NAME

intro — introduction to online games

DESCRIPTION

This section contains various games and tools whose primary value is amusement.

HISTORY

intro appeared in NetBSD 1.4.

NAME

adventure — an exploration game

SYNOPSIS

adventure [saved-file]

DESCRIPTION

The object of the game is to locate and explore Colossal Cave, find the treasures hidden there, and bring them back to the building with you. The program is self-descriptive to a point, but part of the game is to discover its rules.

To terminate a game, enter “quit”; to save a game for later resumption, enter “suspend”.

NAME

arithmetic — quiz on simple arithmetic

SYNOPSIS

arithmetic [**−o** *+−x/*] [**−r** *range*]

DESCRIPTION

arithmetic asks you to solve problems in simple arithmetic. Each question must be answered correctly before going on to the next. After every 20 problems, it prints the score so far and the time taken. You can quit at any time by typing the interrupt or end-of-file character.

The options are as follows:

−o By default, **arithmetic** asks questions on addition of numbers from 0 to 10, and corresponding subtraction. By supplying one or more of the characters *+−x/*, you can ask for problems in addition, subtraction, multiplication, and division, respectively. If you give one of these characters more than once, that kind of problem will be asked correspondingly more often.

−r If a *range* is supplied, **arithmetic** selects the numbers in its problems in the following way. For addition and multiplication, the numbers to be added or multiplied are between 0 and *range*, inclusive. For subtraction and division, both the required result and the number to divide by or subtract will be between 0 and *range*. (Of course, **arithmetic** will not ask you to divide by 0.) The default is 10.

When you get a problem wrong, **arithmetic** will remember the numbers involved, and will tend to select those numbers more often than others, in problems of the same sort. Eventually it will forgive and forget.

arithmetic cannot be persuaded to tell you the right answer. You must work it out for yourself.

DIAGNOSTICS

“What?” if you get a question wrong. “Right!” if you get it right. “Please type a number.” if **arithmetic** doesn’t understand what you typed.

SEE ALSO

bc(1), dc(1)

NAME

atc — air traffic controller game

SYNOPSIS

atc [**-u?lstp**] [**-gf** *game name*] [**-r** *random seed*]

DESCRIPTION

atc lets you try your hand at the nerve wracking duties of the air traffic controller without endangering the lives of millions of travelers each year. Your responsibilities require you to direct the flight of jets and prop planes into and out of the flight arena and airports. The speed (update time) and frequency of the planes depend on the difficulty of the chosen arena.

OPTIONS

-u Print the usage line and exit.

-? Same as **-u**.

-l Print a list of available games and exit. The first game name printed is the default game.

-s Print the score list (formerly the Top Ten list).

-t Same as **-s**.

-p Print the path to the special directory where **atc** expects to find its private files. This is used during the installation of the program.

-g *game*
Play the named game. If the game listed is not one of the ones printed from the **-l** option, the default game is played.

-f *game*
Same as **-g**.

-r *seed*
Set the random seed. The purpose of this flag is questionable.

GOALS

Your goal in **atc** is to keep the game going as long as possible. There is no winning state, except to beat the times of other players. You will need to: launch planes at airports (by instructing them to increase their altitude); land planes at airports (by instructing them to go to altitude zero when exactly over the airport); and maneuver planes out of exit points.

Several things will cause the end of the game. Each plane has a destination (see information area), and sending a plane to the wrong destination is an error. Planes can run out of fuel, or can collide. Collision is defined as adjacency in all three dimensions. A plane leaving the arena in any other way than through its destination exit is an error as well.

Scores are sorted in order of the number of planes safe. The other statistics are provided merely for fun. There is no penalty for taking longer than another player (except in the case of ties).

Suspending a game is not permitted. If you get a talk message, tough. When was the last time an Air Traffic Controller got called away to the phone?

THE DISPLAY

Depending on the terminal you run **atc** on, the screen will be divided into 4 areas. It should be stressed that the terminal driver portion of the game was designed to be reconfigurable, so the display format can vary depending on the version you are playing. The descriptions here are based on the ascii version of the game.

The game rules and input format, however, should remain consistent. Control-L redraws the screen, should it become muddled.

RADAR

The first screen area is the radar display, showing the relative locations of the planes, airports, standard entry/exit points, radar beacons, and ‘‘lines’’ which simply serve to aid you in guiding the planes.

Planes are shown as a single letter with an altitude. If the numerical altitude is a single digit, then it represents thousands of feet. Some distinction is made between the prop planes and the jets. On ascii terminals, prop planes are represented by a upper case letter, jets by a lower case letter.

Airports are shown as a number and some indication of the direction planes must be going to land at the airport. On ascii terminals, this is one of ‘ˆ’, ‘>’, ‘<’, and ‘v’, to indicate north (0 degrees), east (90), west (270) and south (180), respectively. The planes will also take off in this direction.

Beacons are represented as circles or asterisks and a number. Their purpose is to offer a place of easy reference to the plane pilots. See **THE DELAY COMMAND** section below.

Entry/exit points are displayed as numbers along the border of the radar screen. Planes will enter the arena from these points without warning. These points have a direction associated with them, and planes will always enter the arena from this direction. On the ascii version of **`atc`**, this direction is not displayed. It will become apparent what this direction is as the game progresses.

Incoming planes will always enter at the same altitude: 7000 feet. For a plane to successfully depart through an entry/exit point, it must be flying at 9000 feet. It is not necessary for the planes to be flying in any particular direction when they leave the arena (yet).

INFORMATION AREA

The second area of the display is the information area, which lists the time (number of updates since start), and the number of planes you have directed safely out of the arena. Below this is a list of planes currently in the air, followed by a blank line, and then a list of planes on the ground (at airports). Each line lists the plane name and its current altitude, an optional asterisk indicating low fuel, the plane’s destination, and the plane’s current command. Changing altitude is not considered to be a command and is therefore not displayed. The following are some possible information lines:

```
B4*A0: Circle @ b1
g7 E4: 225
```

The first example shows a prop plane named ‘B’ that is flying at 4000 feet. It is low on fuel (note the ‘*’). Its destination is Airport #0. The next command it expects to do is circle when it reaches Beacon #1. The second example shows a jet named ‘g’ at 7000 feet, destined for Exit #4. It is just now executing a turn to 225 degrees (South-West).

INPUT AREA

The third area of the display is the input area. It is here that your input is reflected. See the **INPUT** heading of this manual for more details.

AUTHOR AREA

This area is used simply to give credit where credit is due. :-)

INPUT

A command completion interface is built into the game. At any time, typing ‘?’ will list possible input characters. Typing a backspace (your erase character) backs up, erasing the last part of the command. When a command is complete, a return enters it, and any semantic checking is done at that time. If no errors are detected, the command is sent to the appropriate plane. If an error is discovered during the check, the offending statement will be underscored and a (hopefully) descriptive message will be printed under it.

The command syntax is broken into two parts: *Immediate Only* and *Delayable* commands. *Immediate Only* commands happen on the next update. *Delayable* commands also happen on the next update unless they are followed by an optional predicate called the *Delay* command.

In the following tables, the syntax *[0−9]* means any single digit, and ⟨*dir*⟩ refers to a direction, given by the keys around the ‘s’ key: ‘‘wedcxzaq’’. In absolute references, ‘q’ refers to North-West or 315 degrees, and ‘w’ refers to North, or 0 degrees. In relative references, ‘q’ refers to −45 degrees or 45 degrees left, and ‘w’ refers to 0 degrees, or no change in direction.

All commands start with a plane letter. This indicates the recipient of the command. Case is ignored.

IMMEDIATE ONLY COMMANDS

a [cd+-] *number*

Altitude: Change a plane’s altitude, possibly requesting takeoff. ‘+’ and ‘-’ are the same as ‘c’ and ‘d’.

a *number*	Climb or descend to the given altitude (in thousands of feet).
ac *number*	Climb: relative altitude change.
ad *number*	Descend: relative altitude change.

m	Mark: Display in highlighted mode. Plane and command information is displayed normally.
i	Ignore: Do not display highlighted. Command is displayed as a line of dashes if there is no command.
u	Unmark: Same as ignore, but if a delayed command is processed, the plane will become marked. This is useful if you want to forget about a plane during part, but not all, of its journey.

