens. Tokens can be negative as a deficit, and packets are not dequeued from the interface queue until the tokens become positive again. The *tokenrate* limits the average rate, and the *bucketsize* limits the maximum burst size.

Limiting the burst size is essential to packet scheduling, since the scheduler schedules packets backlogged at the network interface. Limiting the burst size is also needed for drivers which dequeues more packets than they can send and end up with discarding excess packets.

When the *tokenrate* is set to higher than the actual transmission rate, the transmission complete interrupt will trigger the next dequeue. On the other hand, when the *tokenrate* is set to lower than the actual transmission rate, the transmission complete interrupt would occur before the tokens become positive. In this case, the next dequeue will be triggered by a timer event. Because the kernel timer has a limited granularity, a larger *bucketsize* is required for a higher *tokenrate*.

The *interface* parameter is a string of the form “name unit”, for example, “en0”.

The *tokenrate* parameter specifies the average rate in bits per second, and “K” or “M” can be appended to *tokenrate* as a short hand of “Kilo-bps” or “Mega-bps”, respectively. When *tokenrate* is omitted, **tbrconfig** displays the current parameter values.

The *bucketsize* parameter specifies the bucket size in bytes, and “K” can be appended to *bucketsize* as a short hand of “Kilo-bytes”. When *bucketsize* is omitted, **tbrconfig** assumes the regulator is driven by transmission complete interrupts and, using heuristics, assigns a small bucket size according to the *tokenrate*. When the keyword “auto” is given as *bucketsize*, **tbrconfig** assumes the regulator is driven by the kernel timer, and computes the bucket size from *tokenrate* and the kernel clock frequency.

If the **−d** flag is passed before an interface name, **tbrconfig** will remove the token bucket regulator for the specified interface.

Optionally, the **−a** flag may be used instead of an interface name. This flag instructs **tbrconfig** to display information about all interfaces in the system.

EXAMPLES

To configure a token bucket regulator for the interface en0 with 10Mbps token rate and 8KB bucket size,

```
# tbrconfig en0 10M 8K
```

To rate-limit the interface en0 up to 3Mbps,

```
# tbrconfig en0 3M auto
```

SEE ALSO

altq.conf(5), altqd(8)

HISTORY

The **tbrconfig** command first appeared in WIDE/KAME IPv6 protocol stack kit as part of ALTQ tools.

NAME

tcpdchk − tcp wrapper configuration checker

SYNOPSIS

tcpdchk [-a] [-d] [-i inet_conf] [-v]

DESCRIPTION

tcpdchk examines your tcp wrapper configuration and reports all potential and real problems it can find. The program examines the *tcpd* access control files (by default, these are */etc/hosts.allow* and */etc/hosts.deny*), and compares the entries in these files against entries in the *inetd* or *tlid* network configuration files.

tcpdchk reports problems such as non-existent pathnames; services that appear in *tcpd* access control rules, but are not controlled by *tcpd*; services that should not be wrapped; non-existent host names or non-internet address forms; occurrences of host aliases instead of official host names; hosts with a name/address conflict; inappropriate use of wildcard patterns; inappropriate use of NIS netgroups or references to non-existent NIS netgroups; references to non-existent options; invalid arguments to options; and so on.

Where possible, *tcpdchk* provides a helpful suggestion to fix the problem.

OPTIONS

-a Report access control rules that permit access without an explicit ALLOW keyword. This applies only when the extended access control language is enabled (build with -DPROCESS_OPTIONS).

-d Examine *hosts.allow* and *hosts.deny* files in the current directory instead of the default ones.

-i inet_conf
Specify this option when *tcpdchk* is unable to find your *inetd.conf* or *tlid.conf* network configuration file, or when you suspect that the program uses the wrong one.

-v Display the contents of each access control rule. Daemon lists, client lists, shell commands and options are shown in a pretty-printed format; this makes it easier for you to spot any discrepancies between what you want and what the program understands.

FILES

The default locations of the *tcpd* access control tables are:

/etc/hosts.allow
/etc/hosts.deny

SEE ALSO

tcpdmatch(8), explain what tcpd would do in specific cases.
hosts_access(5), format of the tcpd access control tables.
hosts_options(5), format of the language extensions.
inetd.conf(5), format of the inetd control file.

AUTHORS

Wietse Venema (wietse@wzv.win.tue.nl),
Department of Mathematics and Computing Science,
Eindhoven University of Technology
Den Dolech 2, P.O. Box 513,
5600 MB Eindhoven, The Netherlands

NAME

tcpdmatch − tcp wrapper oracle

SYNOPSIS

tcpdmatch [-d] [-i inet_conf] daemon client

tcpdmatch [-d] [-i inet_conf] daemon[@server] [user@]client

DESCRIPTION

tcpdmatch predicts how the tcp wrapper would handle a specific request for service. Examples are given below.

The program examines the *tcpd* access control tables (default */etc/hosts.allow* and */etc/hosts.deny*) and prints its conclusion. For maximal accuracy, it extracts additional information from your *inetd* or *tlid* network configuration file.

When *tcpdmatch* finds a match in the access control tables, it identifies the matched rule. In addition, it displays the optional shell commands or options in a pretty-printed format; this makes it easier for you to spot any discrepancies between what you want and what the program understands.

ARGUMENTS

The following two arguments are always required:

daemon
: A daemon process name. Typically, the last component of a daemon executable pathname.

client
: A host name or network address, or one of the ‘unknown’ or ‘paranoid’ wildcard patterns.

  When a client host name is specified, *tcpdmatch* gives a prediction for each address listed for that client.

  When a client address is specified, *tcpdmatch* predicts what *tcpd* would do when client name lookup fails.

Optional information specified with the *daemon@server* form:

server
: A host name or network address, or one of the ‘unknown’ or ‘paranoid’ wildcard patterns. The default server name is ‘unknown’.

Optional information specified with the *user@client* form:

user
: A client user identifier. Typically, a login name or a numeric userid. The default user name is ‘unknown’.

OPTIONS

-d
: Examine *hosts.allow* and *hosts.deny* files in the current directory instead of the default ones.

-i inet_conf
: Specify this option when *tcpdmatch* is unable to find your *inetd.conf* or *tlid.conf* network configuration file, or when you suspect that the program uses the wrong one.

EXAMPLES

To predict how *tcpd* would handle a telnet request from the local system:

```
tcpdmatch in.telnetd localhost
```

The same request, pretending that hostname lookup failed:

```
tcpdmatch in.telnetd 127.0.0.1
```

To predict what tcpd would do when the client name does not match the client address:

```
tcpdmatch in.telnetd paranoid
```

On some systems, daemon names have no ‘in.’ prefix, or *tcpdmatch* may need some help to locate the inetd configuration file.

FILES

The default locations of the *tcpd* access control tables are:

/etc/hosts.allow
/etc/hosts.deny

SEE ALSO

tcpdchk(8), tcpd configuration checker
hosts_access(5), format of the tcpd access control tables.
hosts_options(5), format of the language extensions.
inetd.conf(5), format of the inetd control file.

AUTHORS

Wietse Venema (wietse@wzv.win.tue.nl),
Department of Mathematics and Computing Science,
Eindhoven University of Technology
Den Dolech 2, P.O. Box 513,
5600 MB Eindhoven, The Netherlands

BUGS

If you specify FQDN hostname as client, they will be recognized only as IPv4 or IPv6 address, which should be recognized as both.

NAME

tcpdrop — drop a TCP connection

SYNOPSIS

tcpdrop *laddr lport faddr fport*

DESCRIPTION

The **tcpdrop** command drops the TCP connection specified by the local address *laddr*, port *lport* and the foreign address *faddr*, port *fport*. Addresses and ports can be specified by name or numeric value.

EXAMPLES

If a connection to httpd(8) is causing congestion on a network link, one can drop the TCP session in charge:

```
$ fstat | grep 'httpd.*internet.*<--'
www      httpd      21307    3* internet stream tcp \
       0xd1007ca8 192.168.5.41:80 <-- 192.168.5.1:26747
```

The following command will drop the connection:

```
# tcpdrop 192.168.5.41 80 192.168.5.1 26747
```

SEE ALSO

fstat(1), netstat(1)

NAME

tcpdump − dump traffic on a network

SYNOPSIS

tcpdump [**−AdDeflLnNOpqRStuUvxX**] [**−c** *count*]
[**−C** *file_size*] [**−F** *file*]
[**−i** *interface*] [**−m** *module*] [**−M** *secret*]
[**−r** *file*] [**−s** *snaplen*] [**−T** *type*] [**−w** *file*]
[**−W** *filecount*]
[**−E** *spi@ipaddr algo:secret,...*]
[**−y** *datalinktype*] [**−Z** *user*]
[*expression*]

DESCRIPTION

Tcpdump prints out a description of the contents of packets on a network interface that match the boolean *expression*. It can also be run with the **−w** flag, which causes it to save the packet data to a file for later analysis, and/or with the **−r** flag, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match *expression* will be processed by *tcpdump*.

Tcpdump will, if not run with the **−c** flag, continue capturing packets until it is interrupted by a SIGINT signal (generated, for example, by typing your interrupt character, typically control-C) or a SIGTERM signal (typically generated with the **kill**(1) command); if run with the **−c** flag, it will capture packets until it is interrupted by a SIGINT or SIGTERM signal or the specified number of packets have been processed.

When *tcpdump* finishes capturing packets, it will report counts of:

packets ‘‘captured’’ (this is the number of packets that *tcpdump* has received and processed);

packets ‘‘received by filter’’ (the meaning of this depends on the OS on which you’re running *tcpdump*, and possibly on the way the OS was configured - if a filter was specified on the command line, on some OSes it counts packets regardless of whether they were matched by the filter expression and, even if they were matched by the filter expression, regardless of whether *tcpdump* has read and processed them yet, on other OSes it counts only packets that were matched by the filter expression regardless of whether *tcpdump* has read and processed them yet, and on other OSes it counts only packets that were matched by the filter expression and were processed by *tcpdump*);

packets ‘‘dropped by kernel’’ (this is the number of packets that were dropped, due to a lack of buffer space, by the packet capture mechanism in the OS on which *tcpdump* is running, if the OS reports that information to applications; if not, it will be reported as 0).

On platforms that support the SIGINFO signal, such as most BSDs (including Mac OS X) and Digital/Tru64 UNIX, it will report those counts when it receives a SIGINFO signal (generated, for example, by typing your ‘‘status’’ character, typically control-T) and will continue capturing packets.

Reading packets from a network interface may require that you have special privileges:

You must have read access to

/dev/bpf .

Reading a saved packet file doesn't require special privileges.

OPTIONS

−A Print each packet (minus its link level header) in ASCII. Handy for capturing web pages.

−a Attempt to convert network and broadcast addresses to names.

−c Exit after receiving *count* packets.

−C Before writing a raw packet to a savefile, check whether the file is currently larger than *file_size* and, if so, close the current savefile and open a new one. Savefiles after the first savefile will have the name specified with the **−w** flag, with a number after it, starting at 1 and continuing upward. The units of *file_size* are millions of bytes (1,000,000 bytes, not 1,048,576 bytes).

−d Dump the compiled packet-matching code in a human readable form to standard output and stop.

−dd Dump packet-matching code as a **C** program fragment.

−ddd Dump packet-matching code as decimal numbers (preceded with a count).

−D Print the list of the network interfaces available on the system and on which *tcpdump* can capture packets. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the **−i** flag to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (e.g., Windows systems, or UNIX systems lacking **ifconfig −a**); the number can be useful on Windows 2000 and later systems, where the interface name is a somewhat complex string.

The **−D** flag will not be supported if *tcpdump* was built with an older version of *libpcap* that lacks the **pcap_findalldevs()** function.

−e Print the link-level header on each dump line.

−E Use *spi@ipaddr algo:secret* for decrypting IPsec ESP packets that are addressed to *addr* and contain Security Parameter Index value *spi*. This combination may be repeated with comma or newline seperation.

Note that setting the secret for IPv4 ESP packets is supported at this time.

Algorithms may be **des-cbc**, **3des-cbc**, **blowfish-cbc**, **rc3-cbc**, **cast128-cbc**, or **none**. The default is **des-cbc**. The ability to decrypt packets is only present if *tcpdump* was compiled with cryptography enabled.

secret is the ASCII text for ESP secret key. If preceded by 0x, then a hex value will be read.

The option assumes RFC2406 ESP, not RFC1827 ESP. The option is only for debugging purposes, and the use of this option with a true 'secret' key is discouraged. By presenting IPsec secret key onto command line you make it visible to others, via *ps*(1) and other occasions.

In addition to the above syntax, the syntax *file name* may be used to have tcpdump read the provided file in. The file is opened upon receiving the first ESP packet, so any special permissions that tcpdump may have been given should already have been given up.

−f Print ‘foreign’ IPv4 addresses numerically rather than symbolically (this option is intended to get around serious brain damage in Sun’s NIS server — usually it hangs forever translating non-local internet numbers).

The test for ‘foreign’ IPv4 addresses is done using the IPv4 address and netmask of the interface on which capture is being done. If that address or netmask are not available, available, either because the interface on which capture is being done has no address or netmask or because the capture is being done on the Linux "any" interface, which can capture on more than one interface, this option will not work correctly.

−F Use *file* as input for the filter expression. An additional expression given on the command line is ignored.

−i Listen on *interface*. If unspecified, *tcpdump* searches the system interface list for the lowest numbered, configured up interface (excluding loopback). Ties are broken by choosing the earliest match.

If the **−D** flag is supported, an interface number as printed by that flag can be used as the *interface* argument.

−l Make stdout line buffered. Useful if you want to see the data while capturing it. E.g.,
‘‘tcpdump −l | tee dat’’ or ‘‘tcpdump −l > dat & tail −f dat’’.

−L List the known data link types for the interface and exit.

−m Load SMI MIB module definitions from file *module*. This option can be used several times to load several MIB modules into *tcpdump*.

−M Use *secret* as a shared secret for validating the digests found in TCP segments with the TCP-MD5 option (RFC 2385), if present.

−n Don’t convert addresses (i.e., host addresses, port numbers, etc.) to names.

−N Don’t print domain name qualification of host names. E.g., if you give this flag then *tcpdump* will print ‘‘nic’’ instead of ‘‘nic.ddn.mil’’.

−O Do not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer.

−p *Don’t* put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, ‘-p’ cannot be used as an abbreviation for ‘ether host {local-hw-addr} or ether broadcast’.

−q Quick (quiet?) output. Print less protocol information so output lines are shorter.

−R Assume ESP/AH packets to be based on old specification (RFC1825 to RFC1829). If specified, *tcpdump* will not print replay prevention field. Since there is no protocol version field in ESP/AH specification, *tcpdump* cannot deduce the version of ESP/AH protocol.

−r Read packets from *file* (which was created with the **−w** option). Standard input is used if *file* is ‘‘-’’.

−S Print absolute, rather than relative, TCP sequence numbers.

−s Snarf *snaplen* bytes of data from each packet rather than the default of 68 (with SunOS’s NIT, the minimum is actually 96). 68 bytes is adequate for IP, ICMP, TCP and UDP but may truncate protocol information from name server and NFS packets (see below). Packets truncated because of a limited snapshot are indicated in the output with ‘‘[|*proto*]’’, where *proto* is the name of the protocol level at which the truncation has occurred. Note that taking larger snapshots both increases the amount of time it takes to process packets and, effectively, decreases the amount of packet buffering. This may cause packets to be lost. You should limit *snaplen* to the smallest number that will capture the protocol information you’re interested in. Setting *snaplen* to 0 means use the required length to catch whole packets.

−T Force packets selected by "*expression*" to be interpreted the specified *type*. Currently known types are **aodv** (Ad-hoc On-demand Distance Vector protocol), **cnfp** (Cisco NetFlow protocol), **rpc** (Remote Procedure Call), **rtp** (Real-Time Applications protocol), **rtcp** (Real-Time Applications control protocol), **snmp** (Simple Network Management Protocol), **tftp** (Trivial File Transfer Protocol), **vat** (Visual Audio Tool), and **wb** (distributed White Board).

−t *Don't* print a timestamp on each dump line.

−tt Print an unformatted timestamp on each dump line.

−ttt Print a delta (in micro-seconds) between current and previous line on each dump line.

−tttt Print a timestamp in default format proceeded by date on each dump line.

−u Print undecoded NFS handles.

−U Make output saved via the **−w** option ‘‘packet-buffered’’; i.e., as each packet is saved, it will be written to the output file, rather than being written only when the output buffer fills.

The **−U** flag will not be supported if *tcpdump* was built with an older version of *libpcap* that lacks the **pcap_dump_flush()** function.

−v When parsing and printing, produce (slightly more) verbose output. For example, the time to live, identification, total length and options in an IP packet are printed. Also enables additional packet integrity checks such as verifying the IP and ICMP header checksum.

When writing to a file with the **−w** option, report, every 10 seconds, the number of packets captured.

−vv Even more verbose output. For example, additional fields are printed from NFS reply packets, and SMB packets are fully decoded.

−vvv Even more verbose output. For example, telnet **SB** ... **SE** options are printed in full. With **−X** Telnet options are printed in hex as well.

−w Write the raw packets to *file* rather than parsing and printing them out. They can later be printed with the −r option. Standard output is used if *file* is ‘‘-’’.

−W Used in conjunction with the **−C** option, this will limit the number of files created to the specified number, and begin overwriting files from the beginning, thus creating a ’rotating’ buffer. In addition, it will name the files with enough leading 0s to support the maximum number of files, allowing them to sort correctly.

−x When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex. The smaller of the entire packet or *snaplen* bytes will be printed. Note that this is the entire link-layer packet, so for link layers that pad (e.g. Ethernet), the padding bytes will also be printed when the higher layer packet is shorter than the required padding.

−xx When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, *including* its link level header, in hex.

−X When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and ASCII. This is very handy for analysing new protocols.

−XX When parsing and printing, in addition to printing the headers of each packet, print the data of each packet, *including* its link level header, in hex and ASCII.

−y Set the data link type to use while capturing packets to *datalinktype*. The available data link types may be found using the −L option.

−Z Drops privileges (if root) and changes user ID to *user* and the group ID to the primary group of *user*.

This behavior can also be enabled by default at compile time.

expression

selects which packets will be dumped. If no *expression* is given, all packets on the net will be dumped. Otherwise, only packets for which *expression* is 'true' will be dumped.

The *expression* consists of one or more *primitives.* Primitives usually consist of an *id* (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

type qualifiers say what kind of thing the id name or number refers to. Possible types are **host**, **net , port** and **portrange**. E.g., 'host foo', 'net 128.3', 'port 20', 'portrange 6000-6008'. If there is no type qualifier, **host** is assumed.

dir qualifiers specify a particular transfer direction to and/or from *id*. Possible directions are **src**, **dst**, **src or dst** and **src and dst**. E.g., 'src foo', 'dst net 128.3', 'src or dst port ftp-data'. If there is no dir qualifier, **src or dst** is assumed. For some link layers, such as SLIP and the ''cooked'' Linux capture mode used for the ''any'' device and for some other device types, the **inbound** and **outbound** qualifiers can be used to specify a desired direction.

proto qualifiers restrict the match to a particular protocol. Possible protos are: **ether**, **fddi**, **tr**, **wlan**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp** and **udp**. E.g., 'ether src foo', 'arp net 128.3', 'tcp port 21', 'udp portrange 7000-7009'. If there is no proto qualifier, all protocols consistent with the type are assumed. E.g., 'src foo' means '(ip or arp or rarp) src foo' (except the latter is not legal syntax), 'net bar' means '(ip or arp or rarp) net bar' and 'port 53' means '(tcp or udp) port 53'.

['fddi' is actually an alias for 'ether'; the parser treats them identically as meaning ''the data link level used on the specified network interface.'' FDDI headers contain Ethernet-like source and destination addresses, and often contain Ethernet-like packet types, so you can filter on these FDDI fields just as with the analogous Ethernet fields. FDDI headers also contain other fields, but you cannot name them explicitly in a filter expression.

Similarly, 'tr' and 'wlan' are aliases for 'ether'; the previous paragraph's statements about FDDI headers also apply to Token Ring and 802.11 wireless LAN headers. For 802.11 headers, the destination address is the DA field and the source address is the SA field; the BSSID, RA, and TA fields aren't tested.]

In addition to the above, there are some special 'primitive' keywords that don't follow the pattern: **gateway**, **broadcast**, **less**, **greater** and arithmetic expressions. All of these are described below.

More complex filter expressions are built up by using the words **and**, **or** and **not** to combine primitives. E.g., 'host foo and not port ftp and not port ftp-data'. To save typing, identical qualifier lists can be omitted. E.g., 'tcp dst port ftp or ftp-data or domain' is exactly the same as 'tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain'.

Allowable primitives are:

dst host *host*

True if the IPv4/v6 destination field of the packet is *host*, which may be either an address or a name.

src host *host*

True if the IPv4/v6 source field of the packet is *host*.

host *host*

True if either the IPv4/v6 source or destination of the packet is *host*.

Any of the above host expressions can be prepended with the keywords, **ip**, **arp**, **rarp**, or **ip6** as in:

ip host *host*

which is equivalent to:

ether proto *\ip* **and host** *host*

If *host* is a name with multiple IP addresses, each address will be checked for a match.

ether dst *ehost*
True if the Ethernet destination address is *ehost*. *Ehost* may be either a name from /etc/ethers or a number (see *ethers*(3N) for numeric format).

ether src *ehost*
True if the Ethernet source address is *ehost*.

ether host *ehost*
True if either the Ethernet source or destination address is *ehost*.

gateway *host*
True if the packet used *host* as a gateway. I.e., the Ethernet source or destination address was *host* but neither the IP source nor the IP destination was *host*. *Host* must be a name and must be found both by the machine's host-name-to-IP-address resolution mechanisms (host name file, DNS, NIS, etc.) and by the machine's host-name-to-Ethernet-address resolution mechanism (/etc/ethers, etc.). (An equivalent expression is

ether host *ehost* **and not host** *host*

which can be used with either names or numbers for *host / ehost*.) This syntax does not work in IPv6-enabled configuration at this moment.

dst net *net*
True if the IPv4/v6 destination address of the packet has a network number of *net*. *Net* may be either a name from the networks database (/etc/networks, etc.) or a network number. An IPv4 network number can be written as a dotted quad (e.g., 192.168.1.0), dotted triple (e.g., 192.168.1), dotted pair (e.g, 172.16), or single number (e.g., 10); the netmask is 255.255.255.255 for a dotted quad (which means that it's really a host match), 255.255.255.0 for a dotted triple, 255.255.0.0 for a dotted pair, or 255.0.0.0 for a single number. An IPv6 network number must be written out fully; the netmask is ff:ff:ff:ff:ff:ff:ff:ff, so IPv6 "network" matches are really always host matches, and a network match requires a netmask length.

src net *net*
True if the IPv4/v6 source address of the packet has a network number of *net*.

net *net* True if either the IPv4/v6 source or destination address of the packet has a network number of *net*.

net *net* **mask** *netmask*
True if the IPv4 address matches *net* with the specific *netmask*. May be qualified with **src** or **dst**. Note that this syntax is not valid for IPv6 *net*.

net *net*/*len*
True if the IPv4/v6 address matches *net* with a netmask *len* bits wide. May be qualified with **src** or **dst**.

dst port *port*
True if the packet is ip/tcp, ip/udp, ip6/tcp or ip6/udp and has a destination port value of *port*. The *port* can be a number or a name used in /etc/services (see *tcp*(4P) and *udp*(4P)). If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (e.g., **dst port 513** will print both tcp/login traffic and udp/who traffic, and **port domain** will print both tcp/domain and udp/domain traffic).

src port *port*
True if the packet has a source port value of *port*.

port *port*
True if either the source or destination port of the packet is *port*.

dst portrange *port1*-*port2*
True if the packet is ip/tcp, ip/udp, ip6/tcp or ip6/udp and has a destination port value between *port1* and *port2*. *port1* and *port2* are interpreted in the same fashion as the *port* parameter for **port**.

src portrange *port1*-*port2*
True if the packet has a source port value between *port1* and *port2*.

portrange *port1*-*port2*
True if either the source or destination port of the packet is between *port1* and *port2*.

Any of the above port or port range expressions can be prepended with the keywords, **tcp** or **udp**, as in:
tcp src port *port*
which matches only tcp packets whose source port is *port*.

less *length*
True if the packet has a length less than or equal to *length*. This is equivalent to:
len <= *length***.**

greater *length*
True if the packet has a length greater than or equal to *length*. This is equivalent to:
len >= *length***.**

ip proto *protocol*
True if the packet is an IPv4 packet (see *ip*(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names **icmp**, **icmp6**, **igmp**, **igrp**, **pim**, **ah**, **esp**, **vrrp**, **udp**, or **tcp**. Note that the identifiers **tcp**, **udp**, and **icmp** are also keywords and must be escaped via backslash (\), which is \\ in the C-shell. Note that this primitive does not chase the protocol header chain.

ip6 proto *protocol*
True if the packet is an IPv6 packet of protocol type *protocol*. Note that this primitive does not chase the protocol header chain.

ip6 protochain *protocol*
True if the packet is IPv6 packet, and contains protocol header with type *protocol* in its protocol header chain. For example,
ip6 protochain 6
matches any IPv6 packet with TCP protocol header in the protocol header chain. The packet may contain, for example, authentication header, routing header, or hop-by-hop option header, between IPv6 header and TCP header. The BPF code emitted by this primitive is complex and cannot be optimized by BPF optimizer code in *tcpdump*, so this can be somewhat slow.

ip protochain *protocol*
Equivalent to **ip6 protochain** *protocol*, but this is for IPv4.

ether broadcast
True if the packet is an Ethernet broadcast packet. The *ether* keyword is optional.

ip broadcast
True if the packet is an IPv4 broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the subnet mask on the interface on which the capture is being done.

If the subnet mask of the interface on which the capture is being done is not available, either because the interface on which capture is being done has no netmask or because the capture is being done on the Linux "any" interface, which can capture on more than one interface, this check will not work correctly.

ether multicast
True if the packet is an Ethernet multicast packet. The **ether** keyword is optional. This is shorthand for '**ether[0] & 1 != 0**'.

ip multicast
True if the packet is an IPv4 multicast packet.

ip6 multicast
True if the packet is an IPv6 multicast packet.

ether proto *protocol*
True if the packet is of ether type *protocol*. *Protocol* can be a number or one of the names **ip**, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **sca**, **lat**, **mopdl**, **moprc**, **iso**, **stp**, **ipx**, or **netbeui**. Note these identifiers are also keywords and must be escaped via backslash (\).

[In the case of FDDI (e.g., '**fddi protocol arp**'), Token Ring (e.g., '**tr protocol arp**'), and IEEE 802.11 wireless LANS (e.g., '**wlan protocol arp**'), for most of those protocols, the protocol identification comes from the 802.2 Logical Link Control (LLC) header, which is usually layered on top of the FDDI, Token Ring, or 802.11 header.

When filtering for most protocol identifiers on FDDI, Token Ring, or 802.11, *tcpdump* checks only the protocol ID field of an LLC header in so-called SNAP format with an Organizational Unit Identifier (OUI) of 0x000000, for encapsulated Ethernet; it doesn't check whether the packet is in SNAP format with an OUI of 0x000000. The exceptions are:

iso *tcpdump* checks the DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) fields of the LLC header;

stp and **netbeui**
tcpdump checks the DSAP of the LLC header;

atalk *tcpdump* checks for a SNAP-format packet with an OUI of 0x080007 and the AppleTalk etype.

In the case of Ethernet, *tcpdump* checks the Ethernet type field for most of those protocols. The exceptions are:

iso, **stp**, and **netbeui**
tcpdump checks for an 802.3 frame and then checks the LLC header as it does for FDDI, Token Ring, and 802.11;

atalk *tcpdump* checks both for the AppleTalk etype in an Ethernet frame and for a SNAP-format packet as it does for FDDI, Token Ring, and 802.11;

aarp *tcpdump* checks for the AppleTalk ARP etype in either an Ethernet frame or an 802.2 SNAP frame with an OUI of 0x000000;

ipx *tcpdump* checks for the IPX etype in an Ethernet frame, the IPX DSAP in the LLC header, the 802.3-with-no-LLC-header encapsulation of IPX, and the IPX etype in a SNAP frame.

decnet src *host*
True if the DECNET source address is *host*, which may be an address of the form "10.123", or a DECNET host name. [DECNET host name support is only available on ULTRIX systems that are configured to run DECNET.]

decnet dst *host*
True if the DECNET destination address is *host*.

decnet host *host*
True if either the DECNET source or destination address is *host*.

ifname *interface*
True if the packet was logged as coming from the specified interface (applies only to packets logged by OpenBSD's **pf**(4)).

on *interface*
Synonymous with the **ifname** modifier.

rnr *num*
True if the packet was logged as matching the specified PF rule number (applies only to packets logged by OpenBSD's **pf**(4)).

rulenum *num*
Synonomous with the **rnr** modifier.

reason *code*
True if the packet was logged with the specified PF reason code. The known codes are: **match**, **bad-offset**, **fragment**, **short**, **normalize**, and **memory** (applies only to packets logged by OpenBSD's **pf**(4)).

rset *name*
True if the packet was logged as matching the specified PF ruleset name of an anchored ruleset (applies only to packets logged by **pf**(4)).

ruleset *name*
Synonomous with the **rset** modifier.

srnr *num*
True if the packet was logged as matching the specified PF rule number of an anchored ruleset (applies only to packets logged by **pf**(4)).

subrulenum *num*
Synonomous with the **srnr** modifier.

action *act*
True if PF took the specified action when the packet was logged. Known actions are: **pass** and **block** (applies only to packets logged by OpenBSD's **pf**(4)).

ip, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **iso**, **stp**, **ipx**, *netbeui*
Abbreviations for:
ether proto *p*
where *p* is one of the above protocols.

lat, **moprc**, **mopdl**
Abbreviations for:
ether proto *p*
where *p* is one of the above protocols. Note that *tcpdump* does not currently know how to parse these protocols.

vlan *[vlan_id]*
True if the packet is an IEEE 802.1Q VLAN packet. If *[vlan_id]* is specified, only true if the packet has the specified *vlan_id*. Note that the first **vlan** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a VLAN packet. The **vlan** *[vlan_id]* expression may be used more than once, to filter on VLAN hierarchies. Each use of that expression increments the filter offsets by 4.