DELAYABLE COMMANDS

c [lr]

Circle: Have the plane circle.

cl	Left: Circle counterclockwise.
cr	Right: Circle clockwise (default).

t [l-r+LR] [dir] or tt [abe∗] *number*

Turn: Change direction.

t<dir>	Turn to direction: Turn to the absolute compass heading given. The shortest turn will be taken.
tl [dir]	Left: Turn counterclockwise: 45 degrees by default, or the amount specified in ⟨dir⟩ (not *to* ⟨dir⟩.) ‘w’ (0 degrees) is no turn. ‘e’ is 45 degrees; ‘q’ gives −45 degrees counterclockwise, that is, 45 degrees clockwise.
t- [dir]	Same as left.
tr [dir]	Right: Turn clockwise, 45 degrees by default, or the amount specified in ⟨dir⟩.
t+ [dir]	Same as right.
tL	Hard left: Turn counterclockwise 90 degrees.
tR	Hard right: Turn clockwise 90 degrees.
tt [abe∗]	Towards: Turn towards a beacon, airport or exit. The turn is just an estimate.
tta *number*	Turn towards the given airport.
ttb *number*	Turn towards the specified beacon.
tte *number*	Turn towards an exit.
tt∗ *number*	Same as ttb.

THE DELAY COMMAND

The *Delay* (a/@) command may be appended to any *Delayable* command. It allows the controller to instruct a plane to do an action when the plane reaches a particular beacon (or other objects in future versions).

ab *number*

Do the delayable command when the plane reaches the specified beacon. The ‘b’ for ‘‘beacon’’ is redundant to allow for expansion. ‘@’ can be used instead of ‘a’.

MARKING, UNMARKING AND IGNORING

Planes are *marked* by default when they enter the arena. This means they are displayed in highlighted mode on the radar display. A plane may also be either *unmarked* or *ignored*. An *ignored* plane is drawn in unhighlighted mode, and a line of dashes is displayed in the command field of the information area. The plane will remain this way until a mark command has been issued. Any other command will be issued, but the command line will return to a line of dashes when the command is completed.

An *unmarked* plane is treated the same as an *ignored* plane, except that it will automatically switch to *marked* status when a delayed command has been processed. This is useful if you want to forget about a plane for a while, but its flight path has not yet been completely set.

As with all of the commands, marking, unmarking and ignoring will take effect at the beginning of the next update. Do not be surprised if the plane does not immediately switch to unhighlighted mode.

EXAMPLES

atlab1	Plane A: turn left at beacon #1
cc	Plane C: circle
gtte4ab2	Plane G: turn towards exit #4 at beacon #2
ma+2	Plane M: altitude: climb 2000 feet
stq	Plane S: turn to 315
xi	Plane X: ignore

OTHER INFORMATION

- Jets move every update; prop planes move every other update.
- All planes turn at most 90 degrees per movement.
- Planes enter at 7000 feet and leave at 9000 feet.
- Planes flying at an altitude of 0 crash if they are not over an airport.
- Planes waiting at airports can only be told to take off (climb in altitude).
- Pressing return (that is, entering an empty command) will perform the next update immediately. This allows you to ‘‘fast forward’’ the game clock if nothing interesting is happening.

NEW GAMES

The `Game_List` file lists the currently available play fields. New field description file names must be placed in this file to be playable. If a player specifies a game not in this file, his score will not be logged.

The game field description files are broken into two parts. The first part is the definition section. Here, the four tunable game parameters must be set. These variables are set with the syntax:

```
variable = number;
```

Variable may be one of: `update`, indicating the number of seconds between forced updates; `newplane`, indicating (about) the number of updates between new plane entries; `width`, indicating the width of the play field; or `height`, indicating the height of the play field.

The second part of the field description files describes the locations of the exits, the beacons, the airports and the lines. The syntax is as follows:

```
     beacon:   (x y) ... ;
     airport:  (x y direction) ... ;
     exit:     (x y direction) ... ;
     line:     [ (x1 y1) (x2 y2) ] ... ;
```

For beacons, a simple x, y coordinate pair is used (enclosed in parenthesis). Airports and exits require a third value, which is one of the directions *wedcxzaq*. For airports, this is the direction that planes must be going to take off and land, and for exits, this is the direction that planes will be going when they *enter* the arena. This may not seem intuitive, but as there is no restriction on direction of exit, this is appropriate. Lines are slightly different, since they need two coordinate pairs to specify the line endpoints. These endpoints must be enclosed in square brackets.

All statements are semi-colon (;) terminated. Multiple item statements accumulate. Each definition must occur exactly once, before any item statements. Comments begin with a hash (#) symbol and terminate with a newline. The coordinates are between zero and width-1 and height-1 inclusive. All of the exit coordinates must lie on the borders, and all of the beacons and airports must lie inside of the borders. Line endpoints may be anywhere within the field, so long as the lines are horizontal, vertical or *exactly* diagonal.

FIELD FILE EXAMPLE

```
# This is the default game.

update = 5;
newplane = 5;
width = 30;
height = 21;

exit:          ( 12  0 x ) ( 29  0 z ) ( 29  7 a ) ( 29 17 a )
               (  9 20 e ) (  0 13 d ) (  0  7 d ) (  0  0 c ) ;

beacon:        ( 12  7 ) ( 12 17 ) ;

airport:       ( 20 15 w ) ( 20 18 d ) ;

line:          [ (  1  1 ) (  6  6 ) ]
               [ ( 12  1 ) ( 12  6 ) ]
               [ ( 13  7 ) ( 28  7 ) ]
               [ ( 28  1 ) ( 13 16 ) ]
               [ (  1 13 ) ( 11 13 ) ]
               [ ( 12  8 ) ( 12 16 ) ]
               [ ( 11 18 ) ( 10 19 ) ]
               [ ( 13 17 ) ( 28 17 ) ]
               [ (  1  7 ) ( 11  7 ) ] ;
```

FILES

Files are kept in a special directory. See the **OPTIONS** section for a way to print this path out. It is normally `/usr/share/games/atc`.

This directory contains the file `Game_List`, which holds the list of playable games, as well as the games themselves.

The scores are kept in `/var/games/atc_score`.

AUTHORS

Ed James, UC Berkeley: edjames@ucbvax.berkeley.edu, ucbvax!edjames

This game is based on someone's description of the overall flavor of a game written for some unknown PC many years ago, maybe.

BUGS

The screen sometimes refreshes after you have quit.

NAME

backgammon — the game of backgammon

SYNOPSIS

backgammon [**-**] [**-nrwb**] [**-pr**] [**-pw**] [**-pb**] [**-t** *term*] [**-s** *file*]

DESCRIPTION

This program lets you play backgammon against the computer or against a "friend". All commands are only one letter, so you don't need to type a carriage return, except at the end of a move. The program is mostly self-explanatory, so that a question mark (?) will usually get some help. If you answer 'y' when the program asks if you want the rules, you will get text explaining the rules of the game, some hints on strategy, instructions on how to use the program, and a tutorial consisting of a practice game against the computer. A description of how to use the program can be obtained by answering 'y' when it asks if you want instructions.

The possible arguments for backgammon (most are unnecessary but some are very convenient) consist of:

-n	don't ask for rules or instructions
-r	player is red (implies n)
-w	player is white (implies n)
-b	two players, red and white (implies n)
-pr	print the board before red's turn
-pw	print the board before white's turn
-pb	print the board before both player's turn
-t *term*	terminal is type *term*, uses /usr/share/misc/termcap
-s *file*	recover previously saved game from *file*

Any unrecognized arguments are ignored. An argument of a lone '-' gets a description of possible arguments.

If *term* has capabilities for direct cursor movement (see termcap(5)) **backgammon** ''fixes'' the board after each move, so the board does not need to be reprinted, unless the screen suffers some horrendous malady. Also, any 'p' option will be ignored. (The 't' option is not necessary unless the terminal type does not match the entry in the /usr/share/misc/termcap data base.)

QUICK REFERENCE

When the program prompts by typing only your color, type a space or carriage return to roll, or

d	to double
p	to print the board
q	to quit
s	to save the game for later

When the program prompts with 'Move:', type

p	to print the board

`q` to quit

`s` to save the game

or a *move*, which is a sequence of

`s-f` move from **`s`** to **`f`**

`s/r` move one man on **`s`** the roll **`r`** separated by commas or spaces and ending with a newline. Available abbreviations are

  `s-f1-f2`
    means **`s-f1,f1-f2`**

  `s/r1r2` means **`s/r1,s/r2`**

Use **`b`** for bar and **`h`** for home, or 0 or 25 as appropriate.

FILES

`/usr/games/teachgammon`	rules and tutorial
`/usr/share/misc/termcap`	terminal capabilities

AUTHORS

Alan Char

BUGS

The program’s strategy needs much work.

NAME

banner — print large banner on printer

SYNOPSIS

banner [**-w** *width*] *message ...*

DESCRIPTION

banner prints a large, high quality banner on the standard output. If the message is omitted, it prompts for and reads one line of its standard input. If **-w** is given, the output is scrunched down from a width of 132 to *width*, suitable for a narrow terminal. For example:

```
/usr/games/banner -w 80 Hello
```

The output should be printed on paper of the appropriate width, with no breaks between the pages.