For example:
vlan 100 && vlan 200
filters on VLAN 200 encapsulated within VLAN 100, and
vlan && vlan 300 && ip
filters IPv4 protocols encapsulated in VLAN 300 encapsulated within any higher order VLAN.

mpls *[label_num]*

True if the packet is an MPLS packet. If *[label_num]* is specified, only true is the packet has the specified *label_num*. Note that the first **mpls** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a MPLS-encapsulated IP packet. The **mpls** *[label_num]* expression may be used more than once, to filter on MPLS hierarchies. Each use of that expression increments the filter offsets by 4.

For example:

mpls 100000 && mpls 1024

filters packets with an outer label of 100000 and an inner label of 1024, and

mpls && mpls 1024 && host 192.9.200.1

filters packets to or from 192.9.200.1 with an inner label of 1024 and any outer label.

pppoed

True if the packet is a PPP-over-Ethernet Discovery packet (Ethernet type 0x8863).

pppoes True if the packet is a PPP-over-Ethernet Session packet (Ethernet type 0x8864). Note that the first **pppoes** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a PPPoE session packet.

For example:

pppoes && ip

filters IPv4 protocols encapsulated in PPPoE.

tcp, **udp**, **icmp**

Abbreviations for:

ip proto *p* **or ip6 proto** *p*

where *p* is one of the above protocols.

iso proto *protocol*

True if the packet is an OSI packet of protocol type *protocol*. *Protocol* can be a number or one of the names **clnp**, **esis**, or **isis**.

clnp, **esis**, **isis**

Abbreviations for:

iso proto *p*

where *p* is one of the above protocols.

l1, **l2**, **iih**, **lsp**, **snp**, **csnp**, **psnp**

Abbreviations for IS-IS PDU types.

vpi *n* True if the packet is an ATM packet, for SunATM on Solaris, with a virtual path identifier of *n*.

vci *n* True if the packet is an ATM packet, for SunATM on Solaris, with a virtual channel identifier of *n*.

lane True if the packet is an ATM packet, for SunATM on Solaris, and is an ATM LANE packet. Note that the first **lane** keyword encountered in *expression* changes the tests done in the remainder of *expression* on the assumption that the packet is either a LANE emulated Ethernet packet or a LANE LE Control packet. If **lane** isn't specified, the tests are done under the assumption that the packet is an LLC-encapsulated packet.

llc True if the packet is an ATM packet, for SunATM on Solaris, and is an LLC-encapsulated packet.

oamf4s True if the packet is an ATM packet, for SunATM on Solaris, and is a segment OAM F4 flow cell (VPI=0 & VCI=3).

oamf4e

True if the packet is an ATM packet, for SunATM on Solaris, and is an end-to-end OAM F4 flow cell (VPI=0 & VCI=4).

oamf4 True if the packet is an ATM packet, for SunATM on Solaris, and is a segment or end-to-end OAM F4 flow cell (VPI=0 & (VCI=3 | VCI=4)).

oam True if the packet is an ATM packet, for SunATM on Solaris, and is a segment or end-to-end OAM F4 flow cell (VPI=0 & (VCI=3 | VCI=4)).

metac True if the packet is an ATM packet, for SunATM on Solaris, and is on a meta signaling circuit (VPI=0 & VCI=1).

bcc True if the packet is an ATM packet, for SunATM on Solaris, and is on a broadcast signaling circuit (VPI=0 & VCI=2).

sc True if the packet is an ATM packet, for SunATM on Solaris, and is on a signaling circuit (VPI=0 & VCI=5).

ilmic True if the packet is an ATM packet, for SunATM on Solaris, and is on an ILMI circuit (VPI=0 & VCI=16).

connectmsg
True if the packet is an ATM packet, for SunATM on Solaris, and is on a signaling circuit and is a Q.2931 Setup, Call Proceeding, Connect, Connect Ack, Release, or Release Done message.

metaconnect
True if the packet is an ATM packet, for SunATM on Solaris, and is on a meta signaling circuit and is a Q.2931 Setup, Call Proceeding, Connect, Release, or Release Done message.

expr relop expr
True if the relation holds, where *relop* is one of >, <, >=, <=, =, !=, and *expr* is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+, -, *, /, &, |, <<, >>], a length operator, and special packet data accessors. Note that all comparisons are unsigned, so that, for example, 0x80000000 and 0xffffffff are > 0. To access data inside the packet, use the following syntax:
proto **[** *expr* **:** *size* **]**
Proto is one of **ether, fddi, tr, wlan, ppp, slip, link, ip, arp, rarp, tcp, udp, icmp, ip6** or **radio**, and indicates the protocol layer for the index operation. (**ether, fddi, wlan, tr, ppp, slip** and **link** all refer to the link layer. **radio** refers to the "radio header" added to some 802.11 captures.) Note that *tcp, udp* and other upper-layer protocol types only apply to IPv4, not IPv6 (this will be fixed in the future). The byte offset, relative to the indicated protocol layer, is given by *expr*. *Size* is optional and indicates the number of bytes in the field of interest; it can be either one, two, or four, and defaults to one. The length operator, indicated by the keyword **len**, gives the length of the packet.

For example, '**ether[0] & 1 != 0**' catches all multicast traffic. The expression '**ip[0] & 0xf != 5**' catches all IPv4 packets with options. The expression '**ip[6:2] & 0x1fff = 0**' catches only unfragmented IPv4 datagrams and frag zero of fragmented IPv4 datagrams. This check is implicitly applied to the **tcp** and **udp** index operations. For instance, **tcp[0]** always means the first byte of the TCP *header*, and never means the first byte of an intervening fragment.

Some offsets and field values may be expressed as names rather than as numeric values. The following protocol header field offsets are available: **icmptype** (ICMP type field), **icmpcode** (ICMP code field), and **tcpflags** (TCP flags field).

The following ICMP type field values are available: **icmp-echoreply**, **icmp-unreach**, **icmp-sourcequench**, **icmp-redirect**, **icmp-echo**, **icmp-routeradvert**, **icmp-routersolicit**, **icmp-timxceed**, **icmp-paramprob**, **icmp-tstamp**, **icmp-tstampreply**, **icmp-ireq**, **icmp-ireqreply**, **icmp-maskreq**, **icmp-maskreply**.

The following TCP flags field values are available: **tcp-fin**, **tcp-syn**, **tcp-rst**, **tcp-push**, **tcp-ack**, **tcp-urg**.

Primitives may be combined using:

A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

Negation (‘**!**’ or ‘**not**’).

Concatenation (‘**&&**’ or ‘**and**’).

Alternation (‘**||**’ or ‘**or**’).

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit **and** tokens, not juxtaposition, are now required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example,
not host vs and ace
is short for
not host vs and host ace
which should not be confused with
not (host vs or ace)

Expression arguments can be passed to *tcpdump* as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

EXAMPLES

To print all packets arriving at or departing from *sundown*:
tcpdump host sundown

To print traffic between *helios* and either *hot* or *ace*:
tcpdump host helios and \(hot or ace \)

To print all IP packets between *ace* and any host except *helios*:
tcpdump ip host ace and not helios

To print all traffic between local hosts and hosts at Berkeley:
tcpdump net ucb-ether

To print all ftp traffic through internet gateway *snup*: (note that the expression is quoted to prevent the shell from (mis-)interpreting the parentheses):
tcpdump ’gateway snup and (port ftp or ftp-data)’

To print traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this stuff should never make it onto your local net).
tcpdump ip and not net *localnet*

To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host.
tcpdump ’tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net *localnet***’**

To print all IPv4 HTTP packets to and from port 80, i.e. print only packets that contain data, not, for example, SYN and FIN packets and ACK-only packets. (IPv6 is left as an exercise for the reader.)
tcpdump ’tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)’

To print IP packets longer than 576 bytes sent through gateway *snup*:
tcpdump ’gateway snup and ip[2:2] > 576’

To print IP broadcast or multicast packets that were *not* sent via Ethernet broadcast or multicast:
tcpdump ’ether[0] & 1 = 0 and ip[16] >= 224’

To print all ICMP packets that are not echo requests/replies (i.e., not ping packets):

tcpdump ’icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply’

OUTPUT FORMAT

The output of *tcpdump* is protocol dependent. The following gives a brief description and examples of most of the formats.

Link Level Headers

If the ’-e’ option is given, the link level header is printed out. On Ethernets, the source and destination addresses, protocol, and packet length are printed.

On FDDI networks, the ’-e’ option causes *tcpdump* to print the ‘frame control’ field, the source and destination addresses, and the packet length. (The ‘frame control’ field governs the interpretation of the rest of the packet. Normal packets (such as those containing IP datagrams) are ‘async’ packets, with a priority value between 0 and 7; for example, ‘**async4**’. Such packets are assumed to contain an 802.2 Logical Link Control (LLC) packet; the LLC header is printed if it is *not* an ISO datagram or a so-called SNAP packet.

On Token Ring networks, the ’-e’ option causes *tcpdump* to print the ‘access control’ and ‘frame control’ fields, the source and destination addresses, and the packet length. As on FDDI networks, packets are assumed to contain an LLC packet. Regardless of whether the ’-e’ option is specified or not, the source routing information is printed for source-routed packets.

On 802.11 networks, the ’-e’ option causes *tcpdump* to print the ‘frame control’ fields, all of the addresses in the 802.11 header, and the packet length. As on FDDI networks, packets are assumed to contain an LLC packet.

(N.B.: The following description assumes familiarity with the SLIP compression algorithm described in RFC-1144.)

On SLIP links, a direction indicator (‘‘I’’ for inbound, ‘‘O’’ for outbound), packet type, and compression information are printed out. The packet type is printed first. The three types are *ip*, *utcp*, and *ctcp*. No further link information is printed for *ip* packets. For TCP packets, the connection identifier is printed following the type. If the packet is compressed, its encoded header is printed out. The special cases are printed out as ***S+***n* and ***SA+***n*, where *n* is the amount by which the sequence number (or sequence number and ack) has changed. If it is not a special case, zero or more changes are printed. A change is indicated by U (urgent pointer), W (window), A (ack), S (sequence number), and I (packet ID), followed by a delta (+n or -n), or a new value (=n). Finally, the amount of data in the packet and compressed header length are printed.

For example, the following line shows an outbound compressed TCP packet, with an implicit connection identifier; the ack has changed by 6, the sequence number by 49, and the packet ID by 6; there are 3 bytes of data and 6 bytes of compressed header:

O ctcp * A+6 S+49 I+6 3 (6)

ARP/RARP Packets

Arp/rarp output shows the type of request and its arguments. The format is intended to be self explanatory. Here is a short sample taken from the start of an ‘rlogin’ from host *rtsg* to host *csam*:

```
arp who-has csam tell rtsg
arp reply csam is-at CSAM
```

The first line says that rtsg sent an arp packet asking for the Ethernet address of internet host csam. Csam replies with its Ethernet address (in this example, Ethernet addresses are in caps and internet addresses in lower case).

This would look less redundant if we had done *tcpdump −n*:

```
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at 02:07:01:00:01:c4
```

If we had done *tcpdump −e*, the fact that the first packet is broadcast and the second is point-to-point would be visible:

```
RTSG Broadcast 0806  64: arp who-has csam tell rtsg
CSAM RTSG 0806  64: arp reply csam is-at CSAM
```

For the first packet this says the Ethernet source address is RTSG, the destination is the Ethernet broadcast address, the type field contained hex 0806 (type ETHER_ARP) and the total length was 64 bytes.

TCP Packets

(N.B.:The following description assumes familiarity with the TCP protocol described in RFC-793. If you are not familiar with the protocol, neither this description nor tcpdump will be of much use to you.)

The general format of a tcp protocol line is:

src > dst: flags data-seqno ack window urgent options

Src and *dst* are the source and destination IP addresses and ports. *Flags* are some combination of S (SYN), F (FIN), P (PUSH), R (RST), W (ECN CWR) or E (ECN-Echo), or a single '.' (no flags). *Data-seqno* describes the portion of sequence space covered by the data in this packet (see example below). *Ack* is sequence number of the next data expected the other direction on this connection. *Window* is the number of bytes of receive buffer space available the other direction on this connection. *Urg* indicates there is 'urgent' data in the packet. *Options* are tcp options enclosed in angle brackets (e.g., <mss 1024>).

Src, dst and *flags* are always present. The other fields depend on the contents of the packet's tcp protocol header and are output only if appropriate.

Here is the opening portion of an rlogin from host *rtsg* to host *csam*.

```
rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1
```

The first line says that tcp port 1023 on rtsg sent a packet to port *login* on csam. The **S** indicates that the *SYN* flag was set. The packet sequence number was 768512 and it contained no data. (The notation is 'first:last(nbytes)' which means 'sequence numbers *first* up to but not including *last* which is *nbytes* bytes of user data'.) There was no piggy-backed ack, the available receive window was 4096 bytes and there was a max-segment-size option requesting an mss of 1024 bytes.

Csam replies with a similar packet except it includes a piggy-backed ack for rtsg's SYN. Rtsg then acks csam's SYN. The '.' means no flags were set. The packet contained no data so there is no data sequence number. Note that the ack sequence number is a small integer (1). The first time *tcpdump* sees a tcp 'conversation', it prints the sequence number from the packet. On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is printed. This means that sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being '1'). '-S' will override this feature, causing the original sequence numbers to be output.

On the 6th line, rtsg sends csam 19 bytes of data (bytes 2 through 20 in the rtsg → csam side of the conversation). The PUSH flag is set in the packet. On the 7th line, csam says it's received data sent by rtsg up to but not including byte 21. Most of this data is apparently sitting in the socket buffer since csam's receive window has gotten 19 bytes smaller. Csam also sends one byte of data to rtsg in this packet. On the 8th and 9th lines, csam sends two bytes of urgent, pushed data to rtsg.

If the snapshot was small enough that *tcpdump* didn't capture the full TCP header, it interprets as much of the header as it can and then reports ‘‘[|*tcp*]’’ to indicate the remainder could not be interpreted. If the header contains a bogus option (one with a length that’s either too small or beyond the end of the header), *tcpdump* reports it as ‘‘[*bad opt*]’’ and does not interpret any further options (since it’s impossible to tell where they start). If the header length indicates options are present but the IP datagram length is not long enough for the options to actually be there, *tcpdump* reports it as ‘‘[*bad hdr length*]’’.

Capturing TCP packets with particular flag combinations (SYN-ACK, URG-ACK, etc.)

There are 8 bits in the control bits section of the TCP header:

CWR | ECE | URG | ACK | PSH | RST | SYN | FIN

Let’s assume that we want to watch packets used in establishing a TCP connection. Recall that TCP uses a 3-way handshake protocol when it initializes a new connection; the connection sequence with regard to the TCP control bits is

```
1) Caller sends SYN
2) Recipient responds with SYN, ACK
3) Caller sends ACK
```

Now we’re interested in capturing packets that have only the SYN bit set (Step 1). Note that we don’t want packets from step 2 (SYN-ACK), just a plain initial SYN. What we need is a correct filter expression for *tcpdump*.

Recall the structure of a TCP header without options:

```
 0                            15                              31
-----------------------------------------------------------------
|          source port          |       destination port        |
-----------------------------------------------------------------
|                        sequence number                        |
-----------------------------------------------------------------
|                     acknowledgment number                     |
-----------------------------------------------------------------
|  HL   | rsvd  |C|E|U|A|P|R|S|F|        window size            |
-----------------------------------------------------------------
|         TCP checksum          |       urgent pointer          |
-----------------------------------------------------------------
```

A TCP header usually holds 20 octets of data, unless options are present. The first line of the graph contains octets 0 - 3, the second line shows octets 4 - 7 etc.

Starting to count with 0, the relevant TCP control bits are contained in octet 13:

```
 0             7|             15|             23|             31
----------------|---------------|---------------|----------------
|  HL   | rsvd  |C|E|U|A|P|R|S|F|        window size            |
----------------|---------------|---------------|----------------
|               |  13th octet   |               |               |
```

Let’s have a closer look at octet no. 13:

```
|               |
|---------------|
|C|E|U|A|P|R|S|F|
|---------------|
|7   5   3     0|
```

These are the TCP control bits we are interested in. We have numbered the bits in this octet from 0 to 7, right to left, so the PSH bit is bit number 3, while the URG bit is number 5.

Recall that we want to capture packets with only SYN set. Let’s see what happens to octet 13 if a TCP datagram arrives with the SYN bit set in its header:

```
            |C|E|U|A|P|R|S|F|
            |---------------|
            |0 0 0 0 0 0 1 0|
            |---------------|
            |7 6 5 4 3 2 1 0|
```

Looking at the control bits section we see that only bit number 1 (SYN) is set.

Assuming that octet number 13 is an 8-bit unsigned integer in network byte order, the binary value of this octet is

```
            00000010
```

and its decimal representation is

```
   7     6     5     4     3     2     1     0
0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 0*2 + 1*2 + 0*2  =  2
```

We’re almost done, because now we know that if only SYN is set, the value of the 13th octet in the TCP header, when interpreted as a 8-bit unsigned integer in network byte order, must be exactly 2.

This relationship can be expressed as
tcp[13] == 2

We can use this expression as the filter for *tcpdump* in order to watch packets which have only SYN set:
tcpdump -i xl0 tcp[13] == 2

The expression says "let the 13th octet of a TCP datagram have the decimal value 2", which is exactly what we want.

Now, let’s assume that we need to capture SYN packets, but we don’t care if ACK or any other TCP control bit is set at the same time. Let’s see what happens to octet 13 when a TCP datagram with SYN-ACK set arrives:

```
    |C|E|U|A|P|R|S|F|
    |---------------|
    |0 0 0 1 0 0 1 0|
    |---------------|
    |7 6 5 4 3 2 1 0|
```

Now bits 1 and 4 are set in the 13th octet. The binary value of octet 13 is

```
                00010010
```

which translates to decimal

```
   7     6     5     4     3     2     1     0
0*2 + 0*2 + 0*2 + 1*2 + 0*2 + 0*2 + 1*2 + 0*2   = 18
```

Now we can’t just use ’tcp[13] == 18’ in the *tcpdump* filter expression, because that would select only those packets that have SYN-ACK set, but not those with only SYN set. Remember that we don’t care if ACK or any other control bit is set as long as SYN is set.

In order to achieve our goal, we need to logically AND the binary value of octet 13 with some other value to preserve the SYN bit. We know that we want SYN to be set in any case, so we’ll logically AND the value in the 13th octet with the binary value of a SYN:

```
        00010010 SYN-ACK              00000010 SYN
    AND  00000010 (we want SYN)   AND  00000010 (we want SYN)
        --------                      --------
    =    00000010                 =    00000010
```

We see that this AND operation delivers the same result regardless whether ACK or another TCP control bit is set. The decimal representation of the AND value as well as the result of this operation is 2 (binary 00000010), so we know that for packets with SYN set the following relation must hold true:

((value of octet 13) AND (2)) == (2)

This points us to the *tcpdump* filter expression
tcpdump -i xl0 'tcp[13] & 2 == 2'

Note that you should use single quotes or a backslash in the expression to hide the AND ('&') special character from the shell.

UDP Packets

UDP format is illustrated by this rwho packet:

```
actinide.who > broadcast.who: udp 84
```

This says that port *who* on host *actinide* sent a udp datagram to port *who* on host *broadcast*, the Internet broadcast address. The packet contained 84 bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information printed. In particular, Domain Name service requests (RFC-1034/1035) and Sun RPC calls (RFC-1050) to NFS.

UDP Name Server Requests

(N.B.:The following description assumes familiarity with the Domain Service protocol described in RFC-1035. If you are not familiar with the protocol, the following description will appear to be written in greek.)

Name server requests are formatted as

src > dst: id op? flags qtype qclass name (len)

```
h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)
```

Host *h2opolo* asked the domain server on *helios* for an address record (qtype=A) associated with the name *ucbvax.berkeley.edu.* The query id was '3'. The '+' indicates the *recursion desired* flag was set. The query length was 37 bytes, not including the UDP and IP protocol headers. The query operation was the normal one, *Query*, so the op field was omitted. If the op had been anything else, it would have been printed between the '3' and the '+'. Similarly, the qclass was the normal one, *C_IN*, and omitted. Any other qclass would have been printed immediately after the 'A'.

A few anomalies are checked and may result in extra fields enclosed in square brackets: If a query contains an answer, authority records or additional records section, *ancount*, *nscount*, or *arcount* are printed as '[*n*a]', '[*n*n]' or '[*n*au]' where *n* is the appropriate count. If any of the response bits are set (AA, RA or rcode) or any of the 'must be zero' bits are set in bytes two and three, '[b2&3=*x*]' is printed, where *x* is the hex value of header bytes two and three.

UDP Name Server Responses

Name server responses are formatted as

src > dst: id op rcode flags a/n/au type class data (len)

```
helios.domain > h2opolo.1538: 3 3/3/7 A 128.32.137.3 (273)
helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)
```

In the first example, *helios* responds to query id 3 from *h2opolo* with 3 answer records, 3 name server records and 7 additional records. The first answer record is type A (address) and its data is internet address 128.32.137.3. The total size of the response was 273 bytes, excluding UDP and IP headers. The op (Query) and response code (NoError) were omitted, as was the class (C_IN) of the A record.

In the second example, *helios* responds to query 2 with a response code of non-existent domain (NXDomain) with no answers, one name server and no authority records. The '*' indicates that the *authoritative answer* bit was set. Since there were no answers, no type, class or data were printed.

Other flag characters that might appear are '−' (recursion available, RA, *not* set) and '|' (truncated message, TC, set). If the 'question' section doesn't contain exactly one entry, '[*n*q]' is printed.

Note that name server requests and responses tend to be large and the default *snaplen* of 68 bytes may not capture enough of the packet to print. Use the **−s** flag to increase the snaplen if you need to seriously investigate name server traffic. '**−s 128**' has worked well for me.

SMB/CIFS decoding

tcpdump now includes fairly extensive SMB/CIFS/NBT decoding for data on UDP/137, UDP/138 and TCP/139. Some primitive decoding of IPX and NetBEUI SMB data is also done.

By default a fairly minimal decode is done, with a much more detailed decode done if -v is used. Be warned that with -v a single SMB packet may take up a page or more, so only use -v if you really want all the gory details.

For information on SMB packet formats and what all te fields mean see www.cifs.org or the pub/samba/specs/ directory on your favorite samba.org mirror site. The SMB patches were written by Andrew Tridgell (tridge@samba.org).

NFS Requests and Replies

Sun NFS (Network File System) requests and replies are printed as:

src.xid > dst.nfs: len op args
src.nfs > dst.xid: reply stat len op results

```
sushi.6709 > wrl.nfs: 112 readlink fh 21,24/10.73165
wrl.nfs > sushi.6709: reply ok 40 readlink "../var"
sushi.201b > wrl.nfs:
      144 lookup fh 9,74/4096.6878 "xcolors"
wrl.nfs > sushi.201b:
      reply ok 128 lookup fh 9,74/4134.3150
```

In the first line, host *sushi* sends a transaction with id *6709* to *wrl* (note that the number following the src host is a transaction id, *not* the source port). The request was 112 bytes, excluding the UDP and IP headers. The operation was a *readlink* (read symbolic link) on file handle (*fh*) 21,24/10.731657119. (If one is lucky, as in this case, the file handle can be interpreted as a major,minor device number pair, followed by the inode number and generation number.) *Wrl* replies 'ok' with the contents of the link.

In the third line, *sushi* asks *wrl* to lookup the name '*xcolors*' in directory file 9,74/4096.6878. Note that the data printed depends on the operation type. The format is intended to be self explanatory if read in conjunction with an NFS protocol spec.

If the −v (verbose) flag is given, additional information is printed. For example:

```
sushi.1372a > wrl.nfs:
      148 read fh 21,11/12.195 8192 bytes @ 24576
wrl.nfs > sushi.1372a:
      reply ok 1472 read REG 100664 ids 417/0 sz 29388
```

(−v also prints the IP header TTL, ID, length, and fragmentation fields, which have been omitted from this example.) In the first line, *sushi* asks *wrl* to read 8192 bytes from file 21,11/12.195, at byte offset 24576. *Wrl* replies 'ok'; the packet shown on the second line is the first fragment of the reply, and hence is only

1472 bytes long (the other bytes will follow in subsequent fragments, but these fragments do not have NFS or even UDP headers and so might not be printed, depending on the filter expression used). Because the −v flag is given, some of the file attributes (which are returned in addition to the file data) are printed: the file type (‘‘REG’’, for regular file), the file mode (in octal), the uid and gid, and the file size.

If the −v flag is given more than once, even more details are printed.

Note that NFS requests are very large and much of the detail won’t be printed unless *snaplen* is increased. Try using ‘**−s 192**’ to watch NFS traffic.

NFS reply packets do not explicitly identify the RPC operation. Instead, *tcpdump* keeps track of ‘‘recent’’ requests, and matches them to the replies using the transaction ID. If a reply does not closely follow the corresponding request, it might not be parsable.

AFS Requests and Replies

Transarc AFS (Andrew File System) requests and replies are printed as:

src.sport > dst.dport: rx packet-type
src.sport > dst.dport: rx packet-type service call call-name args
src.sport > dst.dport: rx packet-type service reply call-name args

```
elvis.7001 > pike.afsfs:
      rx data fs call rename old fid 536876964/1/1 ".newsrc.new"
      new fid 536876964/1/1 ".newsrc"
pike.afsfs > elvis.7001: rx data fs reply rename
```

In the first line, host elvis sends a RX packet to pike. This was a RX data packet to the fs (fileserver) service, and is the start of an RPC call. The RPC call was a rename, with the old directory file id of 536876964/1/1 and an old filename of ‘.newsrc.new’, and a new directory file id of 536876964/1/1 and a new filename of ‘.newsrc’. The host pike responds with a RPC reply to the rename call (which was successful, because it was a data packet and not an abort packet).

In general, all AFS RPCs are decoded at least by RPC call name. Most AFS RPCs have at least some of the arguments decoded (generally only the ‘interesting’ arguments, for some definition of interesting).

The format is intended to be self-describing, but it will probably not be useful to people who are not familiar with the workings of AFS and RX.

If the -v (verbose) flag is given twice, acknowledgement packets and additional header information is printed, such as the the RX call ID, call number, sequence number, serial number, and the RX packet flags.

If the -v flag is given twice, additional information is printed, such as the the RX call ID, serial number, and the RX packet flags. The MTU negotiation information is also printed from RX ack packets.

If the -v flag is given three times, the security index and service id are printed.

Error codes are printed for abort packets, with the exception of Ubik beacon packets (because abort packets are used to signify a yes vote for the Ubik protocol).

Note that AFS requests are very large and many of the arguments won’t be printed unless *snaplen* is increased. Try using ‘**-s 256**’ to watch AFS traffic.

AFS reply packets do not explicitly identify the RPC operation. Instead, *tcpdump* keeps track of ‘‘recent’’ requests, and matches them to the replies using the call number and service ID. If a reply does not closely follow the corresponding request, it might not be parsable.

KIP AppleTalk (DDP in UDP)

AppleTalk DDP packets encapsulated in UDP datagrams are de-encapsulated and dumped as DDP packets (i.e., all the UDP header information is discarded). The file */etc/atalk.names* is used to translate AppleTalk net and node numbers to names. Lines in this file have the form

number name

```
1.254       ether
16.1        icsd-net
1.254.110   ace
```

The first two lines give the names of AppleTalk networks. The third line gives the name of a particular host (a host is distinguished from a net by the 3rd octet in the number − a net number *must* have two octets and a host number *must* have three octets.) The number and name should be separated by whitespace (blanks or tabs). The */etc/atalk.names* file may contain blank lines or comment lines (lines starting with a '#').

AppleTalk addresses are printed in the form

net.host.port

```
144.1.209.2 > icsd-net.112.220
office.2 > icsd-net.112.220
jssmag.149.235 > icsd-net.2
```

(If the */etc/atalk.names* doesn't exist or doesn't contain an entry for some AppleTalk host/net number, addresses are printed in numeric form.) In the first example, NBP (DDP port 2) on net 144.1 node 209 is sending to whatever is listening on port 220 of net icsd node 112. The second line is the same except the full name of the source node is known ('office'). The third line is a send from port 235 on net jssmag node 149 to broadcast on the icsd-net NBP port (note that the broadcast address (255) is indicated by a net name with no host number − for this reason it's a good idea to keep node names and net names distinct in /etc/atalk.names).

NBP (name binding protocol) and ATP (AppleTalk transaction protocol) packets have their contents interpreted. Other protocols just dump the protocol name (or number if no name is registered for the protocol) and packet size.

NBP packets are formatted like the following examples:

```
icsd-net.112.220 > jssmag.2: nbp-lkup 190: "=:LaserWriter@*"
jssmag.209.2 > icsd-net.112.220: nbp-reply 190: "RM1140:LaserWriter@*" 250
techpit.2 > icsd-net.112.220: nbp-reply 190: "techpit:LaserWriter@*" 186
```

The first line is a name lookup request for laserwriters sent by net icsd host 112 and broadcast on net jssmag. The nbp id for the lookup is 190. The second line shows a reply for this request (note that it has the same id) from host jssmag.209 saying that it has a laserwriter resource named "RM1140" registered on port 250. The third line is another reply to the same request saying host techpit has laserwriter "techpit" registered on port 186.

ATP packet formatting is demonstrated by the following example:

```
jssmag.209.165 > helios.132: atp-req  12266<0-7> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:0 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:1 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:2 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:4 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:6 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp*12266:7 (512) 0xae040000
jssmag.209.165 > helios.132: atp-req  12266<3,5> 0xae030001
helios.132 > jssmag.209.165: atp-resp 12266:3 (512) 0xae040000
helios.132 > jssmag.209.165: atp-resp 12266:5 (512) 0xae040000
jssmag.209.165 > helios.132: atp-rel  12266<0-7> 0xae030001
jssmag.209.133 > helios.132: atp-req* 12267<0-7> 0xae030002
```

Jssmag.209 initiates transaction id 12266 with host helios by requesting up to 8 packets (the '<0-7>'). The hex number at the end of the line is the value of the 'userdata' field in the request.

Helios responds with 8 512-byte packets. The ':digit' following the transaction id gives the packet sequence number in the transaction and the number in parens is the amount of data in the packet, excluding the atp header. The '*' on packet 7 indicates that the EOM bit was set.

Jssmag.209 then requests that packets 3 & 5 be retransmitted. Helios resends them then jssmag.209 releases the transaction. Finally, jssmag.209 initiates the next request. The '*' on the request indicates that XO ('exactly once') was *not* set.

IP Fragmentation

Fragmented Internet datagrams are printed as

(frag *id***:***size***@***offset***+)**
(frag *id***:***size***@***offset***)**

(The first form indicates there are more fragments. The second indicates this is the last fragment.)

Id is the fragment id. *Size* is the fragment size (in bytes) excluding the IP header. *Offset* is this fragment's offset (in bytes) in the original datagram.

The fragment information is output for each fragment. The first fragment contains the higher level protocol header and the frag info is printed after the protocol info. Fragments after the first contain no higher level protocol header and the frag info is printed after the source and destination addresses. For example, here is part of an ftp from arizona.edu to lbl-rtsg.arpa over a CSNET connection that doesn't appear to handle 576 byte datagrams:

```
arizona.ftp-data > rtsg.1170: . 1024:1332(308) ack 1 win 4096 (frag 595a:328@0+)
arizona > rtsg: (frag 595a:204@328)
rtsg.1170 > arizona.ftp-data: . ack 1536 win 2560
```

There are a couple of things to note here: First, addresses in the 2nd line don't include port numbers. This is because the TCP protocol information is all in the first fragment and we have no idea what the port or sequence numbers are when we print the later fragments. Second, the tcp sequence information in the first line is printed as if there were 308 bytes of user data when, in fact, there are 512 bytes (308 in the first frag and 204 in the second). If you are looking for holes in the sequence space or trying to match up acks with packets, this can fool you.

A packet with the IP *don't fragment* flag is marked with a trailing **(DF)**.

Timestamps

By default, all output lines are preceded by a timestamp. The timestamp is the current clock time in the form

hh:mm:ss.frac

and is as accurate as the kernel's clock. The timestamp reflects the time the kernel first saw the packet. No attempt is made to account for the time lag between when the Ethernet interface removed the packet from the wire and when the kernel serviced the 'new packet' interrupt.

SEE ALSO

bpf(4), pcap(3)

AUTHORS

The original authors are:

Van Jacobson, Craig Leres and Steven McCanne, all of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA.

It is currently being maintained by tcpdump.org.

The current version is available via http:

http://www.tcpdump.org/

The original distribution is available via anonymous ftp:

ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

IPv6/IPsec support is added by WIDE/KAME project. This program uses Eric Young's SSLeay library, under specific configuration.

BUGS

Please send problems, bugs, questions, desirable enhancements, etc. to:

tcpdump-workers@tcpdump.org

Please send source code contributions, etc. to:

patches@tcpdump.org

Some attempt should be made to reassemble IP fragments or, at least to compute the right length for the higher level protocol.

Name server inverse queries are not dumped correctly: the (empty) question section is printed rather than real query in the answer section. Some believe that inverse queries are themselves a bug and prefer to fix the program generating them rather than *tcpdump*.

A packet trace that crosses a daylight savings time change will give skewed time stamps (the time change is ignored).

Filter expressions on fields other than those in Token Ring headers will not correctly handle source-routed Token Ring packets.

Filter expressions on fields other than those in 802.11 headers will not correctly handle 802.11 data packets with both To DS and From DS set.

ip6 proto should chase header chain, but at this moment it does not. **ip6 protochain** is supplied for this behavior.

Arithmetic expression against transport layer headers, like **tcp[0]**, does not work against IPv6 packets. It only looks at IPv4 packets.

NAME

telnetd — DARPA TELNET protocol server

SYNOPSIS

telnetd [**-BeUhkln**] [**-D** *debugmode*] [**-S** *tos*] [**-X** *authtype*] [**-a** *authmode*]
[**-r***lowpty-highpty*] [**-u** *len*] [**-debug**] [**-L** */bin/login*] [**-y**] [*port*]

DESCRIPTION

The **telnetd** command is a server which supports the DARPA standard TELNET virtual terminal protocol. **Telnetd** is normally invoked by the internet server (see inetd(8)) for requests to connect to the TELNET port as indicated by the /etc/services file (see services(5)). The **-debug** option may be used to start up **telnetd** manually, instead of through inetd(8). If started up this way, *port* may be specified to run **telnetd** on an alternate TCP port number.

The **telnetd** command accepts the following options:

-a *authmode*

This option may be used for specifying what mode should be used for authentication. Note that this option is only useful if **telnetd** has been compiled with support for the AUTHENTICATION option. There are several valid values for *authmode*:

debug Turns on authentication debugging code.

user Only allow connections when the remote user can provide valid authentication information to identify the remote user, and is allowed access to the specified account without providing a password.

valid Only allow connections when the remote user can provide valid authentication information to identify the remote user. The login(1) command will provide any additional user verification needed if the remote user is not allowed automatic access to the specified account.

other Only allow connections that supply some authentication information. This option is currently not supported by any of the existing authentication mechanisms, and is thus the same as specifying **-a valid**.

otp Only allow authenticated connections (as with **-a user**) and also logins with one-time passwords (OTPs). This option will call login with an option so that only OTPs are accepted. The user can of course still type secret information at the prompt.

none This is the default state. Authentication information is not required. If no or insufficient authentication information is provided, then the login(1) program will provide the necessary user verification.

off This disables the authentication code. All user verification will happen through the login(1) program.

-B Ignored.

-D *debugmode*

This option may be used for debugging purposes. This allows **telnetd** to print out debugging information to the connection, allowing the user to see what **telnetd** is doing. There are several possible values for *debugmode*:

options Prints information about the negotiation of TELNET options.

report Prints the **options** information, plus some additional information about what processing is going on.

netdata Displays the data stream received by **telnetd**.

ptydata Displays data written to the pty.

exercise Has not been implemented yet.

−e require encryption to be turned on (in both direction) by the client and disconnects if the client tries to turn the encryption off (in either direction).

−h Disables the printing of host-specific information before login has been completed.

−k

−l Ignored.

−n Disable TCP keep-alives. Normally **telnetd** enables the TCP keep-alive mechanism to probe connections that have been idle for some period of time to determine if the client is still there, so that idle connections from machines that have crashed or can no longer be reached may be cleaned up.

−r *lowpty-highpty*
This option is only enabled when **telnetd** is compiled for UNICOS. It specifies an inclusive range of pseudo-terminal devices to use. If the system has sysconf variable _SC_CRAY_NPTY configured, the default pty search range is 0 to _SC_CRAY_NPTY; otherwise, the default range is 0 to 128. Either *lowpty* or *highpty* may be omitted to allow changing either end of the search range. If *lowpty* is omitted, the - character is still required so that **telnetd** can differentiate *highpty* from *lowpty*.

−S *tos*

−u *len* This option is used to specify the size of the field in the utmp structure that holds the remote host name. If the resolved host name is longer than *len*, the dotted decimal value will be used instead. This allows hosts with very long host names that overflow this field to still be uniquely identified. Specifying **−u0** indicates that only dotted decimal addresses should be put into the utmp file.

−U This option causes **telnetd** to refuse connections from addresses that cannot be mapped back into a symbolic name via the gethostbyaddr(3) routine.

−X *authtype*
This option is only valid if **telnetd** has been built with support for the authentication option. It disables the use of *authtype* authentication, and can be used to temporarily disable a specific authentication type without having to recompile **telnetd**.

−L *pathname*
Specify pathname to an alternative login program.

−y Makes **telnetd** not warn when a user is trying to login with a cleartext password.

Telnetd operates by allocating a pseudo-terminal device (see pty(4)) for a client, then creating a login process which has the slave side of the pseudo-terminal as stdin, stdout and stderr. **Telnetd** manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session is started up, **telnetd** sends TELNET options to the client side indicating a willingness to do the following TELNET options, which are described in more detail below:

```
DO AUTHENTICATION
WILL ENCRYPT
DO TERMINAL TYPE
DO TSPEED
DO XDISPLOC
DO NEW-ENVIRON
DO ENVIRON
WILL SUPPRESS GO AHEAD
DO ECHO
DO LINEMODE
DO NAWS
WILL STATUS
DO LFLOW
DO TIMING-MARK
```

The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with `XTABS and CRMOD` enabled (see `tty`(4)).

`Telnetd` has support for enabling locally the following TELNET options:

WILL ECHO	When the `LINEMODE` option is enabled, a `WILL ECHO` or `WONT ECHO` will be sent to the client to indicate the current state of terminal echoing. When terminal echo is not desired, a `WILL ECHO` is sent to indicate that telnetd will take care of echoing any data that needs to be echoed to the terminal, and then nothing is echoed. When terminal echo is desired, a `WONT ECHO` is sent to indicate that telnetd will not be doing any terminal echoing, so the client should do any terminal echoing that is needed.
WILL BINARY	Indicates that the client is willing to send a 8 bits of data, rather than the normal 7 bits of the Network Virtual Terminal.
WILL SGA	Indicates that it will not be sending `IAC GA`, go ahead, commands.
WILL STATUS	Indicates a willingness to send the client, upon request, of the current status of all TELNET options.
WILL TIMING-MARK	Whenever a `DO TIMING-MARK` command is received, it is always responded to with a `WILL TIMING-MARK`
WILL LOGOUT	When a `DO LOGOUT` is received, a `WILL LOGOUT` is sent in response, and the TELNET session is shut down.
WILL ENCRYPT	Only sent if **`telnetd`** is compiled with support for data encryption, and indicates a willingness to decrypt the data stream.

`Telnetd` has support for enabling remotely the following TELNET options:

DO BINARY	Sent to indicate that telnetd is willing to receive an 8 bit data stream.
DO LFLOW	Requests that the client handle flow control characters remotely.
DO ECHO	This is not really supported, but is sent to identify a 4.2BSD `telnet`(1) client, which will improperly respond with `WILL ECHO`. If a `WILL ECHO` is received, a `DONT ECHO` will be sent in response.
DO TERMINAL-TYPE	Indicates a desire to be able to request the name of the type of terminal that is attached to the client side of the connection.

DO SGA	Indicates that it does not need to receive `IAC GA`, the go ahead command.
DO NAWS	Requests that the client inform the server when the window (display) size changes.
DO TERMINAL-SPEED	Indicates a desire to be able to request information about the speed of the serial line to which the client is attached.
DO XDISPLOC	Indicates a desire to be able to request the name of the X windows display that is associated with the telnet client.
DO NEW-ENVIRON	Indicates a desire to be able to request environment variable information, as described in RFC 1572.
DO ENVIRON	Indicates a desire to be able to request environment variable information, as described in RFC 1408.
DO LINEMODE	Only sent if **telnetd** is compiled with support for linemode, and requests that the client do line by line processing.
DO TIMING-MARK	Only sent if **telnetd** is compiled with support for both linemode and kludge linemode, and the client responded with `WONT LINEMODE`. If the client responds with `WILL TM`, the it is assumed that the client supports kludge linemode. Note that the [**-k**] option can be used to disable this.
DO AUTHENTICATION	Only sent if **telnetd** is compiled with support for authentication, and indicates a willingness to receive authentication information for automatic login.
DO ENCRYPT	Only sent if **telnetd** is compiled with support for data encryption, and indicates a willingness to decrypt the data stream.

FILES

/etc/services	
/etc/inittab	(UNICOS systems only)
/etc/iptos	(if supported)

SEE ALSO

telnet(1), login(1)

STANDARDS

RFC-854	TELNET PROTOCOL SPECIFICATION
RFC-855	TELNET OPTION SPECIFICATIONS
RFC-856	TELNET BINARY TRANSMISSION
RFC-857	TELNET ECHO OPTION
RFC-858	TELNET SUPPRESS GO AHEAD OPTION
RFC-859	TELNET STATUS OPTION
RFC-860	TELNET TIMING MARK OPTION
RFC-861	TELNET EXTENDED OPTIONS - LIST OPTION
RFC-885	TELNET END OF RECORD OPTION
RFC-1073	Telnet Window Size Option
RFC-1079	Telnet Terminal Speed Option
RFC-1091	Telnet Terminal-Type Option
RFC-1096	Telnet X Display Location Option

RFC-1123	Requirements for Internet Hosts -- Application and Support
RFC-1184	Telnet Linemode Option
RFC-1372	Telnet Remote Flow Control Option
RFC-1416	Telnet Authentication Option
RFC-1411	Telnet Authentication: Kerberos Version 4
RFC-1412	Telnet Authentication: SPX
RFC-1571	Telnet Environment Option Interoperability Issues
RFC-1572	Telnet Environment Option

BUGS

Some TELNET commands are only partially implemented.

Because of bugs in the original 4.2 BSD `telnet`(1), **telnetd** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD `telnet`(1).

Binary mode has no common interpretation except between similar operating systems (Unix in this case).

The terminal type name received from the remote client is converted to lower case.

Telnetd never sends TELNET `IAC GA` (go ahead) commands.

NAME

telnetd — DARPA TELNET protocol server

SYNOPSIS

/usr/libexec/telnetd [**-Uhlkns46**] [**-D** *debugmode*] [**-S** *tos*] [**-X** *authtype*] [**-a** *authmode*] [**-edebug**] [**-g** *gettyent*] [**-u** *len*] [**-debug** [*port*]]

DESCRIPTION

The **telnetd** command is a server which supports the DARPA standard TELNET virtual terminal protocol. **telnetd** is normally invoked by the internet server (see inetd(8)) for requests to connect to the TELNET port as indicated by the /etc/services file (see services(5)). The **-debug** option may be used to start up **telnetd** manually, instead of through inetd(8). If started up this way, *port* may be specified to run **telnetd** on an alternate TCP port number.

The **telnetd** command accepts the following options:

-a *authmode*

This option may be used for specifying what mode should be used for authentication. Note that this option is only useful if **telnetd** has been compiled with support for the AUTHENTICATION option. There are several valid values for *authmode*:

debug	Turns on authentication debugging code.
user	Only allow connections when the remote user can provide valid authentication information to identify the remote user, and is allowed access to the specified account without providing a password.
valid	Only allow connections when the remote user can provide valid authentication information to identify the remote user. The login(1) command will provide any additional user verification needed if the remote user is not allowed automatic access to the specified account.
other	Only allow connections that supply some authentication information. This option is currently not supported by any of the existing authentication mechanisms, and is thus the same as specifying **-a valid**.
none	This is the default state. Authentication information is not required. If no or insufficient authentication information is provided, then the login(1) program will provide the necessary user verification.
off	This disables the authentication code. All user verification will happen through the login(1) program.

-D *debugmode*

This option may be used for debugging purposes. This allows **telnetd** to print out debugging information to the connection, allowing the user to see what **telnetd** is doing. There are several possible values for *debugmode*:

options	Prints information about the negotiation of TELNET options.
report	Prints the **options** information, plus some additional information about what processing is going on.
netdata	Displays the data stream received by **telnetd**.

ptydata Displays data written to the pty.

exercise Has not been implemented yet.

−debug Enables debugging on each socket created by **telnetd** (see SO_DEBUG in `socket`(2)).

−edebug If **telnetd** has been compiled with support for data encryption, then the **−edebug** option may be used to enable encryption debugging code.

−g *gettyent*
Specifies which entry from `/etc/gettytab` should be used to get banner strings, login program and other information. The default entry is `default`.

−h Disables the printing of host-specific information before login has been completed.

−k This option is only useful if **telnetd** has been compiled with both linemode and kludge linemode support. If the **−k** option is specified, then if the remote client does not support the LINEMODE option, then **telnetd** will operate in character at a time mode. It will still support kludge linemode, but will only go into kludge linemode if the remote client requests it. (This is done by by the client sending DONT SUPPRESS-GO-AHEAD and DONT ECHO.) The **−k** option is most useful when there are remote clients that do not support kludge linemode, but pass the heuristic (if they respond with WILL TIMING-MARK in response to a DO TIMING-MARK) for kludge linemode support.

−l Specifies line mode. Tries to force clients to use line-at-a-time mode. If the LINEMODE option is not supported, it will go into kludge linemode.

−n Disable TCP keep-alives. Normally **telnetd** enables the TCP keep-alive mechanism to probe connections that have been idle for some period of time to determine if the client is still there, so that idle connections from machines that have crashed or can no longer be reached may be cleaned up.

−s This option is only enabled if **telnetd** is compiled with support for secure logins. It causes the **−s** option to be passed on to `login`(1), and thus is only useful if `login`(1) supports the **−s** flag to indicate that only Kerberos or S/Key validated logins are allowed, and is usually useful for controlling remote logins from outside of a firewall.

−S *tos* This option sets the IP Type-of Service (TOS) option on the connection to the value tos, which may be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file. This option has no effect on NetBSD.

−u *len* This option is used to specify the size of the field in the `utmp` structure that holds the remote host name. If the resolved host name is longer than *len*, the dotted decimal value will be used instead. This allows hosts with very long host names that overflow this field to still be uniquely identified. Specifying **−u0** indicates that only dotted decimal addresses should be put into the `utmp` file.

−U This option causes **telnetd** to refuse connections from addresses that cannot be mapped back into a symbolic name via the `getnameinfo`(3) routine.

−X *authtype*
This option is only valid if **telnetd** has been built with support for the authentication option. It disables the use of *authtype* authentication, and can be used to temporarily disable a specific authentication type without having to recompile **telnetd**.

−4

−6	Specifies address family to be used on **−debug** mode. During normal operation (called from `inetd`(8)) **telnetd** will use the file descriptor passed from `inetd`(8).

telnetd operates by allocating a pseudo-terminal device (see `pty`(4)) for a client, then creating a login process which has the slave side of the pseudo-terminal as `stdin`, `stdout` and `stderr`. **telnetd** manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session is started up, **telnetd** sends TELNET options to the client side indicating a willingness to do the following TELNET options, which are described in more detail below:

```
DO AUTHENTICATION
WILL ENCRYPT
DO TERMINAL TYPE
DO TSPEED
DO XDISPLOC
DO NEW-ENVIRON
DO ENVIRON
WILL SUPPRESS GO AHEAD
DO ECHO
DO LINEMODE
DO NAWS
WILL STATUS
DO LFLOW
DO TIMING-MARK
```

The pseudo-terminal allocated to the client is configured to operate in cooked mode, and with `XTABS and CRMOD` enabled (see `tty`(4)).

telnetd has support for enabling locally the following TELNET options:

WILL ECHO	When the `LINEMODE` option is enabled, a `WILL ECHO` or `WONT ECHO` will be sent to the client to indicate the current state of terminal echoing. When terminal echo is not desired, a `WILL ECHO` is sent to indicate that telnetd will take care of echoing any data that needs to be echoed to the terminal, and then nothing is echoed. When terminal echo is desired, a `WONT ECHO` is sent to indicate that telnetd will not be doing any terminal echoing, so the client should do any terminal echoing that is needed.
WILL BINARY	Indicates that the client is willing to send a 8 bits of data, rather than the normal 7 bits of the Network Virtual Terminal.
WILL SGA	Indicates that it will not be sending `IAC GA`, go ahead, commands.
WILL STATUS	Indicates a willingness to send the client, upon request, of the current status of all TELNET options.
WILL TIMING-MARK	Whenever a `DO TIMING-MARK` command is received, it is always responded to with a `WILL TIMING-MARK`
WILL LOGOUT	When a `DO LOGOUT` is received, a `WILL LOGOUT` is sent in response, and the TELNET session is shut down.
WILL ENCRYPT	Only sent if **telnetd** is compiled with support for data encryption, and indicates a willingness to decrypt the data stream.

telnetd has support for enabling remotely the following TELNET options:

DO BINARY	Sent to indicate that telnetd is willing to receive an 8 bit data stream.
DO LFLOW	Requests that the client handle flow control characters remotely.
DO ECHO	This is not really supported, but is sent to identify a 4.2BSD `telnet`(1) client, which will improperly respond with `WILL ECHO`. If a `WILL ECHO` is received, a `DONT ECHO` will be sent in response.
DO TERMINAL-TYPE	Indicates a desire to be able to request the name of the type of terminal that is attached to the client side of the connection.
DO SGA	Indicates that it does not need to receive `IAC GA`, the go ahead command.
DO NAWS	Requests that the client inform the server when the window (display) size changes.
DO TERMINAL-SPEED	Indicates a desire to be able to request information about the speed of the serial line to which the client is attached.
DO XDISPLOC	Indicates a desire to be able to request the name of the X windows display that is associated with the telnet client.
DO NEW-ENVIRON	Indicates a desire to be able to request environment variable information, as described in RFC 1572.
DO ENVIRON	Indicates a desire to be able to request environment variable information, as described in RFC 1408.
DO LINEMODE	Only sent if **telnetd** is compiled with support for linemode, and requests that the client do line by line processing.
DO TIMING-MARK	Only sent if **telnetd** is compiled with support for both linemode and kludge linemode, and the client responded with `WONT LINEMODE`. If the client responds with `WILL TM`, the it is assumed that the client supports kludge linemode. Note that the [**-k**] option can be used to disable this.
DO AUTHENTICATION	Only sent if **telnetd** is compiled with support for authentication, and indicates a willingness to receive authentication information for automatic login.
DO ENCRYPT	Only sent if **telnetd** is compiled with support for data encryption, and indicates a willingness to decrypt the data stream.

At the end of a login session, **telnetd** invokes the `ttyaction`(3) facility with an action of "telnetd" and user "root" to execute site-specific commands.

FILES

`/etc/services`
`/etc/iptos` (if supported)

SEE ALSO

`login`(1), `skey`(1), `telnet`(1), `ttyaction`(3)

STANDARDS

RFC 854	TELNET PROTOCOL SPECIFICATION
RFC 855	TELNET OPTION SPECIFICATIONS

RFC 856	TELNET BINARY TRANSMISSION
RFC 857	TELNET ECHO OPTION
RFC 858	TELNET SUPPRESS GO AHEAD OPTION
RFC 859	TELNET STATUS OPTION
RFC 860	TELNET TIMING MARK OPTION
RFC 861	TELNET EXTENDED OPTIONS - LIST OPTION
RFC 885	TELNET END OF RECORD OPTION
RFC 1073	Telnet Window Size Option
RFC 1079	Telnet Terminal Speed Option
RFC 1091	Telnet Terminal-Type Option
RFC 1096	Telnet X Display Location Option
RFC 1123	Requirements for Internet Hosts -- Application and Support
RFC 1184	Telnet Linemode Option
RFC 1372	Telnet Remote Flow Control Option
RFC 1416	Telnet Authentication Option
RFC 1411	Telnet Authentication: Kerberos Version 4
RFC 1412	Telnet Authentication: SPX
RFC 1571	Telnet Environment Option Interoperability Issues
RFC 1572	Telnet Environment Option

BUGS

Some TELNET commands are only partially implemented.

Because of bugs in the original 4.2BSD `telnet`(1), **`telnetd`** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2BSD `telnet`(1).

Binary mode has no common interpretation except between similar operating systems (UNIX in this case) .

The terminal type name received from the remote client is converted to lower case.

`telnetd` never sends TELNET `IAC GA` (go ahead) commands.

NAME

tftpd — DARPA Internet Trivial File Transfer Protocol server

SYNOPSIS

tftpd [**-d**] [**-g** *group*] [**-l**] [**-n**] [**-s** *directory*] [**-u** *user*] [*directory ...*]

DESCRIPTION

tftpd is a server which supports the DARPA Trivial File Transfer Protocol. The TFTP server operates at the port indicated in the `tftp` service description; see `services`(5). The server is normally started by `inetd`(8).

The use of `tftp`(1) does not require an account or password on the remote system. Due to the lack of authentication information, **tftpd** will allow only publicly readable files to be accessed. Filenames beginning in "**../**" or containing "/**..**/" are not allowed. Files may be written to only if they already exist and are publicly writable.

Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling tftp service. The server should have the user ID with the lowest possible privilege.

Access to files may be restricted by invoking **tftpd** with a list of directories by including up to 20 pathnames as server program arguments in `/etc/inetd.conf`. In this case access is restricted to files whose names are prefixed by the one of the given directories. The given directories are also treated as a search path for relative filename requests.

The options are:

-d Enable verbose debugging messages to `syslogd`(8).

-g *group* Change gid to that of *group* on startup. If this isn't specified, the gid is set to that of the *user* specified with **-u**.

-l Logs all requests using `syslog`(3).

-n Suppresses negative acknowledgement of requests for nonexistent relative filenames.

-s *directory*
tftpd will `chroot`(2) to *directory* on startup. This is recommended for security reasons (so that files other than those in the `/tftpboot` directory aren't accessible). If the remote host passes the directory name as part of the file name to transfer, you may have to create a symbolic link from 'tftpboot' to '.' under `/tftpboot`.

-u *user* Change uid to that of *user* on startup. If **-u** isn't given, *user* defaults to "nobody". If **-g** isn't also given, change the gid to that of *user* as well.

SEE ALSO

`tftp`(1), `inetd`(8)

The TFTP Protocol (Revision 2), RFC, 1350, July 1992.

TFTP Option Extension, RFC, 2347, May 1998.

TFTP Blocksize Option, RFC, 2348, May 1998.

TFTP Timeout Interval and Transfer Size Options, RFC, 2349, May 1998.

HISTORY

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

The **−s** flag appeared in NetBSD 1.0.

The **−g** and **−u** flags appeared in NetBSD 1.4.

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.

BUGS

Files larger than 33488896 octets (65535 blocks) cannot be transferred without client and server supporting blocksize negotiation (RFCs 2347 and 2348).

Many tftp clients will not transfer files over 16744448 octets (32767 blocks).

SECURITY CONSIDERATIONS

You are *strongly* advised to set up **tftpd** using the **−s** flag in conjunction with the name of the directory that contains the files that **tftpd** will serve to remote hosts (e.g., `/tftpboot`). This ensures that only the files that should be served to remote hosts can be accessed by them.

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

timed — time server daemon

SYNOPSIS

timed [**-dMt**] [**-F** *host ...*] [**-G** *netgroup*] [**-i** *network* | **-n** *network*]

DESCRIPTION

The **timed** utility is a time server daemon which is normally invoked at boot time from the `rc`(8) file. It synchronizes the host's time with the time of other machines, which are also running **timed**, in a local area network. These time servers will slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. The average network time is computed from measurements of clock differences using the ICMP timestamp request message.

The following options are available:

-d Enable debugging mode; do not detach from the terminal.

-F *host ...*
Create a list of trusted hosts. The **timed** utility will only accept trusted hosts as masters. If it finds an untrusted host claiming to be master, **timed** will suppress incoming messages from that host and call for a new election. This option implies the **-M** option. If this option is not specified, all hosts on the connected networks are treated as trustworthy.

-G *netgroup*
Specify a netgroup of trustworthy hosts, in addition to any masters specified with the **-M** flag. This option may only be specified once.

-i *network*
Add *network* to the list of networks to ignore. All other networks to which the machine is directly connected are used by **timed**. This option may be specified multiple times to add more than one network to the list.

-M Allow this host to become a **timed** master if necessary.

-n *network*
Add *network* to the list of allowed networks. All other networks to which the machine is directly connected are ignored by **timed**. This option may be specified multiple times to add more than one network to the list.

-t Enable tracing of received messages and log to the file `/var/log/timed.log`. Tracing can be turned on or off while **timed** is running with the `timedc`(8) utility.

The **-n** and **-i** flags are mutually exclusive and require as arguments real networks to which the host is connected (see `networks`(5)). If neither flag is specified, **timed** will listen on all connected networks.

A **timed** running without the **-M** nor **-F** flags will always remain a slave. If the **-F** flag is not used, **timed** will treat all machines as trustworthy.

The **timed** utility is based on a master-slave scheme. When **timed** is started on a machine, it asks the master for the network time and sets the host's clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls `adjtime`(2) to perform the needed corrections on the host's clock.

It also communicates with `date`(1) in order to set the date globally, and with `timedc`(8), a **timed** control utility. If the machine running the master becomes unreachable, the slaves will elect a new master from among those slaves which are running with at least one of the **-M** and **-F** flags.

At startup **timed** normally checks for a master time server on each network to which it is connected, except as modified by the **−n** and **−i** options described above. It will request synchronization service from the first master server located. If permitted by the **−M** or **−F** flags, it will provide synchronization service on any attached networks on which no trusted master server was detected. Such a server propagates the time computed by the top-level master. The **timed** utility will periodically check for the presence of a master on those networks for which it is operating as a slave. If it finds that there are no trusted masters on a network, it will begin the election process on that network.

One way to synchronize a group of machines is to use `ntpd`(8) to synchronize the clock of one machine to a distant standard or a radio receiver and **−F** *hostname* to tell its **timed** to trust only itself.

Messages printed by the kernel on the system console occur with interrupts disabled. This means that the clock stops while they are printing. A machine with many disk or network hardware problems and consequent messages cannot keep good time by itself. Each message typically causes the clock to lose a dozen milliseconds. A time daemon can correct the result.

Messages in the system log about machines that failed to respond usually indicate machines that crashed or were turned off. Complaints about machines that failed to respond to initial time settings are often associated with "multi-homed" machines that looked for time masters on more than one network and eventually chose to become a slave on the other network.

WARNINGS

Temporal chaos will result if two or more time daemons attempt to adjust the same clock. If both **timed** and another time daemon are run on the same machine, ensure that the **−F** flag is used, so that **timed** never attempts to adjust the local clock.

The protocol is based on UDP/IP broadcasts. All machines within the range of a broadcast that are using the TSP protocol must cooperate. There cannot be more than a single administrative domain using the **−F** flag among all machines reached by a broadcast packet. Failure to follow this rule is usually indicated by complaints concerning "untrusted" machines in the system log.

FILES

`/var/log/timed.log`	tracing file for **timed**
`/var/log/timed.masterlog`	log file for master **timed**

SEE ALSO

`date`(1), `adjtime`(2), `gettimeofday`(2), `icmp`(4), `netgroup`(5), `networks`(5), `ntpd`(8), `timedc`(8)

R. Gusella and S. Zatti, *TSP: The Time Synchronization Protocol for UNIX 4.3BSD*.

HISTORY

The **timed** utility appeared in 4.3BSD.

NAME

timedc — timed control program

SYNOPSIS

timedc [*command* [*argument ...*]]

DESCRIPTION

timedc is used to control the operation of the timed(8) program. It may be used to:

- Measure the differences between machines' clocks,
- Find the location where the master time server is running,
- Enable or disable tracing of messages received by timed(8), and
- Perform various debugging actions.

Without any arguments, **timedc** will prompt for commands from the standard input. If arguments are supplied, **timedc** interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing **timedc** to read commands from a file. Commands may be abbreviated; recognized commands are:

? [*command ...*]

help [*command ...*]
Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

clockdiff *host ...*
Compute the differences between the clock of the host machine and the clocks of the machines given as arguments.

msite [*host ...*]
Show the master time server for specified host(s).

trace { *on* | *off* }
Enable or disable the tracing of incoming messages to timed(8) in the file `/var/log/timed.log`.

election *host*
Asks the daemon on the target host to reset its "election" timers and to ensure that a time master has been elected.

quit Exit from timedc.

Other commands may be included for use in testing and debugging timed(8); the help command and the program source may be consulted for details.