AUTHORS

Mark Horton

BUGS

Several ASCII characters are not defined, notably <, >, [,], \, ^, _, {, }, |, and ~. Also, the characters ", ', and & are funny looking (but in a useful way.)

The **-w** option is implemented by skipping some rows and columns. The smaller it gets, the grainier the output. Sometimes it runs letters together.

NAME

battlestar — a tropical adventure game

SYNOPSIS

battlestar [**-r**] [*saved-file*]

DESCRIPTION

battlestar is an adventure game in the classic style. However, it's slightly less of a puzzle and more a game of exploration. There are a few magical words in the game, but on the whole, simple English should suffice to make one's desires understandable to the parser.

THE SETTING

In the days before the darkness came, when battlestars ruled the heavens...

```
Three He made and gave them to His daughters,
Beautiful nymphs, the goddesses of the waters.
One to bring good luck and simple feats of wonder,
Two to wash the lands and churn the waves asunder,
Three to rule the world and purge the skies with thunder.
```

In those times great wizards were known and their powers were beyond belief. They could take any object from thin air, and, uttering the word 'su' could disappear.

In those times men were known for their lust for gold and desire to wear fine weapons. Swords and coats of mail were fashioned that could withstand a laser blast.

But when the darkness fell, the rightful reigns were toppled. Swords and helms and heads of state went rolling across the grass. The entire fleet of battlestars was reduced to a single ship.

SAMPLE COMMANDS

```
take    ---     take an object
drop    ---     drop an object

wear    ---     wear an object you are holding
draw    ---     carry an object you are wearing

put on  ---     take an object and wear it
take off --     draw an object and drop it

throw  <object> <direction>

!       <shell esc>
```

IMPLIED OBJECTS

```
>-: take watermelon
watermelon:
Taken.
>-: eat
watermelon:
Eaten.
>-: take knife and sword and apple, drop all
knife:
Taken.
broadsword:
```

```
Taken.
apple:
Taken.
knife:
Dropped.
broadsword:
Dropped.
apple:
Dropped.
>-: get
knife:
Taken.
```

Notice that the "shadow" of the next word stays around if you want to take advantage of it. That is, saying "take knife" and then "drop" will drop the knife you just took.

SCORE & INVEN

The two commands “score” and “inven” will print out your current status in the game.

SAVING A GAME

The command “save” will save your game in a file called `.Bstar` in your home directory by default. You can recover a saved game by using the **-r** option when you start up the game, or by giving the name of the saved file as an argument. Save files will be saved to and restored from your home directory unless a path is specified - i.e., “`battlestar -r savedgame`” will look for `savedgame` in your home directory, but “`battlestar -r ./savedgame`” will look in the current directory. “`battlestar -r`” will look for the default file, `.Bstar` in your home directory.

DIRECTIONS

The compass directions N, S, E, and W can be used if you have a compass. If you don’t have a compass, you’ll have to say R, L, A, or B, which stand for Right, Left, Ahead, and Back. Directions printed in room descriptions are always printed in R, L, A, and B relative directions.

HISTORY

I wrote Battlestar in 1979 in order to experiment with the niceties of the C Language. Most interesting things that happen in the game are hardwired into the code, so don’t send me any hate mail about it! Instead, enjoy art for art’s sake!

AUTHORS

David Riggle

INSPIRATION & ASSISTANCE

Chris Guthrie
Peter Da Silva
Kevin Brown
Edward Wang
Ken Arnold & Company

BUGS

Countless.

FAN MAIL

Send to edward%ucbarpa@berkeley.arpa, chris%ucbcory@berkeley.arpa, riggle.pa@xerox.arpa.

NAME

bcd, **ppt**, **morse** — reformat input as punch cards, paper tape or morse code

SYNOPSIS

bcd [*string ...*]
ppt [**−d**|*string ...*]
morse [**−ds** *string ...*]

DESCRIPTION

The **bcd**, **ppt** and **morse** commands read the given input and reformat it in the form of punched cards, paper tape or morse code respectively. Acceptable input are command line arguments or the standard input.

Available option:

−s The **−s** option for morse produces dots and dashes rather than words.

−d Decode **ppt** output, or **morse** output consisting of dots and slashes (as generated by using the **−s** option).

SEE ALSO

ISO 1681:1973: Information processing--Unpunched paper cards--Specification.

ISO 1682:1973: Information processing--80 columns punched paper cards--Dimensions and location of rectangular punched holes.

ECMA-10: ECMA Standard for Data Interchange on Punched Tape.

ITU-T Recommendation F.1: Operational provisions for the international public telegram service, Division B, I. Morse code.

NAME

boggle — word search game

SYNOPSIS

boggle [**-bd**] [**-s** *seed*] [**-t** *time*] [**-w** *length*] [+ [+]] [boardspec]

DESCRIPTION

The object of **boggle** is to find as many words as possible on the Boggle board within the three minute time limit. A Boggle board is a four by four arrangement of Boggle cubes, each side of each cube displaying a letter of the alphabet or 'qu'. Words are formed by finding a sequence of cubes (letters) that are in the game's dictionary. The (N+1)th cube in the word must be horizontally, vertically, or diagonally adjacent to the Nth cube. Cubes cannot be reused. Words consist solely of lower case letters and must be at least 3 letters long.

Command line flags can be given to change the rules of the game.

-b Run **boggle** in batch mode. A *boardspec* must also be given. The dictionary is read from stdin and a list of words appearing in *boardspec* is printed to stdout.

-d Enable debugging output.

-s *seed* Specify a seed *seed* other than the time of day.

-t *time* Set the time limit for each game from the default 3 minutes to *time* seconds.

-w *length*
Change the minimum word length from 3 letters to *length*.

+ This flag allows a cube to be used multiple times, but not in succession.

++ This flag allows the same cubes to be considered adjacent to itself.

boardspec A starting board position can be specified on the command line by listing the board left to right and top to bottom.

Help is available during play by typing '**?**'. More detailed information on the game is given there.

AUTHORS

Boggle is a trademark of Parker Brothers.

Barry Brachman

Dept. of Computer Science

University of British Columbia

BUGS

If there are a great many words in the cube the final display of the words may scroll off of the screen. (On a 25 line screen about 130 words can be displayed.)

No word can contain a 'q' that is not immediately followed by a 'u'.

When using the **+** or **++** options the display of words found in the board doesn't indicate reused cubes.

The dictionary that NetBSD installs omits many words that belong in the English language, most notably inflected forms.

NAME

caesar, **rot13** — decrypt caesar ciphers

SYNOPSIS

caesar [*rotation*]

DESCRIPTION

The **caesar** utility attempts to decrypt caesar ciphers using English letter frequency statistics. **caesar** reads from the standard input and writes to the standard output.

The optional numerical argument *rotation* may be used to specify a specific rotation value.

The frequency (from most common to least) of English letters is as follows:

ETAONRISHDLFCMUGPYWBVKXJQZ

Their frequencies as a percentage are as follows:

E(13), T(10.5), A(8.1), O(7.9), N(7.1), R(6.8), I(6.3), S(6.1), H(5.2), D(3.8), L(3.4), F(2.9), C(2.7), M(2.5), U(2.4), G(2), P(1.9), Y(1.9), W(1.5), B(1.4), V(.9), K(.4), X(.15), J(.13), Q(.11), Z(.07).

Rotated postings to USENET and some of the databases used by the `fortune`(6) program are rotated by 13 characters.

NAME

canfield, **cfscores** — the solitaire card game canfield

SYNOPSIS

canfield
cfscores [**−a**] [*user*]

DESCRIPTION

If you have never played solitaire before, it is recommended that you consult a solitaire instruction book. In Canfield, tableau cards may be built on each other downward in alternate colors. An entire pile must be moved as a unit in building. Top cards of the piles are available to be played on foundations, but never into empty spaces.

Spaces must be filled from the stock. The top card of the stock also is available to be played on foundations or built on tableau piles. After the stock is exhausted, tableau spaces may be filled from the talon and the player may keep them open until he wishes to use them.

Cards are dealt from the hand to the talon by threes and this repeats until there are no more cards in the hand or the player quits. To have cards dealt onto the talon the player types '**ht**' for his move. Foundation base cards are also automatically moved to the foundation when they become available.

The command '**c**' causes **canfield** to maintain card counting statistics on the bottom of the screen. When properly used this can greatly increase one's chances of winning.

The rules for betting are somewhat less strict than those used in the official version of the game. The initial deal costs $13. You may quit at this point or inspect the game. Inspection costs $13 and allows you to make as many moves as possible without moving any cards from your hand to the talon. (The initial deal places three cards on the talon; if all these cards are used, three more are made available.) Finally, if the game seems interesting, you must pay the final installment of $26. At this point you are credited at the rate of $5 for each card on the foundation; as the game progresses you are credited with $5 for each card that is moved to the foundation. Each run through the hand after the first costs $5. The card counting feature costs $1 for each unknown card that is identified. If the information is toggled on, you are only charged for cards that became visible since it was last turned on. Thus the maximum cost of information is $34. Playing time is charged at a rate of $1 per minute.

With no arguments, the program **cfscores** prints out the current status of your canfield account. If a *user* name is specified, it prints out the status of their canfield account. If the **−a** flag is specified, it prints out the canfield accounts for all users that have played the game since the database was set up.

FILES

`/usr/games/canfield`	the game itself
`/usr/games/cfscores`	the database printer
`/var/games/cfscores`	the database of scores

AUTHORS

Originally written: Steve Levine.

Further random hacking by: Steve Feldman, Kirk McKusick, Mikey Olson, and Eric Allman.

BUGS

It is impossible to cheat.

NAME

`ching` — the book of changes and other cookies

SYNOPSIS

`ching` [hexagram]

DESCRIPTION

The *I Ching* or *Book of Changes* is an ancient Chinese oracle that has been in use for centuries as a source of wisdom and advice.

The text of the *oracle* (as it is sometimes known) consists of sixty-four *hexagrams*, each symbolized by a particular arrangement of six straight (−−−) and broken (− −) lines. These lines have values ranging from six through nine, with the even values indicating the broken lines.

Each hexagram consists of two major sections. The **Judgement** relates specifically to the matter at hand (e.g., “It furthers one to have somewhere to go.”) while the **Image** describes the general attributes of the hexagram and how they apply to one’s own life (“Thus the superior man makes himself strong and untiring.”)

When any of the lines have the values six or nine, they are moving lines; for each there is an appended judgement which becomes significant. Furthermore, the moving lines are inherently unstable and change into their opposites; a second hexagram (and thus an additional judgement) is formed.

Normally, one consults the oracle by fixing the desired question firmly in mind and then casting a set of changes (lines) using yarrow−stalks or tossed coins. The resulting hexagram will be the answer to the question.

Using an algorithm suggested by S. C. Johnson, the UNIX *oracle* simply reads a question from the standard input (up to an EOF) and hashes the individual characters in combination with the time of day, process id and any other magic numbers which happen to be lying around the system. The resulting value is used as the seed of a random number generator which drives a simulated coin−toss divination. The answer is then piped through `nroff`(1) for formatting and will appear on the standard output.

For those who wish to remain steadfast in the old traditions, the oracle will also accept the results of a personal divination using, for example, coins. To do this, cast the change and then type the resulting line values as an argument.

The impatient modern may prefer to settle for Chinese cookies; try `fortune`(6).

DIAGNOSTICS

The great prince issues commands,

Founds states, vests families with fiefs.

Inferior people should not be employed.

SEE ALSO

It furthers one to see the great man.

BUGS

Waiting in the mud

Brings about the arrival of the enemy.

If one is not extremely careful,

Somebody may come up from behind and strike him.

Misfortune.

NAME

`countmail` — be obnoxious about how much mail you have

SYNOPSIS

`countmail`

DESCRIPTION

The **`countmail`** program counts your mail and tells you about it rather obnoxiously.

SEE ALSO

`from`(1)

HISTORY

The **`countmail`** program first appeared in NetBSD 1.3.

AUTHORS

`countmail` was first written by Noah Friedman ⟨friedman@splode.com⟩ in 1993. This version was written by Charles M. Hannum ⟨mycroft@NetBSD.org⟩.

CAVEATS

The read loop is horrendously slow on every shell implementation tried. **`countmail`** uses `from`(1) and `wc`(1) instead, though these are not shell builtins.

NAME

cribbage — the card game cribbage

SYNOPSIS

cribbage [**−eqr**]

DESCRIPTION

cribbage plays the card game cribbage, with the program playing one hand and the user the other. The program will initially ask the user if the rules of the game are needed – if so, it will print out the appropriate section from *According to Hoyle* with more(1).

cribbage options include:

−e When the player makes a mistake scoring his hand or crib, provide an explanation of the correct score. (This is especially useful for beginning players.)

−q Print a shorter form of all messages – this is only recommended for users who have played the game without specifying this option.

−r Instead of asking the player to cut the deck, the program will randomly cut the deck.

cribbage first asks the player whether he wishes to play a short game (“once around”, to 61) or a long game (“twice around”, to 121). A response of ‘**s**’ will result in a short game, any other response will play a long game.

At the start of the first game, the program asks the player to cut the deck to determine who gets the first crib. The user should respond with a number between 0 and 51, indicating how many cards down the deck is to be cut. The player who cuts the lower ranked card gets the first crib. If more than one game is played, the loser of the previous game gets the first crib in the current game.

For each hand, the program first prints the player’s hand, whose crib it is, and then asks the player to discard two cards into the crib. The cards are prompted for one per line, and are typed as explained below.

After discarding, the program cuts the deck (if it is the player’s crib) or asks the player to cut the deck (if it’s its crib); in the latter case, the appropriate response is a number from 0 to 39 indicating how far down the remaining 40 cards are to be cut.

After cutting the deck, play starts with the non-dealer (the person who doesn’t have the crib) leading the first card. Play continues, as per cribbage, until all cards are exhausted. The program keeps track of the scoring of all points and the total of the cards on the table.

After play, the hands are scored. The program requests the player to score his hand (and the crib, if it is his) by printing out the appropriate cards (and the cut card enclosed in brackets). Play continues until one player reaches the game limit (61 or 121).

A carriage return when a numeric input is expected is equivalent to typing the lowest legal value; when cutting the deck this is equivalent to choosing the top card.

Cards are specified as rank followed by suit. The ranks may be specified as one of: ‘a’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘t’, ‘j’, ‘q’, and ‘k’, or alternatively, one of: ‘ace’, ‘two’, ‘three’, ‘four’, ‘five’, ‘six’, ‘seven’, ‘eight’, ‘nine’, ‘ten’, ‘jack’, ‘queen’, and ‘king’. Suits may be specified as: ‘s’, ‘h’, ‘d’, and ‘c’, or alternatively as: ‘spades’, ‘hearts’, ‘diamonds’, and ‘clubs’. A card may be specified as: “⟨rank⟩ ⟨suit⟩”, or: “⟨rank⟩ of ⟨suit⟩”. If the single letter rank and suit designations are used, the space separating the suit and rank may be left out. Also, if only one card of the desired rank is playable, typing the rank is sufficient. For example, if your hand was “2H, 4D, 5C, 6H, JC, and KD” and it was desired to discard the king of diamonds, any of the following could be typed: ‘k’, ‘king’, ‘kd’, ‘k d’, ‘k of d’, ‘king d’, ‘king of d’, ‘k diamonds’, ‘k of diamonds’, ‘king diamonds’, ‘king of diamonds’.

FILES

```
/usr/games/cribbage
/var/games/criblog
/usr/share/games/cribbage.instr
```

AUTHORS

Earl T. Cohen wrote the logic. Ken Arnold added the screen oriented interface.

NAME

dab — Dots and Boxes game

SYNOPSIS

dab [**−aw**] [**−n** *ngames*] [**−p** ⟨*c*|*h*⟩⟨*c*|*h*⟩] [*xdim* [*ydim*]]

DESCRIPTION

dab is a game where each player tries to complete the most boxes. A turn consists of putting one border of a box; the player setting the fourth and final border of a box gets the point for the box and has another turn.

The keys used are the vi keys: **k** for up, **j** for down, **h** for left, and **l** for right. To switch between even and odd rows, use one of the following keys: **u** (diagonal right up) , **y** (diagonal left up) , **b** (diagonal left down) , **n** (diagonal right down) ; ⟨**space**⟩ sets a new border, **CTRL-L** and **CTRL-R** redraw the screen, and **q** quits the game.

Support options are:

−a	Don't use the alternate character set.
−n *ngames*	*ngames* games will be played. (Especially useful in **−p** *cc* mode.)
−p ⟨*c*|*h*⟩⟨*c*|*h*⟩	Select which of the two players is a human or a computer. The first argument is the first player; **c** stands for computer and **h** for human.
−w	Wait for a character press between games.

xdim and *ydim* define the size of the board in the x and y dimensions.

SEE ALSO

Elwyn R. Berlekamp, *The Dots and Boxes Game: Sophisticated Child's Play*, *A K Peters*, 2000, http://www.akpeters.com/book.asp?bID=111.

AUTHORS

Christos Zoulas ⟨christos@NetBSD.org⟩

NAME

factor — factor a number

SYNOPSIS

factor [*number ...*]

DESCRIPTION

The **factor** utility will factor integers between −2147483648 and 2147483647 inclusive. When a number is factored, it is printed, followed by a “:”, and the list of factors on a single line. Factors are listed in ascending order, and are preceded by a space. If a factor divides a value more than once, it will be printed more than once.