FILES

`/var/log/timed.log`	tracing file for timed
`/var/log/timed.masterlog`	log file for master timed

DIAGNOSTICS

?Ambiguous command
abbreviation matches more than one command

?Invalid command
no match found

SEE ALSO

`date`(1), `adjtime`(2), `icmp`(4), `timed`(8)

R. Gusella and S. Zatti, *TSP: The Time Synchronization Protocol for UNIX 4.3BSD*.

HISTORY

The **`timedc`** command appeared in 4.3BSD.

NAME

tlsmgr − Postfix TLS session cache and PRNG manager

SYNOPSIS

tlsmgr [generic Postfix daemon options]

DESCRIPTION

The **tlsmgr**(8) manages the Postfix TLS session caches. It stores and retrieves cache entries on request by **smtpd**(8) and **smtp**(8) processes, and periodically removes entries that have expired.

The **tlsmgr**(8) also manages the PRNG (pseudo random number generator) pool. It answers queries by the **smtpd**(8) and **smtp**(8) processes to seed their internal PRNG pools.

The **tlsmgr**(8)'s PRNG pool is initially seeded from an external source (EGD, /dev/urandom, or regular file). It is updated at configurable pseudo-random intervals with data from the external source. It is updated periodically with data from TLS session cache entries and with the time of day, and is updated with the time of day whenever a process requests **tlsmgr**(8) service.

The **tlsmgr**(8) saves the PRNG state to an exchange file periodically and when the process terminates, and reads the exchange file when initializing its PRNG.

SECURITY

The **tlsmgr**(8) is not security-sensitive. The code that maintains the external and internal PRNG pools does not "trust" the data that it manipulates, and the code that maintains the TLS session cache does not touch the contents of the cached entries, except for seeding its internal PRNG pool.

The **tlsmgr**(8) can be run chrooted and with reduced privileges. At process startup it connects to the entropy source and exchange file, and creates or truncates the optional TLS session cache files.

DIAGNOSTICS

Problems and transactions are logged to the syslog daemon.

BUGS

There is no automatic means to limit the number of entries in the TLS session caches and/or the size of the TLS cache files.

CONFIGURATION PARAMETERS

Changes to **main.cf** are not picked up automatically, because **tlsmgr**(8) is a persistent processes. Use the command "**postfix reload**" after a configuration change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

TLS SESSION CACHE

lmtp_tls_loglevel (0)
The LMTP-specific version of the smtp_tls_loglevel configuration parameter.

lmtp_tls_session_cache_database (empty)
The LMTP-specific version of the smtp_tls_session_cache_database configuration parameter.

lmtp_tls_session_cache_timeout (3600s)
The LMTP-specific version of the smtp_tls_session_cache_timeout configuration parameter.

smtp_tls_loglevel (0)
Enable additional Postfix SMTP client logging of TLS activity.

smtp_tls_session_cache_database (empty)
Name of the file containing the optional Postfix SMTP client TLS session cache.

smtp_tls_session_cache_timeout (3600s)
The expiration time of Postfix SMTP client TLS session cache information.

smtpd_tls_loglevel (0)
Enable additional Postfix SMTP server logging of TLS activity.

smtpd_tls_session_cache_database (empty)
Name of the file containing the optional Postfix SMTP server TLS session cache.

smtpd_tls_session_cache_timeout (3600s)
The expiration time of Postfix SMTP server TLS session cache information.

PSEUDO RANDOM NUMBER GENERATOR

tls_random_source (see ’postconf -d’ output)
The external entropy source for the in-memory **tlsmgr**(8) pseudo random number generator (PRNG) pool.

tls_random_bytes (32)
The number of bytes that **tlsmgr**(8) reads from $tls_random_source when (re)seeding the in-memory pseudo random number generator (PRNG) pool.

tls_random_exchange_name (${config_directory}/prng_exch)
Name of the pseudo random number generator (PRNG) state file that is maintained by **tlsmgr**(8).

tls_random_prng_update_period (3600s)
The time between attempts by **tlsmgr**(8) to save the state of the pseudo random number generator (PRNG) to the file specified with $tls_random_exchange_name.

tls_random_reseed_period (3600s)
The maximal time between attempts by **tlsmgr**(8) to re-seed the in-memory pseudo random number generator (PRNG) pool from external sources.

MISCELLANEOUS CONTROLS

config_directory (see ’postconf -d’ output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

smtp(8), Postfix SMTP client
smtpd(8), Postfix SMTP server
postconf(5), configuration parameters
master(5), generic daemon options
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
TLS_README, Postfix TLS configuration and operation

LICENSE

AUTHOR(S)

Lutz Jaenicke
BTU Cottbus
Allgemeine Elektrotechnik
Universitaetsplatz 3-4
D-03044 Cottbus, Germany

Adapted by:
Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

tpctl — touch panel calibration utility

SYNOPSIS

tpctl [**-D** *dispdevname*] [**-d** *devname*] [**-f** *filename*] [**-hnuv**]

DESCRIPTION

tpctl is a touch panel calibration utility. **tpctl** calibrates a touch panel and saves and restores the calibration parameters into/from a parameter database file.

Available command-line flags are:

-D *dispdevname*	Specify display device name.
-d *devname*	Specify touch panel device name.
-f *filename*	Specify alternate parameter database file name.
-h	Print brief description.
-n	Do not change the parameter database file.
-u	Force calibration. Without this flag, **tpctl** won't do calibration if the database file already contains parameters for the touch panel.
-v	Verbose mode.

You calibrate the touch panel the first time you run **tpctl**. If you see a cross cursor on the screen, you should tap the center of the cursor to calibrate the touch panel, or you can abort the calibration with the 'ESC' key. Five cursors will appear on the screen in turn. Once calibration is done, **tpctl** saves the calibration parameters into the database file and uses the saved parameters to calibrate the touch panel.

You can run **tpctl** automatically with `/etc/rc.d/tpctl`.

FILES

`/etc/tpctl.dat`	The default calibration parameter database file. The **-f** flag may be used to specify an alternate database file name. **tpctl** will create an empty database file if it doesn't exist.
`/dev/ttyE0`	The default display device, which is used to display the cursor during calibration. The **-D** flag may be used to specify an alternate display device name. The display device must provide the 'hpcfb' interface as defined in `/usr/include/dev/hpc/hpcfbio.h`.
`/dev/wsmux0`	The default touch panel device. The **-d** flag may be used to specify an alternate touch panel device name.

SEE ALSO

`rc.conf`(5)

BUGS

tpctl isn't available on all ports because it requires a display device which provides the 'hpcfb' interface.

NAME

traceroute − print the route packets take to network host

SYNOPSIS

traceroute [**−aDFPIdlMnrvx**] [**−f** *first_ttl*]
[**−g** *gateway*] [**−i** *iface*] [**−m** max_ttl]
[**−p** *port*] [**−q** *nqueries*] [**−s** *src_addr*]
[**−t** *tos*] [**−w** *waittime*] [**−A** *as_server*]
host [*packetlen*]

DESCRIPTION

The Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking the route one's packets follow (or finding the miscreant gateway that's discarding your packets) can be difficult. *Traceroute* uses the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

The only mandatory parameter is the destination host name or IP number. The default probe datagram length is 40 bytes, but this may be increased by specifying a packet length (in bytes) after the destination host name.

Other options are:

−a Turn on AS# lookups for each hop encountered.

−A Turn on AS# lookups and use the given server instead of the default.

−d Turn on socket-level debugging.

−D Dump the packet data to standard error before transmitting it.

−f Set the initial time-to-live used in the first outgoing probe packet.

−F Set the "don't fragment" bit.

−g Specify a loose source route gateway (8 maximum).

−i Specify a network interface to obtain the source IP address for outgoing probe packets. This is normally only useful on a multi-homed host. (See the **−s** flag for another way to do this.)

−I Use ICMP ECHO instead of UDP datagrams.

−l Display the ttl value of the returned packet. This is useful for checking for asymmetric routing.

−m Set the max time-to-live (max number of hops) used in outgoing probe packets. The default value is taken from the *net.inet.ip.ttl* sysctl(3) variable.

−M If found, show the MPLS Label and the Experimental (EXP) bit for the hop.

−n Print hop addresses numerically rather than symbolically and numerically (saves a nameserver address-to-name lookup for each gateway found on the path).

−p Set the base UDP port number used in probes (default is 33434). Traceroute hopes that nothing is listening on UDP ports *base* to *base + nhops − 1* at the destination host (so an ICMP PORT_UNREACHABLE message will be returned to terminate the route tracing). If something is listening on a port in the default range, this option can be used to pick an unused port range.

−P Set the "don't fragment" bit, and use the next hop mtu each time we get the "need fragmentation" error, thus probing the path MTU.

−q Set the number of probe packets sent for each hop. By default, traceroute sends three probe packets.

−r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by *routed*(8)).

−s Use the following IP address (which usually is given as an IP number, not a hostname) as the source address in outgoing probe packets. On multi-homed hosts (those with more than one IP address), this option can be used to force the source address to be something other than the IP address of the interface the probe packet is sent on. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent. (See the **−i** flag for another way to do this.)

−t Set the *type-of-service* in probe packets to the following value (default zero). The value must be a decimal integer in the range 0 to 255. This option can be used to see if different types-of-service result in different paths. (If you are not running 4.4BSD, this may be academic since the normal network services like telnet and ftp don't let you control the TOS). Not all values of TOS are legal or meaningful − see the IP spec for definitions. Useful values are probably '**-t** *16*' (low delay) and '**-t** *8*' (high throughput).

−v Verbose output. Received ICMP packets other than TIME_EXCEEDED and UNREACHABLEs are listed.

−w Set the time (in seconds) to wait for a response to a probe (default 5 sec.).

−x Toggle checksums. Normally, this prevents traceroute from calculating checksums. In some cases, the operating system can overwrite parts of the outgoing packet but not recalculate the checksum (so in some cases the default is to not calculate checksums and using **−x** causes them to be calculated). Note that checksums are usually required for the last hop when using ICMP ECHO probes (**−I**).

This program attempts to trace the route an IP packet would follow to some internet host by launching UDP probe packets with a small ttl (time to live) then listening for an ICMP "time exceeded" reply from a gateway. We start our probes with a ttl of one and increase by one until we get an ICMP "port unreachable" (which means we got to "host") or hit a max (which defaults to 30 hops & can be changed with the **−m** flag). Three probes (change with **−q** flag) are sent at each ttl setting and a line is printed showing the ttl, address of the gateway and round trip time of each probe. If the probe answers come from different gateways, the address of each responding system will be printed. If there is no response within a 5 sec. timeout interval (changed with the **−w** flag), a "*" is printed for that probe.

We don't want the destination host to process the UDP probe packets so the destination port is set to an unlikely value (if some clod on the destination is using that value, it can be changed with the **−p** flag).

A sample use and output might be:

```
[yak 71]% traceroute nis.nsf.net.
traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 38 byte packet
 1  helios.ee.lbl.gov (128.3.112.1)  19 ms  19 ms  0 ms
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms  19 ms
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  39 ms
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  39 ms  39 ms  39 ms
 6  128.32.197.4 (128.32.197.4)  40 ms  59 ms  59 ms
 7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  59 ms
 8  129.140.70.13 (129.140.70.13)  99 ms  99 ms  80 ms
 9  129.140.71.6 (129.140.71.6)  139 ms  239 ms  319 ms
10  129.140.81.7 (129.140.81.7)  220 ms  199 ms  199 ms
11  nic.merit.edu (35.1.1.48)  239 ms  239 ms  239 ms
```

Note that lines 2 & 3 are the same. This is due to a buggy kernel on the 2nd hop system − lilac-dmc.Berkeley.EDU − that forwards packets with a zero ttl (a bug in the distributed version of 4.3BSD). Note that you have to guess what path the packets are taking cross-country since the NSFNET (129.140) doesn't supply address-to-name translations for its NSSes.

A more interesting example is:

```
[yak 72]% traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
 1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
 6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
 7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
 8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
 9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
12  * * *
13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms
```

Note that the gateways 12, 14, 15, 16 & 17 hops away either don't send ICMP "time exceeded" messages or send them with a ttl too small to reach us. 14 − 17 are running the MIT C Gateway code that doesn't send "time exceeded"s. God only knows what's going on with 12.

The silent gateway 12 in the above may be the result of a bug in the 4.[23]BSD network code (and its derivatives): 4.x (x ≤ 3) sends an unreachable message using whatever ttl remains in the original datagram. Since, for gateways, the remaining ttl is zero, the ICMP "time exceeded" is guaranteed to not make it back to us. The behavior of this bug is slightly more interesting when it appears on the destination system:

```
 1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  39 ms
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  39 ms  19 ms
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  19 ms
 5  ccn-nerif35.Berkeley.EDU (128.32.168.35)  39 ms  39 ms  39 ms
 6  csgw.Berkeley.EDU (128.32.133.254)  39 ms  59 ms  39 ms
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms !
```

Notice that there are 12 "gateways" (13 is the final destination) and exactly the last half of them are "missing". What's really happening is that rip (a Sun-3 running Sun OS3.5) is using the ttl from our arriving datagram as the ttl in its ICMP reply. So, the reply will time out on the return path (with no notice sent to anyone since ICMP's aren't sent for ICMP's) until we probe with a ttl that's at least twice the path length. I.e., rip is really only 7 hops away. A reply that returns with a ttl of 1 is a clue this problem exists. Traceroute prints a "!" after the time if the ttl is ≤ 1. Since vendors ship a lot of obsolete (DEC's ULTRIX, Sun 3.x) or non-standard (HP-UX) software, expect to see this problem frequently and/or take care picking the target host of your probes.

Other possible annotations after the time are **!H**, **!N**, or **!P** (got a host, network or protocol unreachable, respectively), **!S** or **!F** (source route failed or fragmentation needed − neither of these should ever occur and the associated gateway is busted if you see one), **!X** (communication administratively prohibited), or **!<N>** (ICMP unreachable code N). If almost all the probes result in some kind of unreachable, traceroute will give up and exit.

```
traceroute −g 10.3.0.5 128.182.0.0
```

will show the path from the Cambridge Mailbridge to PSC, while

```
traceroute −g 192.5.146.4 −g 10.3.0.5 35.0.0.0
```

will show the path from the Cambridge Mailbridge to Merit, using PSC to reach the Mailbridge.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use *traceroute* during normal operations or from automated scripts.

SEE ALSO

netstat(1), ping(8)

AUTHOR

Implemented by Van Jacobson from a suggestion by Steve Deering. Debugged by a cast of thousands with particularly cogent suggestions or fixes from C. Philip Wood, Tim Seaver and Ken Adelman.

The current version is available via anonymous ftp:

ftp://ftp.ee.lbl.gov/traceroute.tar.Z

BUGS

Please send bug reports to traceroute@ee.lbl.gov.

The AS number capability reports information that may sometimes be inaccurate due to discrepancies between the contents of the routing database server and the current state of the Internet.

NAME

traceroute6 — print the route IPv6 packets will take to the destination

SYNOPSIS

traceroute6 [**-dIlnrv**] [**-f** *firsthop*] [**-g** *gateway*] [**-m** *hoplimit*] [**-p** *port*]
[**-q** *probes*] [**-s** *src*] [**-w** *waittime*] *target* [*datalen*]

DESCRIPTION

-d Debug mode.

-f *firsthop*
Specify how many hops to skip in trace.

-g *gateway*
Specify intermediate gateway (**traceroute6** uses routing header).

-I Use ICMP6 ECHO instead of UDP datagrams.

-l Print both host hostnames and numeric addresses. Normally **traceroute6** prints only hostnames if **-n** is not specified, and only numeric addresses if **-n** is specified.

-m *hoplimit*
Specify maximum hoplimit.

-n Do not resolve numeric address to hostname.

-p *port*
Set UDP port number to *port*.

-q *probes*
Set the number of probe per hop count to *probes*.

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to send probes to a local host through an interface that has no route through it (e.g., after the interface was dropped by route6d(8)).

-s *src*
Src specifies the source IPv6 address to be used.

-v Be verbose.

-w *waittime*
Specify the delay time between probes.

EXIT STATUS

The **traceroute6** command exits 0 on success, and >0 on errors.

SEE ALSO

ping(8), ping6(8), traceroute(8)

HISTORY

The **traceroute6** command first appeared in WIDE hydrangea IPv6 protocol stack kit.

NAME

trivial-rewrite − Postfix address rewriting and resolving daemon

SYNOPSIS

trivial-rewrite [generic Postfix daemon options]

DESCRIPTION

The **trivial-rewrite**(8) daemon processes three types of client service requests:

rewrite *context address*

Rewrite an address to standard form, according to the address rewriting context:

local Append the domain names specified with **$myorigin** or **$mydomain** to incomplete addresses; do **swap_bangpath** and **allow_percent_hack** processing as described below, and strip source routed addresses (*@site,@site:user@domain*) to *user@domain* form.

remote Append the domain name specified with **$remote_header_rewrite_domain** to incomplete addresses. Otherwise the result is identical to that of the **local** address rewriting context. This prevents Postfix from appending the local domain to spam from poorly written remote clients.

resolve *sender address*

Resolve the address to a (*transport*, *nexthop*, *recipient*, *flags*) quadruple. The meaning of the results is as follows:

transport

The delivery agent to use. This is the first field of an entry in the **master.cf** file.

nexthop

The host to send to and optional delivery method information.

recipient

The envelope recipient address that is passed on to *nexthop*.

flags The address class, whether the address requires relaying, whether the address has problems, and whether the request failed.

verify *sender address*

Resolve the address for address verification purposes.

SERVER PROCESS MANAGEMENT

The **trivial-rewrite**(8) servers run under control by the Postfix master server. Each server can handle multiple simultaneous connections. When all servers are busy while a client connects, the master creates a new server process, provided that the trivial-rewrite server process limit is not exceeded. Each trivial-rewrite server terminates after serving at least **$max_use** clients of after **$max_idle** seconds of idle time.

STANDARDS

None. The command does not interact with the outside world.

SECURITY

The **trivial-rewrite**(8) daemon is not security sensitive. By default, this daemon does not talk to remote or local users. It can run at a fixed low privilege in a chrooted environment.

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

CONFIGURATION PARAMETERS

On busy mail systems a long time may pass before a **main.cf** change affecting **trivial-rewrite**(8) is picked up. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

COMPATIBILITY CONTROLS

resolve_dequoted_address (yes)
Resolve a recipient address safely instead of correctly, by looking inside quotes.

resolve_null_domain (no)
Resolve an address that ends in the "@" null domain as if the local hostname were specified, instead of rejecting the address as invalid.

resolve_numeric_domain (no)
Resolve "user@ipaddress" as "user@[ipaddress]", instead of rejecting the address as invalid.

ADDRESS REWRITING CONTROLS

myorigin ($myhostname)
The domain name that locally-posted mail appears to come from, and that locally posted mail is delivered to.

allow_percent_hack (yes)
Enable the rewriting of the form "user%domain" to "user@domain".

append_at_myorigin (yes)
With locally submitted mail, append the string "@$myorigin" to mail addresses without domain information.

append_dot_mydomain (yes)
With locally submitted mail, append the string ".$mydomain" to addresses that have no ".domain" information.

recipient_delimiter (empty)
The separator between user names and address extensions (user+foo).

swap_bangpath (yes)
Enable the rewriting of "site!user" into "user@site".

Available in Postfix 2.2 and later:

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.

ROUTING CONTROLS

The following is applicable to Postfix version 2.0 and later. Earlier versions do not have support for: virtual_transport, relay_transport, virtual_alias_domains, virtual_mailbox_domains or proxy_interfaces.

local_transport (local:$myhostname)
The default mail delivery transport and next-hop destination for final delivery to domains listed with mydestination, and for [ipaddress] destinations that match $inet_interfaces or $proxy_interfaces.

virtual_transport (virtual)
The default mail delivery transport and next-hop destination for final delivery to domains listed with $virtual_mailbox_domains.

relay_transport (relay)
The default mail delivery transport and next-hop destination for remote delivery to domains listed with $relay_domains.

default_transport (smtp)
The default mail delivery transport and next-hop destination for destinations that do not match $mydestination, $inet_interfaces, $proxy_interfaces, $virtual_alias_domains, $virtual_mailbox_domains, or $relay_domains.

parent_domain_matches_subdomains (see ’postconf -d’ output)
What Postfix features match subdomains of "domain.tld" automatically, instead of requiring an explicit ".domain.tld" pattern.

relayhost (empty)
The next-hop destination of non-local mail; overrides non-local domains in recipient addresses.

transport_maps (empty)
Optional lookup tables with mappings from recipient address to (message delivery transport, next-hop destination).

Available in Postfix version 2.3 and later:

sender_dependent_relayhost_maps (empty)
A sender-dependent override for the global relayhost parameter setting.

ADDRESS VERIFICATION CONTROLS

Postfix version 2.1 introduces sender and recipient address verification. This feature is implemented by sending probe email messages that are not actually delivered. By default, address verification probes use the same route as regular mail. To override specific aspects of message routing for address verification probes, specify one or more of the following:

address_verify_local_transport ($local_transport)
Overrides the local_transport parameter setting for address verification probes.

address_verify_virtual_transport ($virtual_transport)
Overrides the virtual_transport parameter setting for address verification probes.

address_verify_relay_transport ($relay_transport)
Overrides the relay_transport parameter setting for address verification probes.

address_verify_default_transport ($default_transport)
Overrides the default_transport parameter setting for address verification probes.

address_verify_relayhost ($relayhost)
Overrides the relayhost parameter setting for address verification probes.

address_verify_transport_maps ($transport_maps)
Overrides the transport_maps parameter setting for address verification probes.

Available in Postfix version 2.3 and later:

address_verify_sender_dependent_relayhost_maps (empty)
Overrides the sender_dependent_relayhost_maps parameter setting for address verification probes.

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

empty_address_recipient (MAILER-DAEMON)
The recipient of mail addressed to the null address.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

relocated_maps (empty)
Optional lookup tables with new contact information for users or domains that no longer exist.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon process.

queue_directory (see 'postconf -d' output)
The location of the Postfix top-level queue directory.

show_user_unknown_table_name (yes)
Display the name of the recipient table in the "User unknown" responses.

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".

Available in Postfix version 2.0 and later:

helpful_warnings (yes)
Log warnings about problematic configuration settings, and provide helpful suggestions.

SEE ALSO

postconf(5), configuration parameters
transport(5), transport table format
relocated(5), format of the "user has moved" table
master(8), process manager
syslogd(8), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
ADDRESS_CLASS_README, Postfix address classes howto
ADDRESS_VERIFICATION_README, Postfix address verification

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

trpt — transliterate protocol trace

SYNOPSIS

trpt [**−a**] [**−f**] [**−j**] [**−p** *hex-address*] [**−s**] [**−t**] [**−N** *system*] [**−M** *core*]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see setsockopt(2)), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

−a In addition to the normal output, print the values of the source and destination addresses for each packet recorded.

−f Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

−j Just give a list of the protocol control block addresses for which there are trace records.

−p Show only trace records associated with the protocol control block at the given address *hex-address*.

−s In addition to the normal output, print a detailed description of the packet sequencing information.

−t in addition to the normal output, print the values for all timers at each point in the trace.

−M *core*
Extract values associated with the name list from core.

−N *system*
Extract the name list from system.

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **−A** option to netstat(1). Then run **trpt** with the **−p** option, supplying the associated protocol control block addresses. The **−f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **−j** option may be useful in checking to see if any trace records are present for the socket in question.

SYSCTLS

The following sysctls are used by **trpt**. The TCP_DEBUG kernel option must be enabled.

net.inet.tcp.debug	Structure containing TCP sockets information used by **trpt**.
net.inet.tcp.debx	Number of TCP debug messages.

DIAGNOSTICS

no namelist
When the image doesn't contain the proper symbols to find the trace buffer; others which should be self explanatory.

SEE ALSO

netstat(1), setsockopt(2),

HISTORY

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

BUGS

Should also print the data for each input or output, but this is not saved in the trace record.

The output format is inscrutable and should be described here.

NAME

ttyflags — set device-specific flags for terminals

SYNOPSIS

ttyflags [**-v**] [**-a** | *tty ...*]

DESCRIPTION

ttyflags sets the device-specific flags for terminals using `TIOCSFLAGS`, based on the flags found on the terminal's line in `/etc/ttys`.

The options are as follows:

-a Set the flags for all terminals in `/etc/ttys`.

-v Be verbose about what the terminals' flags will be set to.

The *tty* arguments are optional, but must not be specified if the **-a** flag is used. If specified, the *tty* arguments should be the base names of the ttys, as found in `/etc/ttys`.

FILES

`/etc/ttys`

SEE ALSO

`getttyent`(3), `tty`(4), `ttys`(5)

HISTORY

The **ttyflags** utility appeared in NetBSD 1.0.

BUGS

The conditions on which to report an error are ill-defined. **ttyflags** tries to report all significant errors, perhaps going over-board at times.

NAME

tunefs — tune up an existing file system

SYNOPSIS

tunefs [**−AFN**] [**−e** *maxbpg*] [**−g** *avgfilesize*] [**−h** *avgfpdir*] [**−m** *minfree*] [**−o** *optimize_preference*] *special* | *filesys*

DESCRIPTION

tunefs is designed to change the dynamic parameters of a file system which affect the layout policies.

The following options are supported by **tunefs**:

−A Cause the values to be updated in all the alternate superblocks instead of just the standard superblock. If this option is not used, then use of a backup superblock by fsck(8) will lose anything changed by **tunefs**. **−A** is ignored when **−N** is specified.

−F Indicates that *special* is a file system image, rather than a device name or file system mount point. *special* will be accessed 'as-is'.

−N Display all the settable options (after any changes from the tuning options) but do not cause any of them to be changed.

−e *maxbpg*
This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

−g *avgfilesize*
This specifies the expected average file size.

−h *avgfpdir*
This specifies the expected number of files per directory.

−m *minfree*
This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value is set during creation of the filesystem, see newfs(8). This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 5% threshold. Note that if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

−o *optimize_preference*
The file system can either try to minimize the time spent allocating blocks, or it can attempt to minimize the space fragmentation on the disk. If the value of minfree (see above) is less than 5%, then the file system should optimize for space to avoid running out of full sized blocks. For values of minfree greater than or equal to 5%, fragmentation is unlikely to be problematical, and the file system can be optimized for time.

optimize_preference can be specified as either space or time.

SEE ALSO

fs(5), dumpfs(8), fsck_ffs(8), newfs(8)

M. McKusick, W. Joy, S. Leffler, and R. Fabry, "A Fast File System for UNIX", *ACM Transactions on Computer Systems 2*, 3, pp 181-197, August 1984, (reprinted in the BSD System Manager’s Manual, SMM:5).

HISTORY

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

BUGS

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the changes will only take effect if the program is run on unmounted file systems. To change the root file system, the system must be rebooted after the file system is tuned.

You can tune a file system, but you can’t tune a fish.

NAME

tzselect − select a time zone

SYNOPSIS

tzselect

DESCRIPTION

The **tzselect** program asks the user for information about the current location, and outputs the resulting time zone description to standard output. The output is suitable as a value for the TZ environment variable.

All interaction with the user is done via standard input and standard error.

ENVIRONMENT VARIABLES

AWK Name of a Posix-compliant *awk* program (default: **awk**).

TZDIR
Name of the directory containing time zone data files (default: **/usr/local/etc/zoneinfo**).

FILES

TZDIR/**iso3166.tab**
Table of ISO 3166 2-letter country codes and country names.

TZDIR/**zone.tab**
Table of country codes, latitude and longitude, TZ values, and descriptive comments.

TZDIR/*TZ*
Time zone data file for time zone *TZ*.

EXIT STATUS

The exit status is zero if a time zone was successfully obtained from the user, nonzero otherwise.

SEE ALSO

newctime(3), tzfile(5), zdump(8), zic(8)

NAME

umount — unmount filesystems

SYNOPSIS

umount [**−fvFR**] [**−t** *fstypelist*] *special* | *node*
umount −a [**−fvF**] [**−h** *host*] [**−t** *fstypelist*]

DESCRIPTION

The **umount** command calls the `unmount`(2) system call to remove a *special device* or the remote node (rhost:path) from the filesystem tree at the point *node*. If either *special* or *node* are not provided, the appropriate information is taken from the `fstab`(5) file.

The options are as follows:

−a All the currently mounted filesystems except the root are unmounted.

−f The filesystem is forcibly unmounted. Active special devices continue to work, but all other files return errors if further accesses are attempted. The root filesystem cannot be forcibly unmounted.

−F Fake the unmount; perform all other processing but do not actually attempt the unmount. (This is most useful in conjunction with **−v**, to see what **umount** would attempt to do).

−R Take the *special* | *node* argument as a path to be passed directly to `unmount`(2), bypassing all attempts to be smart about mechanically determining the correct path from the argument. This option is incompatible with any option that potentially unmounts more than one filesystem, such as **−a**, but it can be used with **−f** and/or **−v**. This is the only way to unmount something that does not appear as a directory (such as a nullfs mount of a plain file); there are probably other cases where it is necessary.

−h *host*
Only filesystems mounted from the specified host will be unmounted. This option is implies the **−a** option and, unless otherwise specified with the **−t** option, will only unmount NFS filesystems.

−t *fstypelist*
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. For example, the **umount** command:

```
umount -a -t nfs,mfs
```

unmounts all filesystems of the type NFS and MFS.

−v Verbose, additional information is printed out as each filesystem is unmounted.

FILES

`/etc/fstab` filesystem table

SEE ALSO

`unmount`(2), `fstab`(5), `mount`(8)

HISTORY

A **umount** command appeared in Version 6 AT&T UNIX.

NAME

unlink — call the unlink function

SYNOPSIS

unlink *file*

DESCRIPTION

The **unlink** utility performs the function call **unlink**(*file*).

file must be the pathname of an existing file.

EXIT STATUS

The **unlink** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

rm(1), rmdir(1), unlink(2), link(8)

STANDARDS

The **unlink** utility conforms to X/Open Commands and Utilities Issue 5 ("XCU5").

NAME

usbdevs — show USB devices connected to the system

SYNOPSIS

usbdevs [**-a** *addr*] [**-d**] [**-f** *dev*] [**-v**]

DESCRIPTION

usbdevs prints a listing of all USB devices connected to the system with some information about each device. The indentation of each line indicates its distance from the root.

The options are as follows:

-a *addr*	only print information about the device at the given address.
-d	Show the device drivers associated with each device.
-f *dev*	only print information for the given USB controller.
-v	Be verbose.

FILES

`/dev/usb[0-9]`	Default USB controllers.

SEE ALSO

usb(4)

HISTORY

The **usbdevs** command appeared in NetBSD 1.4.

NAME

user — manage user login information on the system

SYNOPSIS

user add -D [options]
user add [options] *user*
user del -D [options]
user del [options] *user*
user info [options] *user*
user mod [options] *user*

DESCRIPTION

The **user** utility acts as a frontend to the useradd(8), usermod(8), userinfo(8), and userdel(8) commands. The utilities by default are built with EXTENSIONS. This allows for further functionality.

For a full explanation of the options available, please see the relevant manual page.

EXIT STATUS

The **user** utility exits 0 on success, and >0 if an error occurs.

FILES

/etc/skel/.[A-z]*	Skeleton files for new user
/etc/usermgmt.conf	Configuration file for **user**, group(8) and the backend commands mentioned above.

SEE ALSO

chpass(1), group(5), passwd(5), usermgmt.conf(5), useradd(8), userdel(8), userinfo(8), usermod(8)

HISTORY

The **user** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **user** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

useradd — add a user to the system

SYNOPSIS

useradd **-D** [**-F**] [**-b** *base-dir*] [**-e** *expiry-time*] [**-f** *inactive-time*]
[**-g** *gid* | *name* | =uid] [**-k** *skel-dir*] [**-L** *login-class*] [**-M** *home-perm*]
[**-r** *lowuid*..*highuid*] [**-s** *shell*]
useradd [**-moSv**] [**-b** *base-dir*] [**-c** *comment*] [**-d** *home-dir*] [**-e** *expiry-time*]
[**-f** *inactive-time*] [**-G** *secondary-group*] [**-g** *gid* | *name* | =uid]
[**-k** *skel-dir*] [**-L** *login-class*] [**-M** *home-perm*] [**-p** *password*]
[**-r** *lowuid*..*highuid*] [**-s** *shell*] [**-u** *uid*] *user*

DESCRIPTION

The **useradd** utility adds a user to the system, creating and populating a home directory if necessary. Any skeleton files will be provided for the new user if they exist in the *skel-dir* directory (see the **-k** option). Default values for the base directory, the time of password expiry, the time of account expiry, primary group, the skeleton directory, the range from which the uid will be allocated, and default login shell can be provided in the /etc/usermgmt.conf file, which, if running as root, is created using the built-in defaults if it does not exist.

The first form of the command shown above (using the **-D** option) sets and displays the defaults for the **useradd** utility.

See user(8) for more information about EXTENSIONS.

-b *base-dir*
Set the default base directory. This is the directory to which the user directory is added, which will be created if the **-m** option is specified and no **-d** option is specified.

-D without any further options, **-D** will show the current defaults which will be used by the **useradd** utility. Together with one of the options shown for the first version of the command, **-D** will set the default to be the new value. See usermgmt.conf(5) for more information.

-e *expiry-time*
Set the time at which the new user accounts will expire. It 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. Time in seconds since the epoch (UTC) is also valid. A value of 0 can be used to disable this feature.

-F Force the user to change their password upon next login.

-f *inactive-time*
Set the time at which passwords for the new user accounts will expire. Also see the **-e** option above.

-g *gid* | *groupname* | =uid
Set the default group for new users.

-k *skel-dir*
Set the skeleton directory in which to find files with which to populate new users' home directories.

-L *login-class*
Set the default login class for new users. See login.conf(5) for more information on user login classes. This option is included if built with EXTENSIONS.

-M *home-perm*
sets the default permissions of the newly created home directory if **-m** is given. The permission is specified as an octal number, with or without a leading zero.

-r *lowuid*..*highuid*
Set the low and high bounds of uid ranges for new users. A new user can only be created if there are uids which can be assigned from one of the free ranges. This option is included if built with `EXTENSIONS`.

-s *shell*
Set the default login shell for new users.

In the second form of the command, after setting any defaults, and then reading values from `/etc/usermgmt.conf`, the following command line options are processed:

-b *base-directory*
Set the base directory name, in which the user's new home directory will be created, should the **-m** option be specified.

-c *comment*
Set the comment field (also, for historical reasons known as the GECOS field) which will be added for the user, and typically will include the user's full name, and, perhaps, contact information for the user.

-d *home-directory*
Set the home directory which will be created and populated for the user, should the **-m** option be specified.

-e *expiry-time*
Set the time at which the current password will expire for new users. It 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. Time in seconds since the epoch (UTC) is also valid. A value of 0 can be used to disable this feature. See `passwd`(5) for more details.

-f *inactive-time*
Set the time at which new user accounts will expire. Also see the **-e** option above.

-G *secondary-group*
Add the user to the secondary group *secondary-group* in the `/etc/group` file.

-g *gid* | *name* | =uid
Give the group name or identifier to be used for the new user's primary group. If this is `=uid`, then a uid and gid will be picked which are both unique and the same, and a line added to `/etc/group` to describe the new group.

-k *skeleton directory*
Give the skeleton directory in which to find files with which to populate the new user's home directory.

-L *login-class*
Set the login class for the user being created. See `login.conf`(5) for more information on user login classes. This option is included if built with `EXTENSIONS`.

-M *home-perm*
sets the permissions of the newly created home directory if **-m** is given. The permission is specified as an octal number, with or without a leading zero.

-m Create a new home directory for the new user.

-o Allow the new user to have a uid which is already in use for another user.

-p *password*
Specify an already-encrypted password for the new user. Encrypted passwords can be generated with `pwhash`(1). The password can be changed later by using `chpass`(1) or `passwd`(1). This option is included if built with `EXTENSIONS`.

-S Allow samba user names with a trailing dollar sign to be added to the system. This option is included if built with `EXTENSIONS`.

-s *shell*
Specify the login shell for the new user.

-u *uid*
Specify a uid for the new user. Boundaries for this value can be preset for all users by using the *range* field in the `/etc/usermgmt.conf` file.

-v Enable verbose mode - explain the commands as they are executed. This option is included if built with `EXTENSIONS`.

Once the information has been verified, **useradd** uses `pwd_mkdb`(8) to update the user database. This is run in the background, and, at very large sites could take several minutes. Until this update is completed, the password file is unavailable for other updates and the new information is not available to programs.

EXIT STATUS

The **useradd** utility exits 0 on success, and >0 if an error occurs.

FILES

`/etc/usermgmt.conf`
`/etc/skel/*`
`/etc/login.conf`

SEE ALSO

`chpass`(1), `passwd`(1), `pwhash`(1), `group`(5), `login.conf`(5), `passwd`(5), `usermgmt.conf`(5), `pwd_mkdb`(8), `user`(8), `userdel`(8), `usermod`(8)

HISTORY

The **useradd** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **useradd** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

Support for setting permissions of home directories was added by Hubert Feyrer.

NAME

userdel — remove a user from the system

SYNOPSIS

userdel -D [**-p** *preserve-value*]
userdel [**-rSv**] [**-p** *preserve-value*] *user*

DESCRIPTION

The **userdel** utility removes a user from the system, optionally removing that user's home directory and any subdirectories.

Default values are taken from the information provided in the `/etc/usermgmt.conf` file, which, if running as root, is created using the built-in defaults if it does not exist.

The first form of the command shown above (using the **-D** option) sets and displays the defaults for the **userdel** utility.

See `user`(8) for more information about `EXTENSIONS`.

-D Without any further options, **-D** will show the current defaults which will be used by the **userdel** utility. Together with one of the options shown for the first version of the command, **-D** will set the default to be the new value. This option is included if built with `EXTENSIONS`.

-p *preserve-value*
Set the preservation value. If this value is one of `true`, `yes`, or a non-zero number, then the user login information will be preserved. This option is included if built with `EXTENSIONS`.

In the second form of the command, after setting any defaults, and then reading values from `/etc/usermgmt.conf`, the following command line options are processed:

-p *preserve-value*
Preserve the user information in the password file, but do not allow the user to login, by switching the password to an "impossible" one, and by setting the user's shell to the `nologin`(8) program. This option can be helpful in preserving a user's files for later use by members of that person's group after the user has moved on. This value can also be set in the `/etc/usermgmt.conf` file, using the `preserve` field. If the field has any of the values `true`, `yes`, or a non-zero number, then user information preservation will take place. This option is included if built with `EXTENSIONS`.

-r Remove the user's home directory, any subdirectories, and any files and other entries in them.

-S Allow a samba user name (with a trailing dollar sign) to be deleted. This option is included if built with `EXTENSIONS`.

-v Perform any actions in a verbose manner. This option is included if built with `EXTENSIONS`.

Once the information has been verified, **userdel** uses `pwd_mkdb`(8) to update the user database. This is run in the background, and, at very large sites could take several minutes. Until this update is completed, the password file is unavailable for other updates and the new information is not available to programs.

EXIT STATUS

The **userdel** utility exits 0 on success, and >0 if an error occurs.

FILES

`/etc/usermgmt.conf`

SEE ALSO

`passwd`(5), `usermgmt.conf`(5), `group`(8), `nologin`(8), `pwd_mkdb`(8), `user`(8), `useradd`(8)

HISTORY

The **`userdel`** utility first appeared in NetBSD 1.5. It is based on the *`addnerd`* package by the same author.

AUTHORS

The **`userdel`** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

userinfo — displays user information

SYNOPSIS

userinfo [**-e**] *user*

DESCRIPTION

The **userinfo** utility retrieves the user information from the system. The **userinfo** utility is only available if built with EXTENSIONS. See user(8) for more information.

The following command line option is recognised:

-e Return 0 if the user exists, and non-zero if the user does not exist, on the system. No information is displayed. This form of the command is useful for scripts which need to check whether a particular user name or uid is already in use on the system.

The *user* argument may either be a user's name, or a uid.

EXIT STATUS

The **userinfo** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

passwd(5), group(8), user(8), useradd(8), userdel(8)

HISTORY

The **userinfo** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **userinfo** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

usermod — modify user login information

SYNOPSIS

usermod [**-FmoSv**] [**-C** *yes/no*] [**-c** *comment*] [**-d** *home-dir*] [**-e** *expiry-time*]
[**-f** *inactive-time*] [**-G** *secondary-group*] [**-g** *gid* | *name* | =uid]
[**-L** *login-class*] [**-l** *new-login*] [**-p** *password*] [**-s** *shell*] [**-u** *uid*]
user

DESCRIPTION

The **usermod** utility modifies user login information on the system.

Default values are taken from the information provided in the `/etc/usermgmt.conf` file, which, if running as root, is created using the built-in defaults if it does not exist.

See `user`(8) for more information about `EXTENSIONS`.

After setting any defaults, and then reading values from `/etc/usermgmt.conf`, the following command line options are processed:

-C *yes/no*
Enable user accounts to be temporary locked/closed. The *yes/no* operand can be given as "*yes*" to lock the account or "*no*" to unlock the account.

-c *comment*
Set the comment field (also, for historical reasons known as the GECOS field) for the user. The comment field will typically include the user's full name and, perhaps, contact information for the user.

-d *home-directory*
Set the home directory. without populating it; if the **-m** option is specified, tries to move the old home directory to *home-directory*.

-e *expiry-time*
Set the time at which the account expires. This can be used to implement password aging. It 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. Time in seconds since the epoch (UTC) is also valid. A value of 0 can be used to disable this feature. This value can be preset for all users using the *expire* field in the `/etc/usermgmt.conf` file. See `usermgmt.conf`(5) for more details.

-F Force the user to change their password upon next login.

-f *inactive-time*
Set the time at which the password expires. See the **-e** option.

-G *secondary-group*
Specify a secondary group to which the user will be added in the `/etc/group` file.

-g *gid* | *name* | =uid
Give the group name or identifier to be used for the user's primary group. If this is `=uid`, then a uid and gid will be picked which are both unique and the same, and a line will be added to `/etc/group` to describe the new group. This value can be preset for all users by using the *gid* field in the `/etc/usermgmt.conf` file. See `usermgmt.conf`(5) for more details.

-L *login-class*
Set the login class for the user. See `login.conf`(5) for more information on user login classes. This value can be preset for all users by using the *class* field in the `/etc/usermgmt.conf`

file. See `usermgmt.conf`(5) for more details. This option is included if built with `EXTENSIONS`.

-l *new-user*
: Give the new user name. It can consist of alphanumeric characters and the characters '`.`', '`-`', and '`_`'.

-m Move the home directory from its old position to the new one. If **-d** is not specified, the *new-user* argument of the **-l** option is used; one of **-d** and **-l** is needed.

-o Allow duplicate uids to be given.

-p *password*
: Specify an already-encrypted password for the user. This password can then be changed by using the `chpass`(1) utility. This value can be preset for all users by using the *password* field in the `/etc/usermgmt.conf` file. See `usermgmt.conf`(5) for more details. This option is included if built with `EXTENSIONS`.

-S Allow samba user names with a trailing dollar sign to be modified. This option is included if built with `EXTENSIONS`.

-s *shell*
: Specify the login shell for the user. This value can be preset for all users by using the *shell* field in the `/etc/usermgmt.conf` file. See `usermgmt.conf`(5) for more details.

-u *uid*
: Specify a new uid for the user. Boundaries for this value can be preset for all users by using the *range* field in the `/etc/usermgmt.conf` file. See `usermgmt.conf`(5) for more details.

-v Enable verbose mode - explain the commands as they are executed. This option is included if built with `EXTENSIONS`.

Once the information has been verified, **usermod** uses `pwd_mkdb`(8) to update the user database. This is run in the background. At very large sites this can take several minutes. Until this update is completed, the password file is unavailable for other updates and the new information is not available to programs.

EXIT STATUS

The **usermod** utility exits 0 on success, and >0 if an error occurs.

FILES

`/etc/usermgmt.conf`

SEE ALSO

`chpass`(1), `group`(5), `passwd`(5), `usermgmt.conf`(5), `pwd_mkdb`(8), `user`(8), `useradd`(8), `userdel`(8)

HISTORY

The **usermod** utility first appeared in NetBSD 1.5. It is based on the *addnerd* package by the same author.

AUTHORS

The **usermod** utility was written by Alistair G. Crooks ⟨agc@NetBSD.org⟩.

NAME

utmp_update — update utmpx database

SYNOPSIS

utmp_update *utmpx_entry*

DESCRIPTION

utmp_update is a helper program to allow a user to update his own utmpx(5) entry. **utmp_update** does some consistency checks on the strvis(3)-encoded *utmpx_entry* and then updates the utmpx(5) database of currently logged in users.

utmp_update should not be called directly, but will normally only be called by pututxline(3) if the privileges of the calling user are not sufficient.

EXIT STATUS

utmp_update returns 0 on success, and 1 if an error occurred.

SEE ALSO

pututxline(3), utmpx(5)

NAME

veriexec — file integrity subsystem

DESCRIPTION

Veriexec is an in-kernel, real-time, file-system independent, file integrity subsystem. It can be used for a variety of purposes, including defense against trojaned binaries, indirect attacks via third-party remote file-systems, and malicious configuration file corruption.

CONFIGURATION

Signatures Database

Veriexec requires a signatures database -- a list of monitored files, along with their digital fingerprint and (optionally) access modes. The format of this file is described by `veriexec`(5).

NetBSD provides a tool, `veriexecgen`(8), for generating the signatures database. Example usage:

```
# veriexecgen
```

Although it should be loaded on system boot (see "RC Configuration" below), this list can be loaded manually using `veriexecctl`(8):

```
# veriexecctl load
```

Kernel Configuration

Veriexec requires a pseudo-device to run:

```
pseudo-device veriexec 1
```

Additionally, one or more options for digital fingerprint algorithm support:

```
options VERIFIED_EXEC_FP_SHA256
options VERIFIED_EXEC_FP_SHA512
```

Some kernels already enable *Veriexec* by default. See your kernel's config file for more information.

RC Configuration

Veriexec also allows loading signatures and setting the strict level (see below) during the boot process using the following variables set in `rc.conf`(5):

```
veriexec=YES
veriexec_strict=1 # IDS mode
```

STRICT LEVELS

Veriexec can operate in four modes, also referred to as strict levels:

Learning mode (strict level 0)
: The only level at which the fingerprint tables can be modified, this level is used to help fine-tune the signature database. No enforcement is made, and verbose information is provided (fingerprint matches and mismatches, file removals, incorrect access, etc.).

IDS mode (strict level 1)
: IDS (intrusion detection system) mode provides an adequate level of integrity for the files it monitors. Implications:

- Monitored files cannot be removed
- If raw disk access is granted to a disk with monitored files on it, all monitored files' fingerprints will be invalidated

- Access to files with mismatched fingerprints is denied
- Write access to monitored files is allowed
- Access type is not enforced

IPS mode (strict level 2)
IPS (intrusion prevention system) mode provides a high level of integrity for the files it monitors. Implications:

- All implications of IDS mode
- Write access to monitored files is denied
- Access type is enforced
- Raw disk access to disk devices with monitored files on them is denied
- Execution of non-monitored files is denied
- Write access to kernel memory via `/dev/mem` and `/dev/kmem` is denied

Lockdown mode (strict level 3)
Lockdown mode provides high assurance integrity for the entire system. Implications:

- All implications of IPS mode
- Access to non-monitored files is denied
- Write access to files is allowed only if the file was opened before the strict level was raised to this mode
- Creation of new files is denied
- Raw access to system disks is denied

RUNTIME INFORMATION

Veriexec exports runtime information that may be useful for various purposes.

It reports the currently supported fingerprinting algorithms, for example:

```
# /sbin/sysctl kern.veriexec.algorithms
kern.veriexec.algorithms = RMD160 SHA256 SHA384 SHA512 SHA1 MD5
```

It reports the current verbosity and strict levels, for example:

```
# /sbin/sysctl kern.veriexec.{verbose,strict}
kern.veriexec.verbose = 0
kern.veriexec.strict = 1
```

It reports a summary of currently loaded files and the mount-points they're on, for example:

```
# /sbin/sysctl kern.veriexec.count
kern.veriexec.count.table0.mntpt = /
kern.veriexec.count.table0.fstype = ffs
kern.veriexec.count.table0.nentries = 33
```

Other information may be retrieved using `veriexecctl`(8).

SEE ALSO

`options`(4), `veriexec`(5), `sysctl`(7), `sysctl`(8), `veriexecctl`(8), `veriexecgen`(8)

AUTHORS

Elad Efrat ⟨elad@NetBSD.org⟩

NAME

veriexecctl — manage the *Veriexec* subsystem

SYNOPSIS

veriexecctl [**-ekv**] **load** [file]
veriexecctl **delete** *file* | *mount_point*
veriexecctl **dump**
veriexecctl **flush**
veriexecctl **query** *file*

DESCRIPTION

The **veriexecctl** command is used to manipulate *Veriexec*, the NetBSD file integrity subsystem.

Commands

load [file]
Load the fingerprint entries contained in *file*, if specified, or the default signatures file otherwise.

This operation is only allowed in learning mode (strict level zero).

The following flags are allowed with this command:

-e Evaluate fingerprint on load, as opposed to when the file is accessed.

-k Keep the filenames in the entry for more accurate logging.

delete *file* | *mount_point*
Delete either a single entry *file* or all entries on *mount_point* from being monitored by *Veriexec*.

dump Dump the *Veriexec* database from the kernel. Only entries that have the filename will be presented.

This can be used to recover a lost database:

```
# veriexecctl dump > /etc/signatures
```

flush
Delete all entries in the *Veriexec* database.

query *file*
Query *Veriexec* for information associated with *file*: Filename, mount, fingerprint, fingerprint algorithm, evaluation status, and entry type.

FILES

/dev/veriexec	*Veriexec* pseudo-device
/etc/signatures	default signatures file

SEE ALSO

veriexec(4), veriexec(5), security(8), veriexec(8), veriexecgen(8)

HISTORY

veriexecctl first appeared in NetBSD 2.0.

AUTHORS

Brett Lymn ⟨blymn@NetBSD.org⟩
Elad Efrat ⟨elad@NetBSD.org⟩

NOTES

The kernel is expected to have the “veriexec” pseudo-device.

NAME

veriexecgen — generate fingerprints for Veriexec

SYNOPSIS

veriexecgen [**−AaDrSTvW**] [**−d** dir] [**−o** fingerprintdb] [**−p** prefix] [**−t** *algorithm*]
veriexecgen [**−h**]

DESCRIPTION

veriexecgen can be used to create a fingerprint database for use with *Veriexec*.

If no command line arguments were specified, **veriexecgen** will resort to default operation, implying **−D −o** */etc/signatures* **−t** *sha256*.

If the output file already exists, **veriexecgen** will save a backup copy in the same file only with a ".old" suffix.

The following options are available:

−A Append to the output file, don't overwrite it.

−a Add fingerprints for non-executable files as well.

−D Search system directories, `/bin`, `/sbin`, `/usr/bin`, `/usr/sbin`, `/lib`, `/usr/lib`, `/libexec`, and `/usr/libexec`.

−d *dir* Scan for files in *dir*. Multiple uses of this flag can specify more than one directory.

−h Display the help screen.

−o *fingerprintdb*
Save the generated fingerprint database to *fingerprintdb*.

−p *prefix*
When storing files in the fingerprint database, store the full pathnames of files with the leading "prefix" of the filenames removed.

−r Scan recursively.

−S Set the immutable flag on the created signatures file when done writing it.

−T Put a timestamp on the generated file.

−t *algorithm*
Use *algorithm* for the fingerprints. Must be one of "md5", "sha1", "sha256", "sha384", "sha512", or "rmd160".

−v Verbose mode. Print messages describing what operations are being done.

−W By default, **veriexecgen** will exit when an error condition is encountered. This option will treat errors such as not being able to follow a symbolic link, not being able to find the real path for a directory entry, or not being able to calculate a hash of an entry as a warning, rather than an error. If errors are treated as warnings, **veriexecgen** will continue processing. The default behaviour is to treat errors as fatal.

FILES

`/etc/signatures`

EXAMPLES

Fingerprint files in the common system directories using the default hashing algorithm "sha256" and save to the default fingerprint database in `/etc/signatures`:

```
# veriexecgen
```

Fingerprint files in `/etc`, appending to the default fingerprint database:

```
# veriexecgen -A -d /etc
```

Fingerprint files in `/path/to/somewhere using` "rmd160" as the hashing algorithm, saving to `/etc/somewhere.fp`:

```
# veriexecgen -d /path/to/somewhere -t rmd160 -o /etc/somewhere.fp
```

SEE ALSO

`veriexec`(4), `veriexec`(5), `security`(8), `veriexec`(8), `veriexecctl`(8)

NAME

verify − Postfix address verification server

SYNOPSIS

verify [generic Postfix daemon options]

DESCRIPTION

The **verify**(8) address verification server maintains a record of what recipient addresses are known to be deliverable or undeliverable.

Addresses are verified by injecting probe messages into the Postfix queue. Probe messages are run through all the routing and rewriting machinery except for final delivery, and are discarded rather than being deferred or bounced.

Address verification relies on the answer from the nearest MTA for the specified address, and will therefore not detect all undeliverable addresses.

The **verify**(8) server is designed to run under control by the Postfix master server. It maintains an optional persistent database. To avoid being interrupted by "postfix stop" in the middle of a database update, the process runs in a separate process group.

The **verify**(8) server implements the following requests:

update *address status text*
: Update the status and text of the specified address.

query *address*
: Look up the *status* and *text* for the specified address. If the status is unknown, a probe is sent and an "in progress" status is returned.

SECURITY

The address verification server is not security-sensitive. It does not talk to the network, and it does not talk to local users. The verify server can run chrooted at fixed low privilege.

The address verification server can be coerced to store unlimited amounts of garbage. Limiting the cache size trades one problem (disk space exhaustion) for another one (poor response time to client requests).

DIAGNOSTICS

Problems and transactions are logged to **syslogd**(8).

BUGS

The address verification service is suitable only for sites that handle a low mail volume. Verification probes add additional traffic to the mail queue and perform poorly under high load. Servers may blacklist sites that probe excessively, or that probe excessively for non-existent recipient addresses.

If the persistent database ever gets corrupted then the world comes to an end and human intervention is needed. This violates a basic Postfix principle.

CONFIGURATION PARAMETERS

Changes to **main.cf** are not picked up automatically, as **verify**(8) processes are persistent. Use the command "**postfix reload**" after a configuration change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

CACHE CONTROLS

address_verify_map (empty)
: Optional lookup table for persistent address verification status storage.

address_verify_sender (postmaster)
The sender address to use in address verification probes.

address_verify_positive_expire_time (31d)
The time after which a successful probe expires from the address verification cache.

address_verify_positive_refresh_time (7d)
The time after which a successful address verification probe needs to be refreshed.

address_verify_negative_cache (yes)
Enable caching of failed address verification probe results.

address_verify_negative_expire_time (3d)
The time after which a failed probe expires from the address verification cache.

address_verify_negative_refresh_time (3h)
The time after which a failed address verification probe needs to be refreshed.

PROBE MESSAGE ROUTING CONTROLS

By default, probe messages are delivered via the same route as regular messages. The following parameters can be used to override specific message routing mechanisms.

address_verify_relayhost ($relayhost)
Overrides the relayhost parameter setting for address verification probes.

address_verify_transport_maps ($transport_maps)
Overrides the transport_maps parameter setting for address verification probes.

address_verify_local_transport ($local_transport)
Overrides the local_transport parameter setting for address verification probes.

address_verify_virtual_transport ($virtual_transport)
Overrides the virtual_transport parameter setting for address verification probes.

address_verify_relay_transport ($relay_transport)
Overrides the relay_transport parameter setting for address verification probes.

address_verify_default_transport ($default_transport)
Overrides the default_transport parameter setting for address verification probes.

MISCELLANEOUS CONTROLS

config_directory (see ’postconf -d’ output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

smtpd(8), Postfix SMTP server
cleanup(8), enqueue Postfix message
postconf(5), configuration parameters
syslogd(5), system logging

README FILES

Use "**postconf readme_directory**" or "**postconf html_directory**" to locate this information.
ADDRESS_VERIFICATION_README, address verification howto

LICENSE

The Secure Mailer license must be distributed with this software.

HISTORY

This service was introduced with Postfix version 2.1.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

NAME

verify_krb5_conf — checks krb5.conf for obvious errors

SYNOPSIS

verify_krb5_conf *[config-file]*

DESCRIPTION

verify_krb5_conf reads the configuration file `krb5.conf`, or the file given on the command line, and parses it, thereby verifying that the syntax is not correctly wrong.

If the file is syntactically correct, **verify_krb5_conf** tries to verify that the contents of the file is of relevant nature.

ENVIRONMENT

`KRB5_CONFIG` points to the configuration file to read.

FILES

`/etc/krb5.conf` Kerberos 5 configuration file

DIAGNOSTICS

Possible output from **verify_krb5_conf** include:

<path>: failed to parse <something> as size/time/number/boolean
    Usually means that <something> is misspelled, or that it contains weird characters. The parsing done by **verify_krb5_conf** is more strict than the one performed by libkrb5, so strings that work in real life might be reported as bad.

<path>: host not found (<hostname>)
    Means that <path> is supposed to point to a host, but it can't be recognised as one.

<path>: unknown or wrong type
    Means that <path> is either a string when it should be a list, vice versa, or just that **verify_krb5_conf** is confused.

<path>: unknown entry
    Means that <string> is not known by .

SEE ALSO

`krb5.conf`(5)

BUGS

Since each application can put almost anything in the config file, it's hard to come up with a watertight verification process. Most of the default settings are sanity checked, but this does not mean that every problem is discovered, or that everything that is reported as a possible problem actually is one. This tool should thus be used with some care.

It should warn about obsolete data, or bad practice, but currently doesn't.

NAME

vipw — edit the password file

SYNOPSIS

vipw [**-d** *directory*]

DESCRIPTION

vipw edits the password file after setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already locked for editing by another user, **vipw** will ask you to try again later. The default editor for **vipw** is `vi`(1).

vipw performs a number of consistency checks on the password entries, and will not allow a password file with a "mangled" entry to be installed. If **vipw** rejects the new password file, the user is prompted to re-enter the edit session.

Once the information has been verified, **vipw** uses `pwd_mkdb`(8) to update the user database. This is run in the background, and, at very large sites could take several minutes. Until this update is completed, the password file is unavailable for other updates and the new information is not available to programs.

The options are as follows:

-d *directory*
: Change the root directory of the password file from "`/`" to *directory*.

If a **vipw** session is killed it may leave "`/etc/ptmp`", which will cause future **vipw** executions to fail with "`vipw: the passwd file is busy`", until it is removed.

ENVIRONMENT

If the following environment variable exists it will be used by **vipw**:

`EDITOR` The editor specified by the string `EDITOR` will be invoked instead of the default editor `vi`(1).

FILES

`/etc/master.passwd`	The current password file.
`/etc/ptmp`	Temporary copy of the password file used while editing.

SEE ALSO

`chpass`(1), `passwd`(1), `pwhash`(1), `passwd`(5), `passwd.conf`(5), `pwd_mkdb`(8), `user`(8)

HISTORY

The **vipw** command appeared in 4.0BSD.

NAME

virecover — report recovered vi edit sessions

SYNOPSIS

`/usr/libexec/virecover`

DESCRIPTION

The **virecover** utility sends emails to users who have `vi`(1) recovery files.

This email gives the name of the file that was saved for recovery and instructions for recovering most, if not all, of the changes to the file. This is done by using the **−r** option with `vi`(1). See the **−r** option in `vi`(1) for details.

If the backup files have the execute bit set or are zero length, then they have not been modified, so **virecover** deletes them to clean up. **virecover** also removes recovery files that are corrupted, zero length, or do not have a corresponding backup file.

virecover is normally run automatically at boot time using `/etc/rc.d/virecover`.

FILES

`/var/tmp/vi.recover/recover.*`	`vi`(1) recovery files
`/var/tmp/vi.recover/vi.*`	`vi`(1) editor backup files

SEE ALSO

`vi`(1), `rc.conf`(5)

HISTORY

This script, previously known as **recover.script**, is from nvi and was added to NetBSD in 1996. It was renamed in 2001.

AUTHORS

This man page was written by Jeremy C. Reed ⟨reed@reedmedia.net⟩.

NAME

virtual − Postfix virtual domain mail delivery agent

SYNOPSIS

virtual [generic Postfix daemon options]

DESCRIPTION

The **virtual**(8) delivery agent is designed for virtual mail hosting services. Originally based on the Postfix **local**(8) delivery agent, this agent looks up recipients with map lookups of their full recipient address, instead of using hard-coded unix password file lookups of the address local part only.

This delivery agent only delivers mail. Other features such as mail forwarding, out-of-office notifications, etc., must be configured via virtual_alias maps or via similar lookup mechanisms.