When **factor** is invoked with one or more arguments, each argument will be factored.

When **factor** is invoked with no arguments, **factor** reads numbers, one per line, from standard input, until end of file or error. Leading white-space and empty lines are ignored. Numbers may be preceded by a single - or +. Numbers are terminated by a non-digit character (such as a newline). After a number is read, it is factored. Input lines must not be longer than 255 characters.

DIAGNOSTICS

Out of range or invalid input results in an appropriate error message being written to standard error.

BUGS

factor cannot handle the “10 most wanted” factor list.

NAME

fish — play “Go Fish”

SYNOPSIS

fish [**−p**]

DESCRIPTION

fish is the game “Go Fish”, a traditional children’s card game.

The computer deals the player and itself seven cards, and places the rest of the deck face-down (figuratively). The object of the game is to collect “books”, or all of the members of a single rank. For example, collecting four 2’s would give the player a “book of 2’s”.

The options are as follows:

−p Professional mode.

The computer makes a random decision as to who gets to start the game, and then the computer and player take turns asking each other for cards of a specified rank. If the asked player has any cards of the requested rank, they give them up to the asking player. A player must have at least one of the cards of the rank they request in their hand. When a player asks for a rank of which the other player has no cards, the asker is told to “Go Fish!”. Then, the asker draws a card from the non-dealt cards. If they draw the card they asked for, they continue their turn, asking for more ranks from the other player. Otherwise, the other player gets a turn.

When a player completes a book, either by getting cards from the other player or drawing from the deck, they set those cards aside and the rank is no longer in play.

The game ends when either player no longer has any cards in their hand. The player with the most books wins.

fish provides instructions as to what input it accepts.

BUGS

The computer cheats only rarely.

NAME

fortune — print a random, hopefully interesting, adage

SYNOPSIS

fortune [**−aefilosw**] [**−m** *pattern*] [[*N%*] *file/dir/all*]

DESCRIPTION

When **fortune** is run with no arguments it prints out a random epigram. Epigrams are divided into several categories, where each category is subdivided into those which are potentially offensive and those which are not. The options are as follows:

−a Choose from all lists of maxims, both offensive and not. (See the **−o** option for more information on offensive fortunes.)

−e Consider all fortune files to be of equal size (see discussion below on multiple files).

−f Print out the list of files which would be searched, but don't print a fortune.

−i Ignore case for **−m** patterns.

−l Long dictums only.

−m *pattern*
Print out all fortunes which match the regular expression *pattern*. See regex(3) for a description of patterns.

−o Choose only from potentially offensive aphorisms. **Please, please, please request a potentially offensive fortune if and only if you believe, deep down in your heart, that you are willing to be offended. (And that if you are, you'll just quit using −o rather than give us grief about it, okay?)**

> ... let us keep in mind the basic governing philosophy of The Brotherhood, as handsomely summarized in these words: we believe in healthy, hearty laughter -- at the expense of the whole human race, if needs be. Needs be.
> --H. Allen Smith, "Rude Jokes"

−s Short apothegms only.

−w Wait before termination for an amount of time calculated from the number of characters in the message. This is useful if it is executed as part of the logout procedure to guarantee that the message can be read before the screen is cleared.

The user may specify alternative sayings. You can specify a specific file, a directory which contains one or more files, or the special word *all*, which says to use all the standard databases. Any of these may be preceded by a percentage, which is a number *N* between 0 and 100 inclusive, followed by a '%' character. If it is, there will be an *N* percent probability that an adage will be picked from that file or directory. If the percentages do not sum to 100, and there are specifications without percentages, the remaining percent will apply to those files and/or directories, in which case the probability of selecting from one of them will be based on their relative sizes.

As an example, given two databases *funny* and *not-funny*, with *funny* twice as big, saying

```
$ fortune funny not-funny
```

will get you fortunes out of *funny* two-thirds of the time. The command

```
$ fortune 90% funny 10% not-funny
```

will pick out 90% of its fortunes from *funny* (the “10%” is unnecessary, since 10% is all that’s left). The **−e** option says to consider all files equal; thus

```
$ fortune -e
```

is equivalent to

```
$ fortune 50% funny 50% not-funny
```

Datafiles for **fortune** are created by the `strfile`(8) utility, which is not installed by default, The source code for this utility can be found in `/usr/src/games/fortune/strfile`, if it exists.

FILES

`/usr/share/games/fortune/*`	Fortune files.

SEE ALSO

`regex`(3), `random`(6), `rot13`(6)

NAME

gomoku — game of 5 in a row

SYNOPSIS

gomoku [**−bcdu**] [**−D** *debugfile*] [*inputfile*]

DESCRIPTION

gomoku is a two player game were the object is to get 5 in a row horizontally, vertically or diagonally on a 19 by 19 grid. By convention, black always moves first. With no arguments, **gomoku** will display a playing board and prompt for moves from the user. Valid moves are a letter for the column and a number for the row of an empty board location. Entering ‘‘quit" or ‘‘resign" will end the game. You can save the current state of the game by entering ‘‘save" and supplying a file name when prompted. The optional file *inputfile* can be used to restore a saved game.

The options are:

−b This option sets background mode. Input moves are read from standard input, the computer picks a move, and prints it to standard output. The first input line should be either ‘‘black" or ‘‘white" to specify whether **gomoku** has the first move or not respectively. This option was intended for game tournaments where a referee program handles the board display and pits one program against another.

−c Computer versus computer. **gomoku** will play a game against itself. This is mostly used for testing.

−d Print debugging information. Repeating this option more than once yields more detailed information.

−D *debugfile*
Print the debug information to *debugfile* instead of to the standard output.

−u User versus user. This is mostly used for testing.

AUTHORS

Ralph Campbell

ACKNOWLEDGEMENTS

The board display routines were based on the **goref** program written by Peter Langston.

NAME

hack — exploring The Dungeons of Doom

SYNOPSIS

hack [**−d** *directory*] [**−n**] [**−u** *playername*]
hack [**−d** *directory*] [**−s**] [**−X**] [*playername ...*]

DESCRIPTION

hack is a display oriented dungeons & dragons-like game. Both display and command structure resemble rogue. (For a game with the same structure but entirely different display - a real cave instead of dull rectangles - try Quest.)

To get started you really only need to know two commands. The command **?** will give you a list of the available commands and the command **/** will identify the things you see on the screen.

To win the game (as opposed to merely playing to beat other people's high scores) you must locate the Amulet of Yendor which is somewhere below the 20th level of the dungeon and get it out. Nobody has achieved this yet and if somebody does, he will probably go down in history as a hero among heroes.

When the game ends, either by your death, when you quit, or if you escape from the caves, **hack** will give you (a fragment of) the list of top scorers. The scoring is based on many aspects of your behavior but a rough estimate is obtained by taking the amount of gold you've found in the cave plus four times your (real) experience. Precious stones may be worth a lot of gold when brought to the exit. There is a 10% penalty for getting yourself killed.

The administration of the game is kept in the directory specified with the **−d** option, or, if no such option is given, in the directory specified by the environment variable `HACKDIR`, or, if no such variable exists, in the current directory. This same directory contains several auxiliary files such as lockfiles and the list of topscorers and a subdirectory `save` where games are saved. The game administrator may however choose to install hack with a fixed playing ground, usually `/var/games/hackdir`.

The **−n** option suppresses printing of the news.

The **−u** *playername* option supplies the answer to the question "Who are you?". When *playername* has as suffix one of *-T*, *-S*, *-K*, *-F*, *-C*, or *-W*, then this supplies the answer to the question "What kind of character ... ?".

The **−s** option will print out the list of your scores. It may be followed by arguments **−X** where X is one of the letters C, F, K, S, T, W to print the scores of Cavemen, Fighters, Knights, Speleologists, Tourists or Wizards. It may also be followed by one or more player names to print the scores of the players mentioned.

ENVIRONMENT

`USER` or `LOGNAME`	Your login name.
`HOME`	Your home directory.
`SHELL`	Your shell.
`TERM`	The type of your terminal.
`HACKPAGER, PAGER`	Pager used instead of default pager.
`MAIL`	Mailbox file.
`MAILREADER`	Reader used instead of default (probably `/usr/bin/mail`).
`HACKDIR`	Playground.
`HACKOPTIONS`	String predefining several hack options (see help file).

Several other environment variables are used in debugging (wizard) mode, like `GENOCIDED`, `INVENT`, `MAGIC` and `SHOPTYPE`.