MAILBOX LOCATION

The mailbox location is controlled by the **virtual_mailbox_base** and **virtual_mailbox_maps** configuration parameters (see below). The **virtual_mailbox_maps** table is indexed by the recipient address as described under TABLE SEARCH ORDER below.

The mailbox pathname is constructed as follows:

$virtual_mailbox_base/$virtual_mailbox_maps(*recipient***)**

where *recipient* is the full recipient address.

UNIX MAILBOX FORMAT

When the mailbox location does not end in **/**, the message is delivered in UNIX mailbox format. This format stores multiple messages in one textfile.

The **virtual**(8) delivery agent prepends a "**From** *sender time_stamp*" envelope header to each message, prepends a **Delivered-To:** message header with the envelope recipient address, prepends an **X-Original-To:** header with the recipient address as given to Postfix, prepends a **Return-Path:** message header with the envelope sender address, prepends a **>** character to lines beginning with "**From** ", and appends an empty line.

The mailbox is locked for exclusive access while delivery is in progress. In case of problems, an attempt is made to truncate the mailbox to its original length.

QMAIL MAILDIR FORMAT

When the mailbox location ends in **/**, the message is delivered in qmail **maildir** format. This format stores one message per file.

The **virtual**(8) delivery agent prepends a **Delivered-To:** message header with the final envelope recipient address, prepends an **X-Original-To:** header with the recipient address as given to Postfix, and prepends a **Return-Path:** message header with the envelope sender address.

By definition, **maildir** format does not require application-level file locking during mail delivery or retrieval.

MAILBOX OWNERSHIP

Mailbox ownership is controlled by the **virtual_uid_maps** and **virtual_gid_maps** lookup tables, which are indexed with the full recipient address. Each table provides a string with the numerical user and group ID, respectively.

The **virtual_minimum_uid** parameter imposes a lower bound on numerical user ID values that may be specified in any **virtual_uid_maps**.

CASE FOLDING

All delivery decisions are made using the full recipient address, folded to lower case. See also the next section for a few exceptions with optional address extensions.

TABLE SEARCH ORDER

Normally, a lookup table is specified as a text file that serves as input to the **postmap**(1) command. The result, an indexed file in **dbm** or **db** format, is used for fast searching by the mail system.

The search order is as follows. The search stops upon the first successful lookup.

- When the recipient has an optional address extension the *user+extension@domain.tld* address is looked up first.

  With Postfix versions before 2.1, the optional address extension is always ignored.
- The *user@domain.tld* address, without address extension, is looked up next.
- Finally, the recipient *@domain* is looked up.

When the table is provided via other means such as NIS, LDAP or SQL, the same lookups are done as for ordinary indexed files.

Alternatively, a table can be provided as a regular-expression map where patterns are given as regular expressions. In that case, only the full recipient address is given to the regular-expression map.

SECURITY

The **virtual**(8) delivery agent is not security sensitive, provided that the lookup tables with recipient user/group ID information are adequately protected. This program is not designed to run chrooted.

The **virtual**(8) delivery agent disallows regular expression substitution of $1 etc. in regular expression lookup tables, because that would open a security hole.

The **virtual**(8) delivery agent will silently ignore requests to use the **proxymap**(8) server. Instead it will open the table directly. Before Postfix version 2.2, the virtual delivery agent will terminate with a fatal error.

STANDARDS

RFC 822 (ARPA Internet Text Messages)

DIAGNOSTICS

Mail bounces when the recipient has no mailbox or when the recipient is over disk quota. In all other cases, mail for an existing recipient is deferred and a warning is logged.

Problems and transactions are logged to **syslogd**(8). Corrupted message files are marked so that the queue manager can move them to the **corrupt** queue afterwards.

Depending on the setting of the **notify_classes** parameter, the postmaster is notified of bounces and of other trouble.

BUGS

This delivery agent supports address extensions in email addresses and in lookup table keys, but does not propagate address extension information to the result of table lookup.

Postfix should have lookup tables that can return multiple result attributes. In order to avoid the inconvenience of maintaining three tables, use an LDAP or MYSQL database.

CONFIGURATION PARAMETERS

Changes to **main.cf** are picked up automatically, as **virtual**(8) processes run for only a limited amount of time. Use the command "**postfix reload**" to speed up a change.

The text below provides only a parameter summary. See **postconf**(5) for more details including examples.

MAILBOX DELIVERY CONTROLS

virtual_mailbox_base (empty)
A prefix that the **virtual**(8) delivery agent prepends to all pathname results from $virtual_mailbox_maps table lookups.

virtual_mailbox_maps (empty)
Optional lookup tables with all valid addresses in the domains that match $virtual_mailbox_domains.

virtual_minimum_uid (100)
The minimum user ID value that the **virtual**(8) delivery agent accepts as a result from $virtual_uid_maps table lookup.

virtual_uid_maps (empty)
Lookup tables with the per-recipient user ID that the **virtual**(8) delivery agent uses while writing to the recipient's mailbox.

virtual_gid_maps (empty)
Lookup tables with the per-recipient group ID for **virtual**(8) mailbox delivery.

Available in Postfix version 2.0 and later:

virtual_mailbox_domains ($virtual_mailbox_maps)
Postfix is final destination for the specified list of domains; mail is delivered via the $virtual_transport mail delivery transport.

virtual_transport (virtual)
The default mail delivery transport and next-hop destination for final delivery to domains listed with $virtual_mailbox_domains.

LOCKING CONTROLS

virtual_mailbox_lock (see 'postconf -d' output)
How to lock a UNIX-style **virtual**(8) mailbox before attempting delivery.

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.

RESOURCE AND RATE CONTROLS

virtual_destination_concurrency_limit ($default_destination_concurrency_limit)
The maximal number of parallel deliveries to the same destination via the virtual message delivery transport.

virtual_destination_recipient_limit ($default_destination_recipient_limit)
The maximal number of recipients per delivery via the virtual message delivery transport.

virtual_mailbox_limit (51200000)
The maximal size in bytes of an individual mailbox or maildir file, or zero (no limit).

MISCELLANEOUS CONTROLS

config_directory (see 'postconf -d' output)
The default location of the Postfix main.cf and master.cf configuration files.

daemon_timeout (18000s)
How much time a Postfix daemon process may take to handle a request before it is terminated by a built-in watchdog timer.

delay_logging_resolution_limit (2)
The maximal number of digits after the decimal point when logging sub-second delay values.

ipc_timeout (3600s)
The time limit for sending or receiving information over an internal communication channel.

max_idle (100s)
The maximum amount of time that an idle Postfix daemon process waits for an incoming connection before terminating voluntarily.

max_use (100)
The maximal number of incoming connections that a Postfix daemon process will service before terminating voluntarily.

process_id (read-only)
The process ID of a Postfix command or daemon process.

process_name (read-only)
The process name of a Postfix command or daemon 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".

SEE ALSO

qmgr(8), queue manager
bounce(8), delivery status reports
postconf(5), configuration parameters
syslogd(8), system logging

README_FILES

Use "**postconf readme_directory**" or
"**postconf html_directory**" to locate this information.
VIRTUAL_README, domain hosting howto

LICENSE

The Secure Mailer license must be distributed with this software.

HISTORY

This delivery agent was originally based on the Postfix local delivery agent. Modifications mainly consisted of removing code that either was not applicable or that was not safe in this context: aliases, ˜user/.forward files, delivery to "|command" or to /file/name.

The **Delivered-To:** message header appears in the **qmail** system by Daniel Bernstein.

The **maildir** structure appears in the **qmail** system by Daniel Bernstein.

AUTHOR(S)

Wietse Venema
IBM T.J. Watson Research
P.O. Box 704
Yorktown Heights, NY 10598, USA

Andrew McNamara
andrewm@connect.com.au
connect.com.au Pty. Ltd.

Level 3, 213 Miller St
North Sydney 2060, NSW, Australia

NAME

vnconfig — configure vnode disks

SYNOPSIS

vnconfig [**−crvz**] [**−f** *disktab*] [**−t** *typename*] *vnode_disk regular_file* [*geomspec*]
vnconfig −u [**−Fv**] *vnode_disk*
vnconfig −l [*vnode_disk*]

DESCRIPTION

The **vnconfig** command configures vnode pseudo disk devices. It will associate the vnode disk *vnode_disk* with the regular file *regular_file* allowing the latter to be accessed as though it were a disk. Hence a regular file within the filesystem can be used for swapping or can contain a filesystem that is mounted in the name space. The *vnode_disk* is a special file of raw partition or name of vnode disk like `vnd0`.

Options indicate an action to be performed:

−c Configures the device. If successful, references to *vnode_disk* will access the contents of *regular_file*.

If *geomspec* is specified, the vnode device will emulate the specified disk geometry. The format of the *geomspec* argument is:

secsize/nsectors/ntracks/ncylinders

If geometry is not specified, the kernel will choose a default based on 1MB cylinders. *secsize* is the number of bytes per sector. It must be an even multiple of 512. *nsectors* is the number of sectors per track. *ntracks* is the number of tracks per cylinder. *ncylinders* is the number of cylinders in the device.

−F Force unconfiguration if the device is in use. Does not imply **−u**.

−f *disktab*
Specifies that the **−t** option should look up in *disktab* instead of in `/etc/disktab`.

−l List the vnd devices and indicate which ones are in use. If a specific *vnode_disk* is given, then only that will be described.

−t *typename*
If configuring the device, look up *typename* in `/etc/disktab` and use the geometry specified in the entry. This option and the *geomspec* argument are mutually exclusive.

−r Configure the device as read-only.

−u Unconfigures the device.

−v Print messages to stdout describing actions taken.

−z Assume that *regular_file* is a compressed disk image in cloop2 format, and configure it read-only. See the `vndcompress`(1) manpage on how to create such an image.

If no action option is given, **−c** is assumed.

FILES

`/dev/rvnd??`

```
/dev/vnd??
/etc/disktab
```

EXAMPLES

```
vnconfig vnd0 /tmp/diskimage
```

or

```
vnconfig /dev/rvnd0c /tmp/diskimage
```

Configures the vnode disk `vnd0`. Please note that use of the second form of the command is discouraged because it requires knowledge of the raw partition which varies between architectures.

```
vnconfig vnd0 /tmp/floppy.img 512/18/2/80
```

Configures the vnode disk `vnd0` emulating the geometry of 512 bytes per sector, 18 sectors per track, 2 tracks per cylinder, and 80 cylinders total.

```
vnconfig -t floppy vnd0 /tmp/floppy.img
```

Configures the vnode disk `vnd0` using the geometry specified in the `floppy` entry in `/etc/disktab`.

```
vnconfig -u vnd0
```

Unconfigures the `vnd0` device.

SEE ALSO

`opendisk`(3), `vnd`(4), `mount`(8), `swapctl`(8), `umount`(8)

HISTORY

The **`vnconfig`** command appeared in NetBSD 1.0.

BUGS

This command should really be named **`vndconfig`**.

NAME

w95boot — procedure for booting NetBSD/i386 from Windows 95

DESCRIPTION

dosboot(8) is a program that can boot NetBSD from DOS. However, it must be run under real DOS, and not an emulated DOS session. As a result, it can not be directly run from Windows 95. Windows does, however, have the ability to run a program in "DOS Mode", which is to say, Windows can reboot itself into stand-alone DOS and execute a program. One difficulty of this method is that Windows expects the program to exit, so it can restore the system to boot into Windows. Since dosboot(8) takes over the computer, Windows will be unable to clean up after it, and as a result the next attempt to start Windows will also result in booting NetBSD.

This manual page describes how to set up Windows to boot NetBSD, in such a way as to avoid the problem described above. It assumes that Windows is the default OS on your multiboot system, since if NetBSD were the default, you could simply use the Reboot option of the Windows shutdown menu.

Begin by creating the directory C:\NetBSD on your Windows system. You may use another name if you prefer; just be sure to edit the paths in the instructions below accordingly. In that directory, place a copy of DOSBOOT.COM, which can be found on your NetBSD system as /usr/mdec/dosboot.com. You can use ftp(1) or the msdosfs file system (see mount_msdos(8)) to transfer the file from NetBSD to Windows.

Next create the script GoNetBSD.BAT in that directory, containing the following lines:

```
@ECHO OFF
IF EXIST C:\NetBSD\BOOTED GOTO towin
ECHO >C:\NetBSD\BOOTED
CHOICE /C+ /N /T+,5
C:\NetBSD\DOSBOOT -u
ECHO Error booting NetBSD
CHOICE /C+ /N /T+,60

:towin
DEL C:\NetBSD\BOOTED
ECHO Y >C:\NetBSD\Y
CHOICE /C+ /N /T+,5
C:\WINDOWS\WIN.COM <Y
```

Now, double click the icon for the dosboot(8) program. Windows will bring up a requester entired "Program Requires MS-DOS Mode". Select "Yes" to create a shortcut. Turn off the "Mouse" checkmark on the window that appears and click "OK".

Now right-click the dosboot.pif file that was created. This will be the icon that says dosboot with the MS-DOS logo. Choose "rename" and change the name to GoNetBSD.

Right click this newly-renamed icon again and select "Properties". Click to the "Program" tab. Edit the first text field to "GoNetBSD". Edit the "Cmd line" text field to read "C:\NetBSD\GoNetBSD.BAT". Ensure the "Batch file" field is blank. Click on the "Advanced" button. Make sure "MS-DOS Mode" is checked. Uncheck the "Warn before entering MS-DOS Mode" checkbox if you intend to use the GoNetBSD script from another program, such as the Microsoft Plus! System Agent. Click "OK" to exit the "Advanced Program Settings" window. Click "OK" to exit the "Properties" window.

Now you can click the GoNetBSD.BAT icon to boot into NetBSD.

MICROSOFT PLUS! SYSTEM AGENT

You may also use GoNetBSD.BAT from other programs. For example, it could be scheduled to run under the System Agent (similar to the way Scandisk is invoked, for example) to have the computer automatically switch itself to NetBSD at certain times. In this event, it is necessary to uncheck the ''Warn before entering MS-DOS Mode'' setting as described above.

To set up GoNetBSD to run under System Agent, follow these steps. First, double click the System Agent icon on the taskbar tray. This will open the System Agent window. Select ''Schedule a New Program'' from the ''Program'' menu. In the ''Program'' text field, enter the complete path to the GoNetBSD batch file, for example "C:\NetBSD\GoNetBSD.BAT". In the ''Description'' text field, enter a descriptive name for the scheduled event, such as ''Reboot system under NetBSD''.

Now click the ''When to Run program'' and select the time schedule for the switch to NetBSD. It is advisable to use the ''Wait until I haven't used my computer for...'' feature to prevent the switch if someone is actively using windows at the time the switch is to occur. Now click on the ''Advanced'' button, and set a deadline that is shortly after (say, two minutes) the start time. If you return from NetBSD before the deadline, System Agent will believe that the program did not complete successfully, and switch you back to NetBSD. Setting a very short deadline will prevent this.

Now click ''OK'' in that window and the previous window to get back to the System Agent main window, and then close System Agent. Your system will now automatically switch to NetBSD at the specified time or times.

If you wish to have NetBSD reboot to Windows after some amount of time, the `cron`(8) or `at`(1) programs in conjunction with `reboot`(8) or `shutdown`(8) may be useful.

SEE ALSO

`at`(1), `crontab`(1), `ftp`(1), `cron`(8), `dosboot`(8), `mount_msdos`(8), `reboot`(8), `shutdown`(8)

NAME

wdogctl — Watchdog timer control utility

SYNOPSIS

wdogctl
wdogctl **−d**
wdogctl **−e** [**−A**] [**−p** *seconds*] *timer*
wdogctl **−k** [**−A**] [**−p** *seconds*] *timer*
wdogctl **−t**
wdogctl **−u** [**−A**] [**−p** *seconds*] *timer*
wdogctl **−x** [**−A**] [**−p** *seconds*] *timer*

DESCRIPTION

wdogctl is used to manipulate watchdog timers. Watchdog timers provide a means of ensuring that a system continues to make progress. This is accomplished by use of a timer, provided by either hardware or software; when the timer expires, the watchdog resets the system. In this case of a hardware watchdog timer, this is accomplished by asserting the system's hardware reset signal. In the case of a software watchdog timer, this is accomplished by calling the kernel's normal reboot path. In order to prevent the system from rebooting, something must refresh the timer to prevent it from expiring.

The NetBSD kernel provides three basic modes in which watchdog timers may operate: kernel tickle mode, user tickle mode, and external tickle mode. In kernel tickle mode, a timer in the kernel refreshes the watchdog timer. In user tickle mode, **wdogctl** runs in the background and refreshes the watchdog timer. In kernel tickle mode, progress of the kernel is ensured. In user tickle mode, the ability for user programs to run within a known period of time is ensured. Note that user tickle mode must be used with caution; on a heavily loaded system, the timer may expire accidentally, even though user programs may be making (very slow) progress. A user-mode timer is disarmed (if possible) when the device is closed, unless the timer is activated with the **−x** option.

External-mode watchdogs are similar to user-mode watchdogs, except that the tickle must be done explicitly by a separate invocation of the program with the **−t** option.

In the first two modes, an attempt is made to refresh the watchdog timer in one half the timer's configured period. That is, if the watchdog timer has a period of 30 seconds, a refresh attempt is made every 15 seconds.

If called without arguments, **wdogctl** will list the timers available on the system. When arming a watchdog timer, the *timer* argument is the name of the timer to arm.

Only one timer may be armed at a time; if an attempt is made to arm a timer when one is already armed, an error message will be displayed and no action will be taken.

The options are as follows:

−A	When arming a timer, this flag indicates that an audible alarm is to sound when the watchdog timer expires and resets the system. If the selected timer does not support an audible alarm, this option will be silently ignored.
−d	This flag disarms the currently active timer. Note that not all watchdog timers can be disabled once armed. If the selected timer can not be disabled, an error message will be displayed and the timer will remain armed.
−e	Arm *timer* in external tickle mode.
−k	Arm *timer* in kernel tickle mode.

−p *period* When arming a timer, this flag configures the timer period to *period* seconds. If the specified period is outside the timer's range, an error message will be displayed and no action will be taken.

−t This flag tickles an external mode timer.

−u Arm *timer* in user tickle mode.

−x Arm *timer* in a modified user tickle mode: closing the device will not disarm the timer.

FILES

/dev/watchdog -- the system monitor watchdog timer device

SEE ALSO

i386/elansc(4), i386/gcscpcib(4), i386/geodewdog(4), evbarm/iopwdog(4), ichlpcib(4), ipmi(4), itesio(4), pcweasel(4), swwdog(4)

HISTORY

The **wdogctl** command first appeared in NetBSD 1.5.1.

AUTHORS

The **wdogctl** command and the NetBSD watchdog timer framework were written by Jason R. Thorpe ⟨thorpej@zembu.com⟩, and contributed by Zembu Labs, Inc.

NAME

wiconfig — configure WaveLAN/IEEE devices

SYNOPSIS

wiconfig *interface* [**-Dho**] [**-A** *1*|*2*] [**-a** *access_point_density*] [**-d** *max_data_length*] [**-g** *fragmentation_threshold*] [**-M** *0*|*1*] [**-m** *MAC_address*] [**-R** *1*|*3*] [**-r** *RTS_threshold*] [**-s** *station_name*]

DESCRIPTION

The **wiconfig** command controls the operation of WaveLAN/IEEE wireless networking devices via the wi(4) and awi(4) drivers. The **wiconfig** command can also be used to view the current settings of these parameters and to dump out the values of the card's statistics counters.

Most of the parameters that can be changed relate to the IEEE 802.11 protocol which the WaveLAN implements. This includes the station name, whether the station is operating in ad-hoc (point to point) or BSS (service set) mode, and the network name of a service set to join (IBSS) if BSS mode is enabled.

The *interface* argument given to **wiconfig** should be the logical interface name associated with the WaveLAN/IEEE device (e.g., wi0, wi1, etc.).

OPTIONS

With no extra options, **wiconfig** will display the current settings of the specified WaveLAN/IEEE interface.

The options are as follows:

-A *1*|*2*
Set the authentication type for a specified interface. Permitted values are *1* (Open System Authentication) or *2* (Shared Key Authentication). The default is 1.

-a *access_point_density*
Specify the *access point density* for a given interface. Legal values are 1 (low), 2 (medium), and 3 (high). This setting influences some of the radio modem threshold settings.

-D
This forces the driver to initiate one round of access point scanning. All of the access points found are displayed.

-d *max_data_length*
Set the maximum receive and transmit frame size for a specified interface. The *max data length* can be any number from 256 to 2346. The default is 2304.

-g *fragmentation_threshold*
Set the fragmentation threshold.

-h
Display a short help.

-M *0*|*1*
Enable or disable "microwave oven robustness" on a given interface. This should only be used if needed.

In cases of slow performance where there is a good quality signal but also high levels of noise (i.e., the signal to noise ratio is bad but the signal strength is good), and there is an operating microwave oven in or near the signal path, this option may be of use.

In bad signal-to-noise conditions, the link layer will switch to lower transmit rates. However at lower transmit rates, individual frames take longer to transmit, making them more vulnerable to bursty noise. The option works by enabling data fragmentation in the link layer as the transmit speed lowers in an attempt to shorten the transmit time of each frame so that individual frames are more likely to be transmitted without error.

Note that this does not impact the visible MTU of the link.

-m *MAC_address*
Set the station address for the specified interface. The *MAC address* is specified as a series of six hexadecimal values separated by colons, e.g., 00:60:1d:12:34:56. This programs the new address into the card and updates the interface as well.

-o Print out the statistics counters instead of the card settings. Note that, however, the statistics will only be updated every minute or so.

-R *1*|*3* Enable or disable roaming function on a given interface. The legal values are *1* (Roaming handled by firmware) and *3* (Roaming Disabled). The default is 1.

-r *RTS_threshold*
Set the RTS/CTS threshold for a given interface. This controls the number of bytes used for the RTS/CTS handshake boundary. The *RTS_threshold* can be any value between 0 and 2347. The default is 2347, which indicates RTS/CTS mechanism never to be used.

-s *station_name*
Sets the *station_name* for the specified interface. The *station_name* is used for diagnostic purposes. The Lucent WaveMANAGER software can poll the names of remote hosts.

SEE ALSO

awi(4), wi(4), ifconfig(8)

HISTORY

The **wiconfig** command first appeared in FreeBSD 3.0, as **wicontrol**. It was added to NetBSD 1.5 under its present name.

AUTHORS

The **wiconfig** command was written by Bill Paul ⟨wpaul@ctr.columbia.edu⟩.

NAME

wire-test − test your network interfaces and local IP address

SYNOPSIS

wire-test [*host*]

DESCRIPTION

wire-test is used to find out what amd thinks are the first two network interfaces and network names/numbers used, as well as the IP address used for amd to NFS-mount itself.

If *host* is specified, then **wire-test** will test for the working combinations of NFS protocol and version from the current client to the NFS server *host.* If not specified, *host* defaults to "localhost".

SEE ALSO

amd(8).

‘‘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

wizd — automatically correct typographical errors in man pages

SYNOPSIS

wizd

DESCRIPTION

wizd automatically checks and corrects spelling errors, usage problems with mdoc(7) macros, and other typographical errors in man pages.

wizd is invoked by any cvs(1) commit to a man page. A standalone mode is also available by sending mail to ⟨wiz@NetBSD.org⟩.

SEE ALSO

cvs(1), intro(1), man(1), mdoc(7)

HISTORY

wizd appeared in NetBSD 1.5.

CAVEATS

wizd is not only copyrighted, but also registered.

BUGS

Sleeps sometimes.

NAME

wlanctl — examine IEEE 802.11 wireless LAN client/peer table

SYNOPSIS

wlanctl [**-p**] *interface* [...]
wlanctl [**-p**] **-a**

DESCRIPTION

Use the **wlanctl** utility to print node tables from IEEE 802.11 interfaces. Use the **-a** flag to print the nodes for all interfaces, or list one or more 802.11 interfaces to select their tables for examination. The **-p** flag causes only nodes that do not have encryption enabled to be printed. For example, to examine the node tables for atw0, use:

```
wlanctl atw0
```

wlanctl may print this node table, for example:

```
atw0: mac 00:02:6f:20:f6:2e bss 02:02:6f:20:f6:2e
        node flags 0001<bss>
        ess <netbsd>
        chan 11 freq 2462MHz flags 00a0<cck,2.4GHz>
        capabilities 0022<ibss,short preamble>
        beacon-interval 100 TU tsft 18425852102545544165 us
        rates [1.0] 2.0 5.5 11.0
        assoc-id 0 assoc-failed 0 inactivity 0s
        rssi 161 txseq 10 rxseq 1420
atw0: mac 00:02:2d:2e:3c:f4 bss 02:02:6f:20:f6:2e
        node flags 0000
        ess <netbsd>
        chan 11 freq 2462MHz flags 00a0<cck,2.4GHz>
        capabilities 0002<ibss>
        beacon-interval 100 TU tsft 18425852105450086784 us
        rates [1.0] 2.0 5.5 11.0
        assoc-id 0 assoc-failed 0 inactivity 0s
        rssi 159 txseq 2 rxseq 551
atw0: mac 00:02:6f:20:f6:2e bss 02:02:6f:20:f6:2e
        node flags 0000
        ess <netbsd>
        chan 11 freq 2462MHz flags 00a0<cck,2.4GHz>
        capabilities 0022<ibss,short preamble>
        beacon-interval 100 TU tsft 18425852102558548069 us
        rates [1.0] 2.0 5.5 6.0 9.0 11.0 12.0 18.0 24.0 36.0 48.0 54.0
        assoc-id 0 assoc-failed 0 inactivity 145s
        rssi 163 txseq 9 rxseq 2563
```

This example is taken from a network consisting of three stations running in ad hoc mode. The key for interpreting the node print-outs follows:

mac	In the example node table, the first network node has MAC number 00:02:6f:20:f6:2e.
bss	The first node belongs to the 802.11 network identified by Basic Service Set Identifier (BSSID) 02:02:6f:20:f6:2e.
node flags	Only three node flags, "bss, sta," and "scan," are presently defined. The first node is distinguished from the rest by its node flags: flag "bss" indicates that the node represents the 802.11 network that the interface has joined or created. The MAC number for the node is the same as the MAC number for the interface.

ess the name of the (Extended) Service Set we have joined. This is the same as the network name set by `ifconfig`(8) with the “ssid” option.

chan **`wlanctl`** prints the channel number, the center frequency in megahertz, and the channel flags. The channel flags indicate the frequency band ((“2.4GHz” or “5GHz)”, modulation (“cck”, “gfsk”, “ofdm”, “turbo”, and “dynamic cck-ofdm)”, and operation constraints (“passive scan”). Common combinations of band and modulation are these:

Band	**Modulation**	**Description**
2.4GHz	cck	11Mb/s DSSS 802.11b
2.4GHz	gfsk	1-2Mb/s FHSS 802.11
2.4GHz	ofdm	54Mb/s 802.11g
2.4GHz	dynamic cck-ofdm	mixed 802.11b/g network
5GHz	ofdm	54Mb/s 802.11a
5GHz	turbo	108Mb/s 802.11a

capabilities ad hoc-mode and AP-mode 802.11 stations advertise their capabilities in 802.11 Beacons and Probe Responses. **`wlanctl`** understands these capability flags:

Flag	**Description**
ess	infrastructure (access point) network
ibss	ad hoc network (no access point)
cf pollable	TBD
request cf poll	TBD
privacy	WEP encryption
short preamble	reduce 802.11b overhead
pbcc	22Mbps ‘‘802.11b+’’
channel agility	change channel for licensed services
short slot-time	TBD
rsn	TBD Real Soon Now
dsss-ofdm	TBD

beacon-interval
In the example, beacons are sent once every 100 Time Units. A Time Unit (TU) is 1024 microseconds (a “kilo-microsecond” or “kus”). Thus 100 TU is about one tenth of a second.

tsft 802.11 stations keep a Time Synchronization Function Timer (TSFT) which counts up in microseconds. Ad hoc-mode stations synchronize time with their peers. Infrastructure-mode stations synchronize time with their access point. Power-saving stations wake and sleep at intervals measured by the TSF Timer. The TSF Timer has a role in the coalescence of 802.11 ad hoc networks (“IBSS merges”).

rates 802.11 stations indicate the bit-rates they support, in units of 100kb/s in 802.11 Beacons, Probe Responses, and Association Requests. **`wlanctl`** prints a station’s supported bit-rates in 1Mb/s units. A station’s basic rates are flagged by an asterisk (‘*’). The last bit-rate at which a packet was sent to the station is enclosed by square brackets.

assoc-id In an infrastructure network, the access point assigns each client an Association Identifier which is used to indicate traffic for power-saving stations.

assoc-failed The number of times the station tried and failed to associate with its access point. Only

inactivity Seconds elapsed since a packet was last received from the station. When this value reaches net.link.ieee80211.maxinact, the station is eligible to be purged from the node table. See `sysctl`(8).

rssi Unitless Received Signal Strength Indication (RSSI). Higher numbers indicate stronger signals. Zero is the lowest possible RSSI. On a hostap- or adhoc-mode interface, the node with *node flag* “bss” set uses *rssi* to indicate the signal strength for the last packet received from a station that does not belong to the network. On an infrastructure-mode station, the node with *node flag* “bss” set indicates the strength of packets from

	the access point.
txseq	The next 802.11 packet sent to this station will carry this transmit sequence number. The 802.11 MAC uses the transmit sequence number to detect duplicate packets.
rxseq	The last packet received from this station carried this transmit sequence number.

SEE ALSO

sysctl(8)

HISTORY

wlanctl first appeared in NetBSD 3.0.

AUTHORS

David Young ⟨dyoung@NetBSD.org⟩

NAME

wpa_background − Background information on Wi-Fi Protected Access and IEEE 802.11i

WPA

The original security mechanism of IEEE 802.11 standard was not designed to be strong and has proven to be insufficient for most networks that require some kind of security. Task group I (Security) of IEEE 802.11 working group (http://www.ieee802.org/11/) has worked to address the flaws of the base standard and has in practice completed its work in May 2004. The IEEE 802.11i amendment to the IEEE 802.11 standard was approved in June 2004 and published in July 2004.

Wi-Fi Alliance (http://www.wi-fi.org/) used a draft version of the IEEE 802.11i work (draft 3.0) to define a subset of the security enhancements that can be implemented with existing wlan hardware. This is called Wi-Fi Protected Access<TM> (WPA). This has now become a mandatory component of interoperability testing and certification done by Wi-Fi Alliance. Wi-Fi provides information about WPA at its web site (http://www.wi-fi.org/OpenSection/protected_access.asp).

IEEE 802.11 standard defined wired equivalent privacy (WEP) algorithm for protecting wireless networks. WEP uses RC4 with 40-bit keys, 24-bit initialization vector (IV), and CRC32 to protect against packet forgery. All these choices have proven to be insufficient: key space is too small against current attacks, RC4 key scheduling is insufficient (beginning of the pseudorandom stream should be skipped), IV space is too small and IV reuse makes attacks easier, there is no replay protection, and non-keyed authentication does not protect against bit flipping packet data.

WPA is an intermediate solution for the security issues. It uses Temporal Key Integrity Protocol (TKIP) to replace WEP. TKIP is a compromise on strong security and possibility to use existing hardware. It still uses RC4 for the encryption like WEP, but with per-packet RC4 keys. In addition, it implements replay protection, keyed packet authentication mechanism (Michael MIC).

Keys can be managed using two different mechanisms. WPA can either use an external authentication server (e.g., RADIUS) and EAP just like IEEE 802.1X is using or pre-shared keys without need for additional servers. Wi-Fi calls these "WPA-Enterprise" and "WPA-Personal", respectively. Both mechanisms will generate a master session key for the Authenticator (AP) and Supplicant (client station).

WPA implements a new key handshake (4-Way Handshake and Group Key Handshake) for generating and exchanging data encryption keys between the Authenticator and Supplicant. This handshake is also used to verify that both Authenticator and Supplicant know the master session key. These handshakes are identical regardless of the selected key management mechanism (only the method for generating master session key changes).

IEEE 802.11I / WPA2

The design for parts of IEEE 802.11i that were not included in WPA has finished (May 2004) and this amendment to IEEE 802.11 was approved in June 2004. Wi-Fi Alliance is using the final IEEE 802.11i as a new version of WPA called WPA2. This includes, e.g., support for more robust encryption algorithm (CCMP: AES in Counter mode with CBC-MAC) to replace TKIP and optimizations for handoff (reduced number of messages in initial key handshake, pre-authentication, and PMKSA caching).

SEE ALSO

wpa_supplicant(8)

LEGAL

NAME

wpa_cli − WPA command line client

SYNOPSIS

wpa_cli [**-p** *path to ctrl sockets*] [**-i** *ifname*] [**-hvB**] [**-a** *action file*] [**-P** *pid file*] [*command ...*]

OVERVIEW

wpa_cli is a text-based frontend program for interacting with wpa_supplicant. It is used to query current status, change configuration, trigger events, and request interactive user input.

wpa_cli can show the current authentication status, selected security mode, dot11 and dot1x MIBs, etc. In addition, it can configure some variables like EAPOL state machine parameters and trigger events like reassociation and IEEE 802.1X logoff/logon. wpa_cli provides a user interface to request authentication information, like username and password, if these are not included in the configuration. This can be used to implement, e.g., one-time-passwords or generic token card authentication where the authentication is based on a challenge-response that uses an external device for generating the response.

The control interface of wpa_supplicant can be configured to allow non-root user access (ctrl_interface GROUP= parameter in the configuration file). This makes it possible to run wpa_cli with a normal user account.

wpa_cli supports two modes: interactive and command line. Both modes share the same command set and the main difference is in interactive mode providing access to unsolicited messages (event messages, username/password requests).

Interactive mode is started when wpa_cli is executed without including the command as a command line parameter. Commands are then entered on the wpa_cli prompt. In command line mode, the same commands are entered as command line arguments for wpa_cli.

INTERACTIVE AUTHENTICATION PARAMETERS REQUEST

When wpa_supplicant need authentication parameters, like username and password, which are not present in the configuration file, it sends a request message to all attached frontend programs, e.g., wpa_cli in interactive mode. wpa_cli shows these requests with "CTRL-REQ-<type>-<id>:<text>" prefix. <type> is IDENTITY, PASSWORD, or OTP (one-time-password). <id> is a unique identifier for the current network. <text> is description of the request. In case of OTP request, it includes the challenge from the authentication server.

The reply to these requests can be given with ’identity’, the matching request. ’password’ and ’otp’ commands can be used regardless of whether the request was for PASSWORD or OTP. The main difference between these two commands is that values given with ’password’ are remembered as long as wpa_supplicant is running whereas values given with ’otp’ are used only once and then forgotten, i.e., wpa_supplicant will ask frontend for a new value for every use. This can be used to implement one-time-password lists and generic token card -based authentication.

Example request for password and a matching reply:

```
CTRL-REQ-PASSWORD-1:Password needed for SSID foobar
> password 1 mysecretpassword
```

Example request for generic token card challenge-response:

```
CTRL-REQ-OTP-2:Challenge 1235663 needed for SSID foobar
> otp 2 9876
```

COMMAND ARGUMENTS

-p path Change the path where control sockets should be found.

-i ifname
Specify the interface that is being configured. By default, choose the first interface found with a control socket in the socket path.

-h Help. Show a usage message.

-v Show version information.

-B Run as a daemon in the background.

-a file Run in daemon mode executing the action file based on events from wpa_supplicant. The specified file will be executed with the first argument set to interface name and second to "CONNECTED" or "DISCONNECTED" depending on the event. This can be used to execute networking tools required to configure the interface.

Additionally, three environmental variables are available to the file: WPA_CTRL_DIR, WPA_ID, and WPA_ID_STR. WPA_CTRL_DIR contains the absolute path to the ctrl_interface socket. WPA_ID contains the unique network_id identifier assigned to the active network, and WPA_ID_STR contains the content of the id_str option.

-P file Set the location of the PID file.

command
Run a command. The available commands are listed in the next section.

COMMANDS

The following commands are available:

status get current WPA/EAPOL/EAP status

mib get MIB variables (dot1x, dot11)

help show this usage help

interface [ifname]
show interfaces/select interface

level <debug level>
change debug level

license show full wpa_cli license

logoff IEEE 802.1X EAPOL state machine logoff

logon IEEE 802.1X EAPOL state machine logon

set set variables (shows list of variables when run without arguments)

pmksa show PMKSA cache

reassociate
force reassociation

reconfigure
force wpa_supplicant to re-read its configuration file

preauthenticate <BSSID>
force preauthentication

identity <network id> <identity>
configure identity for an SSID

password <network id> <password>
configure password for an SSID

pin <network id> <pin>
configure pin for an SSID

otp <network id> <password>
configure one-time-password for an SSID

bssid <network id> <BSSID>
set preferred BSSID for an SSID

list_networks
list configured networks

terminate
terminate **wpa_supplicant**

quit exit wpa_cli

SEE ALSO

wpa_supplicant(8)

LEGAL

NAME

wpa_cli — text-based frontend program for interacting with wpa_supplicant

SYNOPSIS

wpa_cli [*commands*]

DESCRIPTION

The **wpa_cli** utility is a text-based frontend program for interacting with `wpa_supplicant`(8). It is used to query current status, change configuration, trigger events, and request interactive user input.

The **wpa_cli** utility can show the current authentication status, selected security mode, dot11 and dot1x MIBs, etc. In addition, **wpa_cli** can configure EAPOL state machine parameters and trigger events such as reassociation and IEEE 802.1X logoff/logon.

The **wpa_cli** utility provides an interface to supply authentication information such as username and password when it is not provided in the `wpa_supplicant.conf`(5) configuration file. This can be used, for example, to implement one-time passwords or generic token card authentication where the authentication is based on a challenge-response that uses an external device for generating the response.

The **wpa_cli** utility supports two modes: interactive and command line. Both modes share the same command set and the main difference is that in interactive mode, **wpa_cli** provides access to unsolicited messages (event messages, username/password requests).

Interactive mode is started when **wpa_cli** is executed without any parameters on the command line. Commands are then entered from the controlling terminal in response to the **wpa_cli** prompt. In command line mode, the same commands are entered as command line arguments.

The control interface of `wpa_supplicant`(8) can be configured to allow non-root user access by using the *ctrl_interface_group* parameter in the `wpa_supplicant.conf`(5) configuration file. This makes it possible to run **wpa_cli** with a normal user account.

AUTHENTICATION PARAMETERS

When `wpa_supplicant`(8) needs authentication parameters, such as username and password, that are not present in the configuration file, it sends a request message to all attached frontend programs, e.g., **wpa_cli** in interactive mode. The **wpa_cli** utility shows these requests with a "`CTRL-REQ-`⟨*type*⟩`-`⟨*id*⟩:⟨*text*⟩" prefix, where ⟨*type*⟩ is `IDENTITY`, `PASSWORD`, or `OTP` (one-time password), ⟨*id*⟩ is a unique identifier for the current network, and ⟨*text*⟩ is description of the request. In the case of a `OTP` (One Time Password) request, it includes the challenge from the authentication server.

A user must supply `wpa_supplicant`(8) the needed parameters in response to these requests.

For example,

```
CTRL-REQ-PASSWORD-1:Password needed for SSID foobar
> password 1 mysecretpassword

Example request for generic token card challenge-response:

CTRL-REQ-OTP-2:Challenge 1235663 needed for SSID foobar
> otp 2 9876
```

COMMANDS

The following commands may be supplied on the command line or at a prompt when operating interactively.

status Report the current WPA/EAPOL/EAP status for the current interface.

mib Report MIB variables (dot1x, dot11) for the current interface.

help Show usage help.

interface [*ifname*]
Show available interfaces and/or set the current interface when multiple are available.

level *debug_level*
Change the debugging level in wpa_supplicant(8). Larger numbers generate more messages.

license
Display the full license for **wpa_cli**.

logoff Send the IEEE 802.1X EAPOL state machine into the "logoff" state.

logon Send the IEEE 802.1X EAPOL state machine into the "logon" state.

set [*settings*]
Set variables. When no arguments are supplied, the known variables and their settings are displayed.

pmksa Show the contents of the PMKSA cache.

reassociate
Force a reassociation to the current access point.

reconfigure
Force wpa_supplicant(8) to re-read its configuration file.

preauthenticate *BSSID*
Force preauthentication of the specified *BSSID*.

identity *network_id identity*
Configure an identity for an SSID.

password *network_id password*
Configure a password for an SSID.

otp *network_id password*
Configure a one-time password for an SSID.

terminate
Force wpa_supplicant(8) to terminate.

quit Exit **wpa_cli**.

SEE ALSO
wpa_supplicant.conf(5), wpa_supplicant(8)

HISTORY
The **wpa_cli** utility first appeared in NetBSD 4.0.

AUTHORS
The **wpa_cli** utility was written by Jouni Malinen ⟨jkmaline@cc.hut.fi⟩. This manual page is derived from the README file included in the **wpa_supplicant** distribution.

NAME

wpa_gui − WPA Graphical User Interface

SYNOPSIS

wpa_gui [**-p** *path to ctrl sockets*] [**-i** *ifname*]

OVERVIEW

wpa_gui is a QT graphical frontend program for interacting with wpa_supplicant. It is used to query current status, change configuration and request interactive user input.

wpa_gui supports (almost) all of the interactive status and configuration features of the command line client, wpa_cli. Refer to the wpa_cli manpage for a comprehensive list of the interactive mode features.

COMMAND ARGUMENTS

-p path Change the path where control sockets should be found.

-i ifname
Specify the interface that is being configured. By default, choose the first interface found with a control socket in the socket path.

SEE ALSO

wpa_cli(8) **wpa_supplicant**(8)

LEGAL

NAME

wpa_passphrase − Generate a WPA PSK from an ASCII passphrase for a SSID

SYNOPSIS

wpa_passphrase [*ssid*] [*passphrase*]

OVERVIEW

wpa_passphrase pre-computes PSK entries for network configuration blocks of a *wpa_supplicant.conf* file. An ASCII passphrase and SSID are used to generate a 256-bit PSK.

OPTIONS

ssid The SSID whose passphrase should be derived.

passphrase
The passphrase to use. If not included on the command line, passphrase will be read from standard input.

SEE ALSO

wpa_supplicant.conf(5) **wpa_supplicant**(8)

LEGAL

NAME

wpa_passphrase — Set WPA passphrase for a SSID

SYNOPSIS

wpa_passphrase *ssid passphrase*

DESCRIPTION

The **wpa_passphrase** utility pre-computes PSK entries for network configuration blocks of a `wpa_supplicant.conf`(5) file.

The following arguments must be specified on the command line:

ssid	The SSID whose passphrase should be derived.
passphrase	The passphrase to use. If not included on the command line, passphrase will be read from standard input.

SEE ALSO

`wpa_supplicant.conf`(5), `ifconfig`(8), `wpa_cli`(8), `wpa_supplicant`(8)

HISTORY

The **wpa_passphrase** utility first appeared in NetBSD 4.0.

AUTHORS

The **wpa_passphrase** utility was written by Jouni Malinen ⟨jkmaline@cc.hut.fi⟩. This manual page is derived from the `wpa_passphrase.sgml` file included in the `wpa_supplicant`(8) distribution.

NAME

wpa_priv − wpa_supplicant privilege separation helper

SYNOPSIS

wpa_priv [**-c** *ctrl path*] [**-Bdd**] [**-P** *pid file*] [**driver:ifname** *[driver:ifname ...]*]

OVERVIEW

wpa_priv is a privilege separation helper that minimizes the size of **wpa_supplicant** code that needs to be run with root privileges.

If enabled, privileged operations are done in the wpa_priv process while leaving rest of the code (e.g., EAP authentication and WPA handshakes) to operate in an unprivileged process (wpa_supplicant) that can be run as non-root user. Privilege separation restricts the effects of potential software errors by containing the majority of the code in an unprivileged process to avoid the possibility of a full system compromise.

wpa_priv needs to be run with network admin privileges (usually, root user). It opens a UNIX domain socket for each interface that is included on the command line; any other interface will be off limits for **wpa_supplicant** in this kind of configuration. After this, **wpa_supplicant** can be run as a non-root user (e.g., all standard users on a laptop or as a special non-privileged user account created just for this purpose to limit access to user files even further).

EXAMPLE CONFIGURATION

The following steps are an example of how to configure **wpa_priv** to allow users in the ’wpapriv’ group to communicate with **wpa_supplicant** with privilege separation:

Create user group (e.g., wpapriv) and assign users that should be able to use wpa_supplicant into that group.

Create /var/run/wpa_priv directory for UNIX domain sockets and control user access by setting it accessible only for the wpapriv group:

```
mkdir /var/run/wpa_priv
chown root:wpapriv /var/run/wpa_priv
chmod 0750 /var/run/wpa_priv
```

Start **wpa_priv** as root (e.g., from system startup scripts) with the enabled interfaces configured on the command line:

```
wpa_priv -B -c /var/run/wpa_priv -P /var/run/wpa_priv.pid wext:wlan0
```

Run **wpa_supplicant** as non-root with a user that is in the wpapriv group:

```
wpa_supplicant -i ath0 -c wpa_supplicant.conf
```

COMMAND ARGUMENTS

-c ctrl path
Specify the path to wpa_priv control directory (Default: /var/run/wpa_priv/).

-B Run as a daemon in the background.

-P file Set the location of the PID file.

driver:ifname [driver:ifname ...]
The <driver> string dictates which of the supported **wpa_supplicant** driver backends is to be used. To get a list of supported driver types see wpa_supplicant help (e.g, wpa_supplicant -h). The driver backend supported by most good drivers is ’wext’.

The <ifname> string specifies which network interface is to be managed by **wpa_supplicant** (e.g., wlan0 or ath0).

wpa_priv does not use the network interface before **wpa_supplicant** is started, so it is fine to include network interfaces that are not available at the time wpa_priv is started. wpa_priv can control multiple interfaces with one process, but it is also possible to run multiple **wpa_priv** processes at the same time, if desired.

SEE ALSO

wpa_supplicant(8)

LEGAL

NAME

wpa_supplicant − Wi-Fi Protected Access client and IEEE 802.1X supplicant

SYNOPSIS

wpa_supplicant [**-BddfhKLqqtuvwW**] [**-i***ifname*] [**-c***config file*] [**-D***driver*] [**-P***PID_file*] [**-f***output file*]

OVERVIEW

Wireless networks do not require physical access to the network equipment in the same way as wired networks. This makes it easier for unauthorized users to passively monitor a network and capture all transmitted frames. In addition, unauthorized use of the network is much easier. In many cases, this can happen even without user's explicit knowledge since the wireless LAN adapter may have been configured to automatically join any available network.

Link-layer encryption can be used to provide a layer of security for wireless networks. The original wireless LAN standard, IEEE 802.11, included a simple encryption mechanism, WEP. However, that proved to be flawed in many areas and network protected with WEP cannot be consider secure. IEEE 802.1X authentication and frequently changed dynamic WEP keys can be used to improve the network security, but even that has inherited security issues due to the use of WEP for encryption. Wi-Fi Protected Access and IEEE 802.11i amendment to the wireless LAN standard introduce a much improvement mechanism for securing wireless networks. IEEE 802.11i enabled networks that are using CCMP (encryption mechanism based on strong cryptographic algorithm AES) can finally be called secure used for applications which require efficient protection against unauthorized access.

wpa_supplicant is an implementation of the WPA Supplicant component, i.e., the part that runs in the client stations. It implements WPA key negotiation with a WPA Authenticator and EAP authentication with Authentication Server. In addition, it controls the roaming and IEEE 802.11 authentication/association of the wireless LAN driver.

wpa_supplicant is designed to be a "daemon" program that runs in the background and acts as the backend component controlling the wireless connection. **wpa_supplicant** supports separate frontend programs and an example text-based frontend, **wpa_cli**, is included with wpa_supplicant.

Before wpa_supplicant can do its work, the network interface must be available. That means that the physical device must be present and enabled, and the driver for the device must have be loaded. Note, however, that the '-w' option of the wpa_supplicant daemon instructs the daemon to continue running and to wait for the interface to become available. Without the '-w' option, the daemon will exit immediately if the device is not already available.

After **wpa_supplicant** has configured the network device, higher level configuration such as DHCP may proceed. There are a variety of ways to integrate wpa_supplicant into a machine's networking scripts, a few of which are described in sections below.

The following steps are used when associating with an AP using WPA:

- **wpa_supplicant** requests the kernel driver to scan neighboring BSSes
- **wpa_supplicant** selects a BSS based on its configuration
- **wpa_supplicant** requests the kernel driver to associate with the chosen BSS
- If WPA-EAP: integrated IEEE 802.1X Supplicant completes EAP authentication with the authentication server (proxied by the Authenticator in the AP)
- If WPA-EAP: master key is received from the IEEE 802.1X Supplicant
- If WPA-PSK: **wpa_supplicant** uses PSK as the master session key
- **wpa_supplicant** completes WPA 4-Way Handshake and Group Key Handshake with the Authenticator (AP)
- **wpa_supplicant** configures encryption keys for unicast and broadcast

- normal data packets can be transmitted and received

SUPPORTED FEATURES

Supported WPA/IEEE 802.11i features:

- WPA-PSK ("WPA-Personal")
- WPA with EAP (e.g., with RADIUS authentication server) ("WPA-Enterprise") Following authentication methods are supported with an integrate IEEE 802.1X Supplicant:
  - EAP-TLS
  - EAP-PEAP/MSCHAPv2 (both PEAPv0 and PEAPv1)
  - EAP-PEAP/TLS (both PEAPv0 and PEAPv1)
  - EAP-PEAP/GTC (both PEAPv0 and PEAPv1)
  - EAP-PEAP/OTP (both PEAPv0 and PEAPv1)
  - EAP-PEAP/MD5-Challenge (both PEAPv0 and PEAPv1)
  - EAP-TTLS/EAP-MD5-Challenge
  - EAP-TTLS/EAP-GTC
  - EAP-TTLS/EAP-OTP
  - EAP-TTLS/EAP-MSCHAPv2
  - EAP-TTLS/EAP-TLS
  - EAP-TTLS/MSCHAPv2
  - EAP-TTLS/MSCHAP
  - EAP-TTLS/PAP
  - EAP-TTLS/CHAP
  - EAP-SIM
  - EAP-AKA
  - EAP-PSK
  - EAP-PAX
  - LEAP (note: requires special support from the driver for IEEE 802.11 authentication)
  - (following methods are supported, but since they do not generate keying material, they cannot be used with WPA or IEEE 802.1X WEP keying)
  - EAP-MD5-Challenge
  - EAP-MSCHAPv2
  - EAP-GTC
  - EAP-OTP
- key management for CCMP, TKIP, WEP104, WEP40
- RSN/WPA2 (IEEE 802.11i)
  - pre-authentication
  - PMKSA caching

AVAILABLE DRIVERS

The available drivers to specify with the -D option are:

hostap (default) Host AP driver (Intersil Prism2/2.5/3). (this can also be used with Linuxant DriverLoader).

hermes
Agere Systems Inc. driver (Hermes-I/Hermes-II).

madwifi
MADWIFI 802.11 support (Atheros, etc.).

atmel ATMEL AT76C5XXx (USB, PCMCIA).

wext Linux wireless extensions (generic).

ndiswrapper
Linux ndiswrapper.

broadcom
Broadcom wl.o driver.

ipw Intel ipw2100/2200 driver.

wired wpa_supplicant wired Ethernet driver

bsd BSD 802.11 support (Atheros, etc.).

ndis Windows NDIS driver.

COMMAND LINE OPTIONS

-b br_ifname
Optional bridge interface name.

-B Run daemon in the background.

-c filename
Path to configuration file.

-C ctrl_interface
Path to ctrl_interface socket (only used if -c is not).

-i ifname
Interface to listen on.

-d Increase debugging verbosity (-dd even more).

-D driver
Driver to use. See the available options below.

-f output file
Log output to specified file instead of stdout.

-g global ctrl_interface
Path to global ctrl_interface socket.

-K Include keys (passwords, etc.) in debug output.

-t Include timestamp in debug messages.

-h Help. Show a usage message.

-L Show license (GPL and BSD).

-p Driver parameters.

-P PID_file
Path to PID file.

-q Decrease debugging verbosity (-qq even less).

-u Enabled DBus control interface.

-v Show version.

-w Wait for interface to be added, if needed. Normally, **wpa_supplicant** will exit if the interface is not there yet.

-W Wait for a control interface monitor before starting.

-N Start describing new interface.

EXAMPLES

In most common cases, **wpa_supplicant** is started with:

```
wpa_supplicant -Bw -c/etc/wpa_supplicant.conf -iwlan0
```

This makes the process fork into background and wait for the wlan0 interface if it is not available at startup time.

The easiest way to debug problems, and to get debug log for bug reports, is to start **wpa_supplicant** on foreground with debugging enabled:

```
wpa_supplicant -c/etc/wpa_supplicant.conf -iwlan0 -d
```

wpa_supplicant can control multiple interfaces (radios) either by running one process for each interface separately or by running just one process and list of options at command line. Each interface is separated with -N argument. As an example, following command would start wpa_supplicant for two interfaces:

```
wpa_supplicant \
        -c wpa1.conf -i wlan0 -D hostap -N \
        -c wpa2.conf -i ath0 -D madwifi
```

OS REQUIREMENTS

Current hardware/software requirements:

- Linux kernel 2.4.x or 2.6.x with Linux Wireless Extensions v15 or newer
- FreeBSD 6-CURRENT
- Microsoft Windows with WinPcap (at least WinXP, may work with other versions)

SUPPORTED DRIVERS

Host AP driver for Prism2/2.5/3 (development snapshot/v0.2.x)
(http://hostap.epitest.fi/) Driver needs to be set in Managed mode ('iwconfig wlan0 mode managed'). Please note that station firmware version needs to be 1.7.0 or newer to work in WPA mode.

Linuxant DriverLoader
(http://www.linuxant.com/driverloader/) with Windows NDIS driver for your wlan card supporting WPA.

Agere Systems Inc. Linux Driver
(http://www.agere.com/support/drivers/) Please note that the driver interface file (driver_hermes.c) and hardware specific include files are not included in the wpa_supplicant distribution. You will need to copy these from the source package of the Agere driver.

madwifi driver for cards based on Atheros chip set (ar521x)
(http://sourceforge.net/projects/madwifi/) Please note that you will need to modify the wpa_supplicant .config file to use the correct path for the madwifi driver root directory (CFLAGS += -I../madwifi/wpa line in example defconfig).

ATMEL AT76C5XXx driver for USB and PCMCIA cards
(http://atmelwlandriver.sourceforge.net/).

Linux ndiswrapper
(http://ndiswrapper.sourceforge.net/) with Windows NDIS driver.

Broadcom wl.o driver
This is a generic Linux driver for Broadcom IEEE 802.11a/g cards. However, it is proprietary driver that is not publicly available except for couple of exceptions, mainly Broadcom-based APs/wireless routers that use Linux. The driver binary can be downloaded, e.g., from Linksys support site (http://www.linksys.com/support/gpl.asp) for Linksys WRT54G. The GPL tarball includes cross-compiler and the needed header file, wlioctl.h, for compiling wpa_supplicant. This driver support in wpa_supplicant is expected to work also with other devices based on Broadcom driver (assuming the driver includes client mode support).

Intel ipw2100 driver
(http://sourceforge.net/projects/ipw2100/)

Intel ipw2200 driver
(http://sourceforge.net/projects/ipw2200/)

Linux wireless extensions
In theory, any driver that supports Linux wireless extensions can be used with IEEE 802.1X (i.e., not WPA) when using ap_scan=0 option in configuration file.

Wired Ethernet drivers
Use ap_scan=0.

BSD net80211 layer (e.g., Atheros driver)
At the moment, this is for FreeBSD 6-CURRENT branch.

Windows NDIS
The current Windows port requires WinPcap (http://winpcap.polito.it/). See README-Windows.txt for more information.

wpa_supplicant was designed to be portable for different drivers and operating systems. Hopefully, support for more wlan cards and OSes will be added in the future. See developer.txt for more information about the design of wpa_supplicant and porting to other drivers. One main goal is to add full WPA/WPA2 support to Linux wireless extensions to allow new drivers to be supported without having to implement new driver-specific interface code in wpa_supplicant.

ARCHITECTURE

The **wpa_supplicant** system consists of the following components:

wpa_supplicant.conf
the configuration file describing all networks that the user wants the computer to connect to.

wpa_supplicant
the program that directly interacts with the network interface.

wpa_cli
the client program that provides a high-level interface to the functionality of the daemon.

wpa_passphrase
a utility needed to construct *wpa_supplicant.conf* files that include encrypted passwords.

QUICK START

First, make a configuration file, e.g. */etc/wpa_supplicant.conf*, that describes the networks you are interested in. See **wpa_supplicant.conf**(5) for details.

Once the configuration is ready, you can test whether the configuration works by running **wpa_supplicant** with following command to start it on foreground with debugging enabled:

```
wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf -d
```

Assuming everything goes fine, you can start using following command to start **wpa_supplicant** on background without debugging:

```
wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf -B
```

Please note that if you included more than one driver interface in the build time configuration (.config), you may need to specify which interface to use by including -D<driver name> option on the command line.

INTERFACE TO PCMCIA-CS/CARDMRG

For example, following small changes to pcmcia-cs scripts can be used to enable WPA support:

Add MODE="Managed" and WPA="y" to the network scheme in */etc/pcmcia/wireless.opts*.

Add the following block to the end of 'start' action handler in */etc/pcmcia/wireless*:

```
if [ "$WPA" = "y" -a -x /usr/local/bin/wpa_supplicant ]; then
   /usr/local/bin/wpa_supplicant -Bw -c/etc/wpa_supplicant.conf -i$DEVICE
fi
```

Add the following block to the end of 'stop' action handler (may need to be separated from other actions) in */etc/pcmcia/wireless*:

```
if [ "$WPA" = "y" -a -x /usr/local/bin/wpa_supplicant ]; then
   killall wpa_supplicant
fi
```

This will make **cardmgr** start **wpa_supplicant** when the card is plugged in. **wpa_supplicant** will wait until the interface is set up--either when a static IP address is configured or when DHCP client is started--and will then negotiate keys with the AP.

SEE ALSO

wpa_background(8) **wpa_supplicant.conf**(5) **wpa_cli**(8) **wpa_passphrase**(8)

LEGAL

wpa_supplicant is copyright (c) 2003-2007, Jouni Malinen <j@w1.fi> and contributors. All Rights Reserved.

This program is dual-licensed under both the GPL version 2 and BSD license. Either license may be used at your option.

NAME

wpa_supplicant — WPA/802.11i Supplicant for wireless network devices

SYNOPSIS

wpa_supplicant [**−BdehLqvw**] **−i** *ifname* **−c** *config-file*
[**−N −i** *ifname* **−c** *config-file ...*]

DESCRIPTION

The **wpa_supplicant** utility is an implementation of the WPA Supplicant component, i.e., the part that runs in the client stations. It implements WPA key negotiation with a WPA Authenticator and EAP authentication with an Authentication Server. In addition, **wpa_supplicant** controls the roaming and IEEE 802.11 authentication/association support and can be used to configure static WEP keys based on identified networks.

The **wpa_supplicant** utility is designed to be a "daemon" program that runs in the background and acts as the backend component controlling the wireless connection. It supports separate frontend programs such as the text-based `wpa_cli`(8) program.

The following arguments must be specified on the command line:

−i *ifname*
Use the specified wireless interface.

−c *config-file*
Use the settings in the specified configuration file when managing the wireless interface. See `wpa_supplicant.conf`(5) for a description of the configuration file syntax and contents.

Changes to the configuration file can be reloaded by sending a `SIGHUP` signal to the **wpa_supplicant** process or with the `wpa_cli`(8) utility, using "`wpa_cli reconfigure`".

OPTIONS

The following options are available:

−B Detach from the controlling terminal and run as a daemon process in the background.

−d Enable debugging messages. If this option is supplied twice, more verbose messages are displayed.

−e Use an external IEEE 802.1X Supplicant program and disable the internal Supplicant. This option is not normally used.

−h Show help text.

−K Include key information in debugging output.

−L Display the license for this program on the terminal and exit.

−N −i *ifname* **−c** *config-file ...*
Specify an additional interface and configuration file. If multiple interfaces are specified then **wpa_supplicant** will manage them all with a single process.

−q Decrease debugging verbosity (i.e., counteract the use of the **−d** flag).

−v Display version information on the terminal and exit.

−w If the specified interface is not present, wait for it to be added; e.g. a cardbus device to be inserted.

SEE ALSO

`ath`(4), `ipw`(4), `iwi`(4), `ral`(4), `wi`(4), `wpa_supplicant.conf`(5), `ifconfig`(8), `wpa_cli`(8)

HISTORY

The **`wpa_supplicant`** utility first appeared in NetBSD 4.0.

AUTHORS

The **`wpa_supplicant`** utility was written by Jouni Malinen ⟨jkmaline@cc.hut.fi⟩. This manual page is derived from the `README` file included in the **`wpa_supplicant`** distribution.