FILES

`hack`	The hack program.
`data, rumors`	Data files used by hack.
`help, hh`	Help data files.
`record`	The list of topscorers.
`save`	A subdirectory containing the saved games.
`bones_dd`	Descriptions of the ghost and belongings of a deceased adventurer.
`xlock.dd`	Description of a dungeon level.
`safelock`	Lock file for xlock.
`record_lock`	Lock file for record.

AUTHORS

Jay Fenlason (+ Kenny Woodland, Mike Thome and Jon Payne) wrote the original hack, very much like rogue (but full of bugs).

Andries Brouwer continuously deformed their sources into the current version - in fact an entirely different game.

BUGS

Probably infinite. Mail complaints to mcvax!aeb .

NAME

hangman — computer version of the game hangman

SYNOPSIS

hangman [**-d** *wordlist*] [**-m** *minlen*]

DESCRIPTION

In **hangman**, the computer picks a word from the on-line word list and you must try to guess it. The computer keeps track of which letters have been guessed and how many wrong guesses you have made on the screen in a graphic fashion.

OPTIONS

-d	Use the specified *wordlist* instead of the default one named below.
-m	Set the minimum word length to use. The default is 6 letters.

FILES

`/usr/share/dict/words` On-line word list

AUTHORS

Ken Arnold

NAME

hunt — a multi-player multi-terminal game

SYNOPSIS

hunt [**−bcfmqSs**] [**−n** *name*] [**−t** *team*] [**−p** *port*] [**−w** *message*] [*host*]

DESCRIPTION

The object of the game **hunt** is to kill off the other players. There are no rooms, no treasures, and no monsters. Instead, you wander around a maze, find grenades, trip mines, and shoot down walls and players. The more players you kill before you die, the better your score is. If the **−m** flag is given, you enter the game as a monitor (you can see the action but you cannot play).

hunt normally looks for an active game on the local network; if none is found, it starts one up on the local host. The location of the game may be specified by giving the *host* argument. This presupposes that a hunt game is already running on that host, see huntd(6) for details on how to set up a game on a specific host. If more than one game if found, you may pick which game to play in.

If the **−q** flag is given, **hunt** queries the local network (or specific host) and reports on all active games found. This is useful for shell startup scripts, e.g., csh(1)'s .login.

The player name may be specified on the command line by using the **−n** option.

The **−c**, **−s**, and **−f** options are for entering the game cloaked, scanning, or flying respectively.

The **−b** option turns off beeping when you reach the typeahead limit.

The **−t** option aids team playing by making everyone else on one's team appear as the team name. A team name is a single digit to avoid conflicting with other characters used in the game.

The **−p** *port* option allows the rendezvous port number to be set. This is a useful way for people playing on dialup lines to avoid playing with people on 9600 baud terminals.

The **−w** *message* option is the only way to send a message to everyone else's screen when you start up. It is most often used to say "eat slime death - NickD's coming in".

When you die and are asked if you wish to re-enter the game, there are other answers than just yes or no. You can also reply with a **w** for write a message before continuing or **o** to change how you enter the game (cloaked, scanning, or flying).

To be notified automatically when a **hunt** starts up, add your login to the *hunt-players* mailing list (see huntd(6)).

PLAYING HINTS

hunt only works on CRT (vdt) terminals with at least 24 lines, 80 columns, and cursor addressing. The screen is divided in to 3 areas. On the right hand side is the status area. It shows damage sustained, charges remaining, who's in the game, who's scanning (the "*" in front of the name), who's cloaked (the "+" in front of the name), and other players' scores. The rest of the screen is taken up by your map of the maze. The 24th line is used for longer messages that don't fit in the status area.

hunt uses the same keys to move as vi(1) does, i.e., **h**, **j**, **k**, and **l** for left, down, up, right respectively. To change which direction you're facing in the maze, use the upper case version of the movement key (i.e., **HJKL**). You can only fire or throw things in the direction you're facing.
Other commands are:

f or **1**	Fire a bullet (Takes 1 charge)

g or **2**	Throw grenade (Takes 9 charges)
F or **3**	Throw satchel charge (Takes 25 charges)
G or **4**	Throw bomb (Takes 49 charges)
5	Throw big bomb (Takes 81 charges)
6	Throw even bigger bomb (Takes 121 charges)
7	Throw even more big bomb (Takes 169 charges)
8	Throw even more bigger bomb (Takes 225 charges)
9	Throw very big bomb (Takes 289 charges)
0	Throw very, very big bomb (Takes 361 charges)
@	Throw biggest bomb (Takes 441 charges)
o	Throw small slime (Takes 15 charges)
O	Throw big slime (Takes 30 charges)
p	Throw bigger slime (Takes 45 charges)
P	Throw biggest slime (Takes 60 charges)
s	Scan (show where other players are) (Takes 1 charge)
c	Cloak (hide from scanners) (Takes 1 charge)
^L	Redraw screen
q	Quit

The symbols on the screen are:

```
-|+     walls
/\      diagonal (deflecting) walls
#       doors (dispersion walls)
;       small mine
g       large mine
:       bullet
o       grenade
O       satchel charge
@       bomb
s       small slime
$       big slime
><^v    you facing right, left, up, or down
}{i!    other players facing right, left, up, or down
*       explosion
\|/
-*-     grenade and large mine explosion
/|\
```

Other helpful hints:

- You can only fire in the direction you are facing.
- You can only fire three shots in a row, then the gun must cool off.

- Shots move 5 times faster than you do.
- To stab someone, you face that player and move at them.
- Stabbing does 2 points worth of damage and shooting does 5 points.
- Slime does 5 points of damage each time it hits.
- You start with 15 charges and get 5 more every time a player enters or re-enters.
- Grenade explosions cover a 3 by 3 area, each larger bomb cover a correspondingly larger area (ranging from 5 by 5 to 21 by 21). All explosions are centered around the square the shot hits and do the most damage in the center.
- Slime affects all squares it oozes over. The number of squares is equal to the number of charges used.
- One small mine and one large mine is placed in the maze for every new player. A mine has a 2% probability of tripping when you walk forward on to it; 50% when going sideways; 95% when backing up. Tripping a mine costs you 5 points or 10 points respectively. Defusing a mine is worth 1 charge or 9 charges respectively.
- You cannot see behind you.
- Cloaking consumes 1 ammo charge per 20 of your moves.
- Scanning consumes 1 ammo charge per (20 × the number of players) of other player moves.
- Turning on cloaking turns off scanning — turning on scanning turns off cloaking.
- When you kill someone, you get 2 more damage capacity points and 2 damage points get taken away.
- Maximum typeahead is 5 characters.
- A shot destroys normal (i.e., non-diagonal, non-door) walls.
- Diagonal walls deflect shots and change orientation.
- Doors disperse shots in random directions (up, down, left, right).
- Diagonal walls and doors cannot be destroyed by direct shots but may be destroyed by an adjacent grenade explosion.
- Slime goes around walls, not through them.
- Walls regenerate, reappearing in the order they were destroyed. One percent of the regenerated walls will be diagonal walls or doors. When a wall is generated directly beneath a player, he is thrown in a random direction for a random period of time. When he lands, he sustains damage (up to 20 percent of the amount of damage already sustained); i.e., the less damage he had, the more nimble he is and therefore less likely to hurt himself on landing.
- Every 30 deaths or so, a “?” will appear. It is a wandering bomb which will explode when it hits someone, or when it is slimed.
- If no one moves, everything stands still.
- The environment variable `HUNT` is checked to get the player name. If you don’t have this variable set, **hunt** will ask you what name you want to play under. If you wish to set other options than just your name, you can enumerate the options as follows:

  ```
  setenv HUNT name=Sneaky,team=1,cloak,mapkey=zoFfGg1f2g3F4G
  ```

  sets the player name to Sneaky, sets the team to one, sets the enter game attribute to cloaked, and the maps **z** to **o**, **F** to **f**, **G** to **g**, **1** to **f**, **2** to **g**, **3** to **F**, and **4** to **G**. The *mapkey* option must be last. Other options are: *scan*, *fly*, *nobeep*, *port=string*, *host=string*, and *message=string*, which correspond to the command line options. String options cannot contain commas since commas are used to separate options.

- It's a boring game if you're the only one playing.

Your score is the decayed average of the ratio of number of kills to number of times you entered the game and is only kept for the duration of a single session of **hunt**.

hunt normally drives up the load average to be approximately (number_of_players + 0.5) greater than it would be without a **hunt** game executing.

STATISTICS

The **−S** option fetches the current game statistics. The meaning of the column headings are as follows:

score	the player's last score
ducked	how many shots a player ducked
absorb	how many shots a player absorbed
faced	how many shots were fired at player's face
shot	how many shots were fired at player
robbed	how many of player's shots were absorbed
missed	how many of player's shots were ducked
slimeK	how many slime kills player had
enemy	how many enemies were killed how many friends were killed (self and same team)
deaths	how many times player died
still	how many times player died without typing in any commands
saved	how many times a shot/bomb would have killed player if he hadn't ducked or absorbed it.

SEE ALSO

`huntd`(6)

AUTHORS

Conrad Huang, Ken Arnold, and Greg Couch;
University of California, San Francisco, Computer Graphics Lab

ACKNOWLEDGEMENTS

We thank Don Kneller, John Thomason, Eric Pettersen, Mark Day, and Scott Weiner for providing endless hours of play-testing to improve the character of the game. We hope their significant others will forgive them; we certainly don't.

BUGS

To keep up the pace, not everything is as realistic as possible.

NAME

huntd — hunt daemon, back-end for hunt game

SYNOPSIS

huntd [**−s**] [**−p** *port*]

DESCRIPTION

huntd controls the multi-player `hunt`(6) game. When it starts up, it tries to notify all members of the *hunt-players* mailing list (see `sendmail`(8)) by faking a `talk`(1) request from user “Hunt Game”.

The **−s** option is for running **huntd** forever (server mode). This is similar to running it under the control of `inetd`(8) (see below), but it consumes a process table entry when no one is playing.

The **−p** option changes the UDP port number used to rendezvous with the player process and thus allows for private games of hunt. This option turns off the notification of players on the *hunt-players* mailing list.

INETD

To run **huntd** from `inetd`(8), you’ll need to uncomment the following line in `/etc/inetd.conf`:

```
hunt dgram udp wait nobody /usr/games/huntd huntd
```

Do not use any of the command line options; if you want `inetd`(8) to start up **huntd** on a private port, change the port listed for **hunt** in `/etc/services`.

NETWORK RENDEZVOUS

When `hunt`(6) starts up, it broadcasts on the local area net (using the broadcast address for each interface) to find a **hunt** game in progress. If a **huntd** hears the request, it sends back the port number for the **hunt** process to connect to. Otherwise, the **hunt** process starts up a **huntd** on the local machine and tries to rendezvous with it.

SEE ALSO

`talk`(1), `hunt`(6), `sendmail`(8)

AUTHORS

Conrad Huang, Ken Arnold, and Greg Couch;
University of California, San Francisco, Computer Graphics Lab

NAME

larn — exploring the caverns of Larn

SYNOPSIS

larn [**-r**] [**-H** *number*] [**-n**] [**-h**] [**-o** *optsfile*]

DESCRIPTION

larn is a fantasy games in which your child has contracted a strange disease, and none of your home remedies seem to have any effect. You set out to find a remedy in a limited amount of time, and to collect gold along the way of course!

The options are:

-r The **-r** option restores a checkpointed game after it has died.

-H The **-H** option sets the hardness of the game.

-n The **-n** option suppresses the welcome message at start up, putting you directly into the game.

-h The **-h** option prints the command line options.

-o The **-o** option specifies a different options file than `~/.larnopts`.

COMMANDS

These are the movement commands:

h move to the left	H run left	. stay here
j move down	J run down	Z teleport yourself
k move up	K run up	c cast a spell
l move to the right	L run right	r read a scroll
y move northwest	Y run northwest	q quaff a potion
u move northeast	U run northeast	W wear armor
b move southwest	B run southwest	T take off armor
n move southeast	N run southeast	w wield a weapon
ˆ identify a trap	g give present pack weight	P give tax status
d drop an item	i inventory your pockets	Q quit the game
v print program version	S save the game	D list all items found
? this help screen	A create diagnostic file (wizards only)	e eat something

OPTIONS FILE

The file `~/.larnopts` may be used to set a few options for **larn**. A sequence of words terminated by whitespace is used to specify options.

Word	**Meaning**
bold-objects	Select bold display of objects.
inverse-objects	Select inverse video display of objects.
no-introduction	Do not display intro message.
enable-checkpointing	Turn on periodic checkpointing.
no-beep	Disable beeping of the terminal.
male	Choose your sex to be a man.
female	Choose your sex to be a woman.
name: “your name”	Choose your playing name.

monster: “monst name”	Choose a name for a monster.
savefile: “save-file-name”	Define what the savegame filename will be.

Your name and monster names must be enclosed in double quotation marks and may be up to 34 characters long. Longer names are truncated. Anything enclosed in quotation marks is considered one word, and must be separated from other words by whitespace.

SPECIAL NOTES

When **dropping gold**, if you type ’∗’ as your amount, all your gold gets dropped. In general, typing in ’∗’ means all of what you are interested in. This is true when visiting the bank, or when contributing at altars.

You can get out of the store, trading post, school, or home by hitting ⟨**esc**⟩.

When casting a spell, if you need a list of spells you can cast, type **D** as the first letter of your spell. The available list of spells will be shown, after which you may enter the spell code. This only works on the 1st letter of the spell you are casting.

FILES

`/var/games/larn.scores`	Score file.
`~/.larnopts`	Options file.

AUTHORS

Noah Morgan

NAME

mille − play Mille Bornes

SYNOPSIS

mille [file]

DESCRIPTION

Mille plays a two-handed game reminiscent of the Parker Brother's game of Mille Bornes with you. The rules are described below. If a file name is given on the command line, the game saved in that file is started.

When a game is started up, the bottom of the score window will contain a list of commands. They are:

P Pick a card from the deck. This card is placed in the 'P' slot in your hand.

D Discard a card from your hand. To indicate which card, type the number of the card in the hand (or "P" for the just-picked card) followed by a <RETURN> or <SPACE>. The <RETURN or <SPACE> is required to allow recovery from typos which can be very expensive, like discarding safeties.

U Use a card. The card is again indicated by its number, followed by a <RETURN> or <SPACE>.

O Toggle ordering the hand. By default off, if turned on it will sort the cards in your hand appropriately. This is not recommended for the impatient on slow terminals.

Q Quit the game. This will ask for confirmation, just to be sure. Hitting <DELETE> (or <RUBOUT>) is equivalent.

S Save the game in a file. If the game was started from a file, you will be given an opportunity to save it on the same file. If you don't wish to, or you did not start from a file, you will be asked for the file name. If you type a <RETURN> without a name, the save will be terminated and the game resumed.

R Redraw the screen from scratch. The command ˆL (control 'L') will also work.

W Toggle window type. This switches the score window between the startup window (with all the command names) and the end-of-game window. Using the end-of-game window saves time by eliminating the switch at the end of the game to show the final score. Recommended for hackers and other miscreants.

If you make a mistake, an error message will be printed on the last line of the score window, and a bell will beep.

At the end of each hand or game, you will be asked if you wish to play another. If not, it will ask you if you want to save the game. If you do, and the save is unsuccessful, play will be resumed as if you had said you wanted to play another hand/game. This allows you to use the "**S**" command to reattempt the save.

AUTHORS

Ken Arnold
(The game itself is a product of Parker Brothers, Inc.)

SEE ALSO

curses(3X), *Screen Updating and Cursor Movement Optimization: A Library Package*, Ken Arnold

CARDS

Here is some useful information. The number in parentheses after the card name is the number of that card in the deck:

Hazard	Repair	Safety
Out of Gas (2)	Gasoline (6)	Extra Tank (1)
Flat Tire (2)	Spare Tire (6)	Puncture Proof (1)
Accident (2)	Repairs (6)	Driving Ace (1)
Stop (4)	Go (14)	Right of Way (1)
Speed Limit (3)	End of Limit (6)	

25 − (10), 50 − (10), 75 − (10), 100 − (12), 200 − (4)

RULES

Object: The point of this game is to get a total of 5000 points in several hands. Each hand is a race to put down exactly 700 miles before your opponent does. Beyond the points gained by putting down milestones, there are several other ways of making points.

Overview: The game is played with a deck of 101 cards. *Distance* cards represent a number of miles traveled. They come in denominations of 25, 50, 75, 100, and 200. When one is played, it adds that many miles to the player's trip so far this hand. *Hazard* cards are used to prevent your opponent from putting down Distance cards. They can only be played if your opponent has a *Go* card on top of the Battle pile. The cards are *Out of Gas*, *Accident*, *Flat Tire*, *Speed Limit*, and *Stop*. *Remedy* cards fix problems caused by Hazard cards played on you by your opponent. The cards are *Gasoline*, *Repairs*, *Spare Tire*, *End of Limit*, and *Go*. *Safety* cards prevent your opponent from putting specific Hazard cards on you in the first place. They are *Extra Tank*, *Driving Ace*, *Puncture Proof*, and *Right of Way*, and there are only one of each in the deck.