NAME

wsconscfg — configure and switch between virtual terminals on a wscons display

SYNOPSIS

wsconscfg [**−e** *emul*] [**−f** *ctldev*] [**−t** *type*] *index*
wsconscfg −d [**−F**] [**−f** *ctldev*] *index*
wsconscfg −g [**−f** *ctldev*]
wsconscfg −k | **−m** [**−d**] [**−f** *ctldev*] [*index*]
wsconscfg −s [**−f** *ctldev*] *index*

DESCRIPTION

The **wsconscfg** tool allows to create, delete and switch between virtual terminals on display devices controlled by the wscons terminal framework if the underlying display hardware driver supports multiple screens. Further it controls the assignment of keyboards to displays. The *index* argument specifies which virtual terminal is to be configured; the allowed numbers are from 0 to an implementation-specified value (currently 7, allowing for 8 virtual terminals on a display). In keyboard configuration mode, it specifies the wskbd(4) device to attach or detach. Without further option arguments, a virtual terminal is created with implementation specific properties and a default terminal emulation variant selected at kernel compile time.

The options are:

−d Delete the specified terminal. A terminal opened by a program will not be deleted unless the **−F** option is applied. Terminals used by the operating system console or a graphics program (X server) cannot be deleted. With the **−k** flag, the keyboard specified by *index* will be detached from the wscons display. With the **−m** flag, the multiplexor specified by *index* will be detached from the wscons display.

−e *emul* Specify the terminal emulation to use for the virtual terminal. The set of available terminal emulations is determined at kernel compile time. See wscons(4) for details.

−F Force deleting of a terminal even if it is in use by a user space program.

−f *ctldev*
Specify the control device of the wscons display to operate on. Default is `/dev/ttyEcfg`.

−g Print the index of the current virtual terminal.

−k Do keyboard related operations instead of virtual screen configuration. Without other flags, a keyboard will be attached to the display device. The *index* argument can be omitted, in this case the first free keyboard will be used.

−m Do multiplexor related operations instead of virtual screen configuration. Without other flags, a multiplexor will be attached to the display device.

−s Switch to the specified virtual terminal.

−t *type* Specify a screen type to use. Screen types refer to display format, colour depth and other low-level display properties. Valid *type* arguments are defined by the underlying display device driver.

Typically, the **wsconscfg** utility will be invoked in system startup by the `/etc/rc.d/wscons` script, controlled by the `/etc/wscons.conf` configuration file.

FILES

`/etc/wscons.conf`

EXAMPLES

```
wsconscfg -t 80x50 -e vt100 1
```

Configure screen 1 (i.e., the second), it will get the type `80x50` and use the VT100 terminal emulation. (Note: `80x50` is a screen type offered by the `vga`(4) display driver. In this particular case, an 8×8-font must be loaded before to make the screen useful. See `wsfontload`(8).)

```
wsconscfg -k
```

Connect the first unconnected keyboard to the display.

```
wsconscfg 3
```

Create screen 3.

```
wsconscfg -d 3
```

Delete screen 3.

```
wsconscfg -s 2
```

Switch to screen 2.

SEE ALSO

`wscons`(4), `wskbd`(4), `wsconsctl`(8), `wsfontload`(8)

BUGS

There should be an easy way to get a list of the screen types available on a display, and of the emulations supported by the kernel.

NAME

wsconsctl — get or set wscons state

SYNOPSIS

wsconsctl [**-dkmn**] [**-f** *file*] **-a**
wsconsctl [**-dkmn**] [**-f** *file*] *name ...*
wsconsctl [**-dkmn**] [**-f** *file*] **-w** *name=value ...*
wsconsctl [**-dkmn**] [**-f** *file*] **-w** *name+=value ...*

DESCRIPTION

The **wsconsctl** command displays or sets various wscons system driver variables. If a list of variables is present on the command line, then **wsconsctl** prints the current value of those variables for the specified device.

-a Specify all variables for the device.

-d Select the display portion of the device.

-f *file*
Specify an alternative control device.

-k Select the keyboard portion of the device (this is the default).

-m Select the mouse portion of the device.

-n Suppress the printing of the variable name in the output - only the value will appear.

-w Set or modify the specified variables to the given values. The value can be specified as either an absolute value, by using the '=' symbol or as a relative value, by using the '+=' symbol. See the **EXAMPLES** section for more details.

The **wsconsctl** utility can be used to view and modify aspects of the keyboard, display, and mouse, using the standard, machine-independent workstation console device driver wscons(4).

The keyboard type can be modified, the keyboard bell's pitch, period, and duration can be modified, the *typematic* value can be changed, and the keyboard encoding can be modified to switch keys, should the user find a keyboard's default layout difficult to use. The keyboard types and other relevant definitions can all be found in the /usr/include/dev/wscons/wsksymdef.h file.

The mouse types are defined in the /usr/include/dev/wscons/wsconsio.h file.

The display types, height, width, depth (bits per pixel), color map size, and color map are defined in the /usr/include/dev/wscons/wsconsio.h file. There are also definitions relating to video control and cursor control, which are not applicable to all display types, and to text emulation and graphics (mapped) modes.

In addition to British, US, and US-Dvorak keyboard encodings, support currently exists for the following languages: Danish, Finnish, French, German, Italian, Japanese, Norwegian, Portuguese, Spanish, and Swedish. Additionally, a user-defined encoding is supported.

FILES

/dev/wskbd	keyboard control device
/dev/wsmouse	mouse control device
/dev/ttyE0	display control device

EXAMPLES

The following are just a few examples of **wsconsctl** and its functionality.

```
wsconsctl -w encoding=uk
```

Set a UK keyboard encoding.

```
wsconsctl -w map+="keysym Caps_Lock = Control_L"
```

Modify the current keyboard encoding so that when the *Caps Lock* key is pressed, the same encoding sequence as *Left Control* is sent. For a full list of keysyms and keycodes, please refer to the `/usr/include/dev/wscons/wsksymdef.h` file.

```
wsconsctl -w encoding=us.swapctrlcaps
```

Set a US keyboard encoding, with the *Caps Lock* and *Left Control* keys swapped. The *.swapctrlcaps* encoding does not work for all national keyboard encodings. For most purposes, the ability to set the value returned by the *Caps Lock* key is enough - see the previous example for details.

```
wsconsctl -w bell.pitch=1200
```

Set the bell pitch to be 1200.

```
wsconsctl -w bell.pitch+=200
```

Add 200 to the current pitch of the bell.

```
wsconsctl -d -w msg.kernel.attrs=color,hilit msg.kernel.bg=red
msg.kernel.fg=brown
```

Set the color of kernel messages to brown on red with the highlighting flag set (becoming yellow on red).

```
wsconsctl -w repeat.del1=200 repeat.deln=50
```

Set the initial delay for keyboard auto repeat to 200ms, and subsequent delays to 50ms.

```
wsconsctl -w repeat.del1=0
```

Turn off auto repeat.

```
wsconsctl -d -w scroll.fastlines=50
```

If scroll support is enabled in the kernel, set the number of lines used in the fast scroll function to 50.

```
wsconsctl -d -w scroll.slowlines=2
```

If scroll support is enabled in the kernel, set the number of lines used in the slow scroll function to 2. In order to use this function, you have to have `Cmd_ScrollSlowDown` and `Cmd_ScrollSlowUp` defined in your keyboard map.

SEE ALSO

`pckbd`(4), `wscons`(4), `wsconscfg`(8), `wsfontload`(8)

HISTORY

The **wsconsctl** command first appeared in NetBSD 1.4.

NAME

wsfontload — load a font bitmap into the wsfont pool or a wscons display device

SYNOPSIS

wsfontload [**-f** *wsdev*] [**-w** *width*] [**-h** *height*] [**-e** *encoding*] [**-N** *name*] [**-b**] [**-B**] [**-v**] [*fontfile*]

DESCRIPTION

The **wsfontload** utility loads a font bitmap into the wsfont font pool (or a wscons device if the device driver supports this). The font gets assigned a name in this process which it can be referred to by later for use on a display screen. The font is loaded from the specified *fontfile*, or from standard input if *fontfile* is not provided.

The options are:

-f *wsdev* Specify the device to operate on. Default is `/dev/wsfont`.

-w *width* Sets the width of a font character in pixels. Default is 8.

-h *height* Sets the height of a font character in pixels. Default is 16.

-e *encoding*
Sets the encoding of the font. This can be either a symbolic abbreviation or a numeric value. Currently recognized abbreviations are `iso` for ISO-8859-1 encoding, `ibm` for IBM encoded fonts and `pcvt` for the custom encoding of the supplemental fonts which came with the BSD "pcvt" console driver. Per default, `iso` is assumed.

-N *name* Specifies a name which can be used later to refer to the font. If none is given, the *fontfile* name is used to create one.

-b Specifies that the font data is ordered right-to-left bit wise. The default is left-to-right.

-B Specifies that the font data is ordered right-to-left byte wise. The default is left-to-right.

-v Prints the font's properties before loading it.

Typically, the **wsfontload** utility will be executed in system startup by the `/etc/rc.d/wscons` script, controlled by the `/etc/wscons.conf` configuration file.

FILES

`/etc/wscons.conf` `/usr/share/wscons/fonts`

EXAMPLES

```
wsfontload -N myname -h 8 -e ibm /usr/share/wscons/fonts/vt220l.808
```

Load the IBM-encoded 8×8-font from the `wscons`(4) distribution. This (or another 8×8-font) is necessary to use the 50-line screen type on `vga`(4) displays.

```
wsfontload -N orator -e ibm /usr/share/wscons/fonts/orator.816
wsconsctl -dw font=orator
```

Load the "orator" IBM-encoded 8×16 font and switch the first console screen (ttyE0, wsconsctl's default) to this alternate font.

SEE ALSO

`wscons`(4), `wsconscfg`(8), `wsconsctl`(8)

BUGS

Many features are missing.

There is no way to remove a loaded font.

NAME

wsmoused — multipurpose mouse daemon

SYNOPSIS

wsmoused [**-d** *device*] [**-f** *conf_file*] [**-m** *modes*] [**-n**]

DESCRIPTION

The **wsmoused** daemon provides mouse support in console, allowing copying and pasting text. The left mouse button is used to select text when held and you use the right button to paste it in the active console.

Supported options are as follows:

-d *device*	specifies the device file to be used as the wsmouse(4) device. Defaults to /dev/wsmouse.
-f *conf_file*	specifies the configuration file to be used. Defaults to /etc/wsmoused.conf.
-m *modes*	specifies which modes should be activated. Mode names are given in the argument as a whitespace separated list. Overrides the 'modes' directive in the configuration file.
-n	do not fork in the background (for debugging purposes). Overrides the 'nodaemon' directive in the configuration file.

Many other details can be tuned. See wsmoused.conf(5) for more information.

wsmoused is designed to be a multipurpose mouse daemon. Functionality is provided through independent *modes*, enabled either through the **-m** flag or through the 'modes' property in the configuration file (the former takes precedence).

The action mode

The 'action' mode executes commands upon receiving mouse button events. Commands can be associated on a button basis, and can differentiate between push or release events.

The selection mode

The 'selection' mode provides visual copy and paste support in text consoles when using the wscons(4) device. A selection is created by clicking with the primary mouse button at any point on the screen and dragging it while clicked. When the button is released, the selected text is copied to an internal buffer for further pasting with the secondary button.

FILES

/dev/ttyE[0-n]	tty devices
/dev/ttyEstat	wsdisplay status notification device
/dev/wsmouse[0-n]	mouse control device
/etc/wsmoused.conf	default configuration file

SECURITY CONSIDERATIONS

When using the 'action' mode, commands specified in the configuration file are executed as the user who started the daemon. By default, this user is 'root' when using the rc.subr(8) framework. You should set 'wsmoused_user="<some_user>"' in rc.conf(5) to a safer user (and adjust file permissions accordingly) if the commands you want to execute do not require superuser privileges. An alternative is to use su(1) as part of the command string in the configuration file.

NOTES

The following notes apply to all work modes:

- When switching from the X screen to a text terminal, there is a small delay (five seconds) until the mouse works again. This time is used by X to close the mouse device properly.

The following notes apply to the 'selection' mode only:

- The mouse cursor is only visible for a short period of time. It will disappear when you stop moving it to avoid console corruption (which happens if it is visible and there is text output).
- You need to change the getty program which is run in the first virtual terminal to use `/dev/ttyE0` instead of `/dev/console`. To do this, edit `/etc/ttys` and `/etc/wscons.conf`.

SEE ALSO

`su`(1), `wscons`(4), `wsdisplay`(4), `wsmouse`(4), `rc.conf`(5), `ttys`(5), `wscons.conf`(5), `wsmoused.conf`(5), `moused`(8), `rc.subr`(8)

HISTORY

The **wsmoused** command first appeared in NetBSD 2.0.

AUTHORS

The **wsmoused** command was developed by Julio M. Merino Vidal ⟨jmmv@NetBSD.org⟩.

NAME

wsmuxctl — configure wsmuxes

SYNOPSIS

wsmuxctl [**-a** *dev*] **-f** *ctldev* [**-l**] [**-L**] [**-r** *dev*]

DESCRIPTION

The **wsmuxctl** allows to adding and removing devices connected to a wsmux(4).

Added and removed devices are specified by a name, not a path name. Simply use *wsmouseN*, *wskbdN*, or *wsmuxN* where N is the number of the device.

The options are:

-a *dev* Add the specified device to the mux.

-f *ctldev*
Specify the control device of the wsmux to operate on. A number may be given, which is then taken to be the number of the mux.

-l List all devices connected to a mux.

-L List all devices connected to a mux and recursively list mux subdevices.

-r *dev* Remove the specified device from the mux.

SEE ALSO

wsmux(4)

NAME

ypbind — create and maintain a binding to a NIS server

SYNOPSIS

ypbind [**−broadcast**] [**−insecure**] [**−ypset**] [**−ypsetme**]

DESCRIPTION

ypbind finds the server for a particular NIS domain and stores information about it in a "binding file". This binding information includes the IP address of the server associated with that particular domain and which port the server is using. This information is stored in the directory `/var/yp/binding` in a file named with the convention `<domain>.version`, where ⟨domain⟩ is the relevant domain. The NIS system only supplies information on version 2.

If **ypbind** is started without the **−broadcast** option, **ypbind** steps through the list of NIS servers specified in `/var/yp/binding/<domain>.ypservers` and contacts each in turn attempting to bind to that server. It is strongly recommended that these hosts are in the local hosts file, and that hosts are looked up in local files before the NIS hosts map.

If **ypbind** is started with the **−broadcast** option, or if `/var/yp/binding/<domain>.ypservers` does not exist, **ypbind** broadcasts to find a process willing to serve maps for the client's domain.

Once a binding is established, **ypbind** maintains this binding by periodically communicating with the server to which it is bound. If the binding is somehow lost, e.g by server reboot, **ypbind** marks the domain as unbound and attempts to re-establish the binding. When the binding is once again successful, **ypbind** marks the domain as bound and resumes its periodic check.

The options are as follows:

−broadcast
sends a broadcast requesting a NIS server to which to bind.

−insecure do not require that the server is running on a reserved port. This may be necessary when connecting to SunOS 3.x or ULTRIX NIS servers.

−ypset `ypset`(8) may be used to change the server to which a domain is bound.

−ypsetme `ypset`(8) may be used only from this machine to change the server to which a domain is bound.

The **−broadcast −ypset**, and **−ypsetme**, options are inherently insecure and should be avoided.

FILES

`/var/yp/binding/<domain>.version` - binding file for <domain>.
`/var/yp/binding/<domain>.ypservers` - explicit list of servers to bind to for <domain>.

DIAGNOSTICS

Messages are sent to `syslogd`(8) using the `LOG_DAEMON` facility.

SEE ALSO

`domainname`(1), `ypcat`(1), `ypmatch`(1), `ypwhich`(1), `nis`(8), `yppoll`(8), `ypset`(8)

AUTHORS

This version of **ypbind** was originally implemented by Theo de Raadt. The ypservers support was implemented by Luke Mewburn.

NAME

ypinit — initialize NIS subsystem

SYNOPSIS

ypinit -c [*domainname*] [**-l** *server1,...,serverN*]
ypinit -m [*domainname*] [**-l** *server1,...,serverN*]
ypinit -s *master_server* [*domainname*] [**-l** *server1,...,serverN*]

DESCRIPTION

ypinit initializes the files and directories that are required for a NIS client or server.

If *domainname* isn't specified, the default domain (as returned by domainname(1)) is used.

The following options are available:

-c Create a NIS client. Initializes `/var/yp/binding/<domain>.ypservers` to contain a list of ypservers for ypbind(8) to connect to.

-l *server1,...,serverN*
Set the list of client servers from the command line rather than prompting for them interactively. The format is a comma separated list of server names with no spaces.

-m Create a master NIS server. Generates map data from local files (`/etc/master.passwd`, `/etc/group`, etc.).

-s *master_server*
Create a slave server. Downloads the maps from *master_server*, which should be the active master NIS server.

To rebuild or refresh the maps for the NIS domain <domain>, change to the `/var/yp/<domain>` directory and run **make**.

FILES

`/var/yp`	master NIS directory; contains the template makefiles.
`/var/yp/<domain>`	directory to store NIS maps for <domain>.
`/var/yp/binding/<domain>.ypservers`	list of NIS servers to bind to.

SEE ALSO

domainname(1), make(1), makedbm(8), mknetid(8), nis(8), stdethers(8), stdhosts(8), ypbind(8), yppush(8), ypserv(8)

AUTHORS

Originally written by Mats O Jansson ⟨moj@stacken.kth.se⟩. Modified by Jason R. Thorpe ⟨thorpej@NetBSD.org⟩.

NAME

yppoll — ask version of NIS map from NIS server

SYNOPSIS

yppoll [**-T**] [**-h** *host*] [**-d** *domain*] *mapname*

DESCRIPTION

yppoll asks a NIS server process for the order number and which host is the master server for *mapname*.

The options are as follows:

-T Use TCP protocol instead of UDP for request if *-h* option is present.

-h *host*
Ask the NIS server process running on *host* for information about *mapname*. If *host* is not specified, the server polled is the default server returned by ypwhich(1).

-d *domain*
Use the NIS domain *domain* instead of the default domain as returned by domainname(1).

SEE ALSO

domainname(1), ypcat(1), ypmatch(1), ypwhich(1), nis(8), ypbind(8), ypset(8)

AUTHORS

Theo de Raadt and John Brezak

NAME

yppush — force distribution of NIS map

SYNOPSIS

yppush [**-v**] [**-d** *domainname*] [**-h** *hostname*] *mapname*

DESCRIPTION

yppush is used to distribute a NIS map from a master server to any slave server in the domain. The list of servers of `domainname` are fetched from the NIS map `ypservers`.

The options are as follows:

-d *domainname*
NIS domain to use instead of the default domain.

-h *hostname* Distribute map only to one host and not to the hosts in the ypserver map.

-v Verbose. Announce what the program is doing.

SEE ALSO

`domainname`(1), `nis`(8), `ypserv`(8)

AUTHORS

Charles D. Cranor

NAME

ypserv — NIS server daemon

SYNOPSIS

ypserv [**-dfl**] [**-p** *port*]

DESCRIPTION

ypserv is a fundamental part of the network information system called NIS. This server provides information from NIS maps to the NIS clients on the network.

A NIS map is stored on the server as a `db`(3) database. A number of NIS maps is grouped together in a domain. **ypserv** determines the domains it serves by looking for a directory with the domain name in */var/yp*.

In an effort to improve the security of NIS (which has, historically, not been very good), this **ypserv** has support for libwrap-based access control. See `hosts_access`(5) for more information. The *daemon* used for access control is the name which **ypserv** was invoked as (typically "ypserv"). If a host is not allowed to query this NIS server, **ypserv** will return the NIS result code YP_NODOM. To avoid problems with DNS lookups causing **ypserv** to hang, **ypserv** disables DNS lookups for its client `hosts_access`(5) lists. The result is that **ypserv** can only use address based patterns. This also means that wildcard patterns such as LOCAL or KNOWN will not work.

The process pid of the **ypserv** process can be found in the file `/var/run/ypserv.pid`.

The options are as follows:

-d Use internet Domain Name System. If a query to map `hosts.byname` or `hosts.byaddr` fails, make a DNS query and return the result if successful.

-f Run in the foreground.

-l Enable logging of all requests.

-p *port*
Bind to the specified *port* instead of dynamically allocating one.

All messages are sent to the system log with the facility LOG_DAEMON. Error messages have the priority LOG_ERR. Refused requests are logged with the priority LOG_WARNING. All other messages are logged with the priority LOG_INFO.

FILES

`/var/run/ypserv.pid`

SEE ALSO

`syslog`(3), `hosts_access`(5), `nis`(8), `syslogd`(8), `ypbind`(8), `ypinit`(8)

AUTHORS

This implementation of **ypserv** was originally written by Mats O Jansson ⟨moj@stacken.kth.se⟩. The access control code was later re-written from scratch by
Jason R. Thorpe ⟨thorpej@NetBSD.org⟩.

NAME

ypset — tell `ypbind`(8) which NIS server process to use

SYNOPSIS

ypset [**-h** *host*] [**-d** *domain*] *server*

DESCRIPTION

ypset tells the `ypbind`(8) process on the current machine which NIS server process to communicate with. If *server* is down or is not running a NIS server process, it is not discovered until a NIS client process attempts to access a NIS map, at which time `ypbind`(8) tests the binding and takes appropriate action.

ypset is most useful for binding a NIS client that is not on the same broadcast network as the closest NIS server, but can also be used for debugging a local network's NIS configuration, testing specific NIS client programs, or binding to a specific server when there are many servers on the local network supplying NIS maps.

The options are as follows:

-h *host*
: Set the NIS binding on *host* instead of the local machine.

-d *domain*
: Use the NIS domain *domain* instead of the default domain as returned by `domainname`(1).

SEE ALSO

`domainname`(1), `ypcat`(1), `ypmatch`(1), `ypwhich`(1), `nis`(8), `ypbind`(8), `yppoll`(8)

AUTHORS

Theo de Raadt

NAME

yptest — calls different NIS routines

SYNOPSIS

yptest

DESCRIPTION

yptest is a utility written to check if the NIS server works as expected.

SEE ALSO

nis(8), ypserv(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

ypxfr — get a NIS map from NIS server

SYNOPSIS

ypxfr [**-bcf**] [**-C** *tid prog ipadd port*] [**-d** *domain*] [**-h** *host*] [**-s** *domain*]
mapname

DESCRIPTION

ypxfr is the utility in NIS that transfers maps to the local host.

The options are as follows:

-b Preserve the entry in the database informing a NIS server to use DNS to get information about unknown hosts. This option will only have effect on the maps `hosts.byname` and `hosts.byaddr`.

-c Don't send a "Clear current map" to local **ypserv** process. Useful if **ypserv** isn't running locally to avoid timeout message.

-C *tid prog ipadd port*
This option is only used by **ypserv**. This is to open communication with **yppush** on another host.

-d *domain*
Don't use default domain, use the specified domain.

-f Force map transfer, even if the master's version is older than the local copy.

-h *host*
Get map from host instead of the maps master host.

-s *domain*
Specify a source domain other than the target domain.

SEE ALSO

`nis`(8), `yppush`(8), `ypserv`(8)

AUTHORS

Mats O Jansson ⟨moj@stacken.kth.se⟩

NAME

zdump — time zone dumper

SYNOPSIS

zdump [**--version**] [**-v**] [**-c** *cutoffyear*] [*zonename ...*]

DESCRIPTION

zdump prints the current time in each *zonename* named on the command line.

These options are available:

--version
Output version information and exit.

-v For each *zonename* on the command line, print the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each detected time discontinuity, the time at one day less than the highest possible time value, and the time at the highest possible time value, Each line ends with

```
isdst=1
```

if the given time is Daylight Saving Time or

```
isdst=0
```

otherwise.

-c *cutoffyear*
Cut off the verbose output near the start of the given year.

SEE ALSO

ctime(3), tzfile(5), zic(8)

NAME

zic — time zone compiler

SYNOPSIS

zic [**--version**] [**-d** *directory*] [**-L** *leapsecondfilename*] [**-l** *localtime*]
[**-p** *posixrules*] [**-s**] [**-v**] [**-y** *command*] [*Filename ...*]

DESCRIPTION

zic reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is `-`, the standard input is read.

These options are available:

--version Output version information and exit.

-d *directory*
Create time conversion information files in the named directory rather than in the standard directory named below.

-L *leapsecondfilename*
Read leap second information from the file with the given name. If this option is not used, no leap second information appears in output files.

-l *timezone*
Use the given time zone as local time. **zic** will act as if the input contained a link line of the form

```
Link  timezone    localtime
```

-p *timezone*
Use the given time zone's rules when handling POSIX-format time zone environment variables. **zic** will act as if the input contained a link line of the form

```
Link  timezone    posixrules
```

-s Limit time values stored in output files to values that are the same whether they're taken to be signed or unsigned. You can use this option to generate SVVS-compatible files.

-v Complain if a year that appears in a data file is outside the range of years representable by `time`(3) values. Also complain if a time of 24:00 (which cannot be handled by pre-1998 versions of **zic**) appears in the input.

-y *command*
Use the given *command* rather than *yearistype* when checking year types (see below).

Input lines are made up of fields. Fields are separated from one another by any number of white space characters. Leading and trailing white space on input lines is ignored. An unquoted sharp character (#) in the input introduces a comment which extends to the end of the line the sharp character appears on. White space characters and sharp characters may be enclosed in double quotes (") if they're to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, and link lines.

A rule line has the form

```
Rule  NAME  FROM  TO    TYPE  IN    ON    AT    SAVE  LETTER/S
```

For example:

```
Rule  US    1967  1973  −     Apr   lastSun     2:00  1:00  D
```

The fields that make up a rule line are:

NAME	Gives the (arbitrary) name of the set of rules this rule is part of.
FROM	Gives the first year in which the rule applies. Any integer year can be supplied; the Gregorian calendar is assumed. The word *minimum* (or an abbreviation) means the minimum year representable as an integer. The word *maximum* (or an abbreviation) means the maximum year representable as an integer. Rules

can describe times that are not representable as time values, with the unrepresentable times ignored; this allows rules to be portable among hosts with differing time value types.

TO	Gives the final year in which the rule applies. In addition to *minimum* and *maximum* (as above), the word *only* (or an abbreviation) may be used to repeat the value of the *FROM* field.
TYPE	Gives the type of year in which the rule applies. If *TYPE* is - then the rule applies in all years between *FROM* and *TO* inclusive. If *TYPE* is something else, then **zic** executes the command **yearistype** *year type* to check the type of a year: an exit status of zero is taken to mean that the year is of the given type; an exit status of one is taken to mean that the year is not of the given type.
IN	Names the month in which the rule takes effect. Month names may be abbreviated.
ON	Gives the day on which the rule takes effect. Recognized forms include: (see below) Names of days of the week may be abbreviated or spelled out in full. Note that there must be no spaces within the *ON* field.

5	the fifth of the month
lastSun	the last Sunday in the month
lastMon	the last Monday in the month
Sun≥8	first Sunday on or after the eighth
Sun≤25	last Sunday on or before the 25th

AT Gives the time of day at which the rule takes effect. Recognized forms include:

2	time in hours
2:00	time in hours and minutes
15:00	24-hour format time (for times after noon)
1:28:14	time in hours, minutes, and seconds
−	equivalent to 0

where hour 0 is midnight at the start of the day, and hour 24 is midnight at the end of the day. Any of these forms may be followed by the letter *w* if the given time is local "wall clock" time, *s* if the given time is local "standard" time, or *u* (or *g* or *z*) if the given time is universal time; in the absence of an indicator, wall clock time is assumed.

SAVE	Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the *AT* field (although, of course, the *w* and *s* suffixes are not used).
LETTER/S	Gives the "variable part" (for example, the "S" or "D" in "EST" or "EDT") of time zone abbreviations to be used when this rule is in effect. If this field is -, the variable part is null.

A zone line has the form

```
     Zone  NAME        GMTOFF      RULES/SAVE  FORMAT      [UNTIL]
```

For example:

```
     Zone  Australia/Adelaide      9:30  Aus   CST   1971  Oct
     31 2:00
```

The fields that make up a zone line are:

NAME	The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF	The amount of time to add to UTC to get standard time in this zone. This field has the same format as the *AT* and *SAVE* fields of rule lines; begin the field with a minus sign if time must be subtracted from UTC.
RULES/SAVE	The name of the rule(s) that apply in the time zone or, alternatively, an amount of time to add to local standard time. If this field is - then standard time always applies in the time zone.
FORMAT	The format for time zone abbreviations in this time zone. The pair of characters *%s* is used to show where the "variable part" of the time zone abbreviation goes. Alternatively, a slash (`/`) separates standard and daylight abbreviations.
UNTIL	The time at which the UTC offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. If this is specified, the time zone information is generated from the given UTC offset and rule change until the time specified. The month, day, and time of day have the same format as the IN, ON, and AT columns of a rule; trailing columns can be omitted, and default to the earliest possible value for the missing columns.

The next line must be a "continuation" line; this has the same form as a zone line except that the string "Zone" and the name are omitted, as the continuation line will place information starting at the time specified as the *UNTIL* field in the previous line in the file used by the previous line. Continuation lines may contain an *UNTIL* field, just as zone lines do, indicating that the next line is a further continuation.

A link line has the form

```
Link  LINK-FROM   LINK-TO
```

For example:

```
Link  Europe/Istanbul   Asia/Istanbul
```

The *LINK-FROM* field should appear as the *NAME* field in some zone line; the *LINK-TO* field is used as an alternative name for that zone.

Except for continuation lines, lines may appear in any order in the input.

Lines in the file that describes leap seconds have the following form:

```
Leap  YEAR  MONTH DAY   HH:MM:SS    CORR  R/S
```

For example:

```
Leap  1974  Dec   31    23:59:60    +     S
```

The *YEAR*, *MONTH*, *DAY*, and *HH:MM:SS* fields tell when the leap second happened. The *CORR* field should be "+" if a second was added or "-" if a second was skipped. The *R/S* field should be (an abbreviation of) "Stationary" if the leap second time given by the other fields should be interpreted as UTC or (an abbreviation of) "Rolling" if the leap second time given by the other fields should be interpreted as local wall clock time.

NOTES

For areas with more than two types of local time, you may need to use local standard time in the *AT* field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

FILES

`/usr/share/zoneinfo` - standard directory used for created files

SEE ALSO

`ctime`(3), `tzfile`(5), `zdump`(8)