Board Layout: The board is split into several areas. From top to bottom, they are: **SAFETY AREA** (unlabeled): This is where the safeties will be placed as they are played. **HAND**: These are the cards in your hand. **BATTLE**: This is the Battle pile. All the Hazard and Remedy Cards are played here, except the *Speed Limit* and *End of Limit* cards. Only the top card is displayed, as it is the only effective one. **SPEED**: The Speed pile. The *Speed Limit* and *End of Limit* cards are played here to control the speed at which the player is allowed to put down miles. **MILEAGE**: Miles are placed here. The total of the numbers shown here is the distance traveled so far.

Play: The first pick alternates between the two players. Each turn usually starts with a pick from the deck. The player then plays a card, or if this is not possible or desirable, discards one. Normally, a play or discard of a single card constitutes a turn. If the card played is a safety, however, the same player takes another turn immediately.

This repeats until one of the players reaches 700 points or the deck runs out. If someone reaches 700, they have the option of going for an *Extension*, which means that the play continues until someone reaches 1000 miles.

Hazard and Remedy Cards: Hazard Cards are played on your opponent's Battle and Speed piles. Remedy Cards are used for undoing the effects of your opponent's nastiness.

Go (Green Light) must be the top card on your Battle pile for you to play any mileage, unless you have played the *Right of Way* card (see below).
Stop is played on your opponent's *Go* card to prevent them from playing mileage until they play a *Go* card.
Speed Limit is played on your opponent's Speed pile. Until they play an *End of Limit* they can only play 25 or 50 mile cards, presuming their *Go* card allows them to do even that.
End of Limit is played on your Speed pile to nullify a *Speed Limit* played by your opponent.
Out of Gas is played on your opponent's *Go* card. They must then play a *Gasoline* card, and then a *Go* card before they can play any more mileage.
Flat Tire is played on your opponent's *Go* card. They must then play a *Spare Tire* card, and then a *Go* card before they can play any more mileage.

Accident is played on your opponent's *Go* card. They must then play a *Repairs* card, and then a *Go* card before they can play any more mileage.

Safety Cards: Safety cards prevent your opponent from playing the corresponding Hazard cards on you for the rest of the hand. It cancels an attack in progress, and *always entitles the player to an extra turn*.
Right of Way prevents your opponent from playing both *Stop* and *Speed Limit* cards on you. It also acts as a permanent *Go* card for the rest of the hand, so you can play mileage as long as there is not a Hazard card on top of your Battle pile. In this case only, your opponent can play Hazard cards directly on a Remedy card other than a Go card.
Extra Tank When played, your opponent cannot play an *Out of Gas* on your Battle Pile.
Puncture Proof When played, your opponent cannot play a *Flat Tire* on your Battle Pile.
Driving Ace When played, your opponent cannot play an *Accident* on your Battle Pile.

Distance Cards: Distance cards are played when you have a *Go* card on your Battle pile, or a Right of Way in your Safety area and are not stopped by a Hazard Card. They can be played in any combination that totals exactly 700 miles, except that *you cannot play more than two 200 mile cards in one hand*. A hand ends whenever one player gets exactly 700 miles or the deck runs out. In that case, play continues until neither someone reaches 700, or neither player can use any cards in their hand. If the trip is completed after the deck runs out, this is called *Delayed Action*.

Coup Fourré: This is a French fencing term for a counter-thrust move as part of a parry to an opponent's attack. In current French colloquial language it means a sneaky, underhanded blow. In Mille Bornes, it is used as follows: If an opponent plays a Hazard card, and you have the corresponding Safety in your hand, you play it immediately, even *before* you draw. This immediately removes the Hazard card from your Battle pile, and protects you from that card for the rest of the game. This gives you more points (see "Scoring" below).

Scoring: Scores are totaled at the end of each hand, whether or not anyone completed the trip. The terms used in the Score window have the following meanings:
Milestones Played: Each player scores as many miles as they played before the trip ended.
Each Safety: 100 points for each safety in the Safety area.
All 4 Safeties: 300 points if all four safeties are played.
Each Coup Fourré: 300 points for each Coup Fourré accomplished.

The following bonus scores can apply only to the winning player.
Trip Completed: 400 points bonus for completing the trip to 700 or 1000.
Safe Trip: 300 points bonus for completing the trip without using any 200 mile cards.
Delayed Action: 300 points bonus for finishing after the deck was exhausted.
Extension: 200 points bonus for completing a 1000 mile trip.
Shut-Out: 500 points bonus for completing the trip before your opponent played any mileage cards.

Running totals are also kept for the current score for each player for the hand (**Hand Total**), the game (**Overall Total**), and number of games won (**Games**).

NAME

monop — Monopoly game

SYNOPSIS

monop [*file*]

DESCRIPTION

monop is reminiscent of the Parker Brother's game Monopoly, and monitors a game between 2 to 9 users. It is assumed that the rules of Monopoly are known. The game follows the standard rules, with the exception that, if a property goes up for auction and there are only two solvent players, no auction is held and the property remains unowned.

The game, in effect, lends the player money, so it is possible to buy something which you cannot afford. However, as soon as a person goes into debt, he must "fix the problem", i.e., make himself solvent, before play can continue. If this is not possible, the player's property reverts to his debtee, either a player or the bank. A player can resign at any time to any person or the bank, which puts the property back on the board, unowned.

Any time that the response to a question is a *string*, e.g., a name, place or person, you can type '**?**' to get a list of valid answers. It is not possible to input a negative number, nor is it ever necessary.

A Summary of Commands:

quit quit game: This allows you to quit the game. It asks you if you're sure.

print
print board: This prints out the current board. The columns have the following meanings (column headings are the same for the **where**, **own holdings**, and **holdings** commands):

Name	The first ten characters of the name of the square.
Own	The *number* of the owner of the property.
Price	The cost of the property (if any).
Mg	This field has a '*' in it if the property is mortgaged.
#	If the property is a Utility or Railroad, this is the number of such owned by the owner. If the property is land, this is the number of houses on it.
Rent	Current rent on the property. If it is not owned, there is no rent.

where
where players are: Tells you where all the players are. A '*' indicates the current player.

own holdings
List your own holdings, i.e., money, get-out-of-jail-free cards, and property.

holdings
holdings list: Look at anyone's holdings. It will ask you whose holdings you wish to look at. When you are finished, type '**done**'.

mortgage
mortgage property: Sets up a list of mortgageable property, and asks which you wish to mortgage.

unmortgage
unmortgage property: Unmortgage mortgaged property.

buy buy houses: Sets up a list of monopolies on which you can buy houses. If there is more than one, it asks you which you want to buy for. It then asks you how many for each piece of property, giving the current amount in parentheses after the property name. If you build in an unbalanced manner (a disparity of more than one house within the same monopoly), it asks you to re-input things.

`sell` sell houses: Sets up a list of monopolies from which you can sell houses. It operates in an analogous manner to **`buy`**.

`card` card for jail: Use a get-out-of-jail-free card to get out of jail. If you're not in jail, or you don't have one, it tells you so.

`pay` pay for jail: Pay $50 to get out of jail, from whence you are put on Just Visiting. Difficult to do if you're not there.

`trade`
This allows you to trade with another player. It asks you whom you wish to trade with, and then asks you what each wishes to give up. You can get a summary at the end, and, in all cases, it asks for confirmation of the trade before doing it.

`resign`
Resign to another player or the bank. If you resign to the bank, all property reverts to its virgin state, and get-out-of-jail-free cards revert to the deck.

`save` save game: Save the current game in a file for later play. You can continue play after saving, either by adding the file in which you saved the game after the **`monop`** command, or by using the **`restore`** command (see below). It will ask you which file you wish to save it in, and, if the file exists, confirm that you wish to overwrite it.

`restore`
restore game: Read in a previously saved game from a file. It leaves the file intact.

`roll` Roll the dice and move forward to your new location. If you simply hit the ⟨RETURN⟩ key instead of a command, it is the same as typing **`roll`**.

AUTHORS

Ken Arnold

BUGS

No command can be given an argument instead of a response to a query.

NAME

number — convert Arabic numerals to English

SYNOPSIS

number [**-l**] [*number ...*]

DESCRIPTION

The **number** utility prints the English equivalent of the number to the standard output, with each 10ˆ3 magnitude displayed on a separate line. If no argument is specified, **number** reads lines from the standard input.

The options are as follows:

-l Display the number on a single line.

BUGS

Although **number** understand fractions, it doesn't understand exponents.

NAME

phantasia — an interterminal fantasy game

SYNOPSIS

phantasia [**−abHmpSsx**]

DESCRIPTION

phantasia is a role playing game which allows players to roll up characters of various types to fight monsters and other players. Progression of characters is based upon gaining experience from fighting monsters (and other players).

Most of the game is menu driven and self-explanatory (more or less). The screen is cursor updated, so be sure to set up the TERM variable in your environment.

The options provide for a variety of functions to support the game. They are:

−a Get a listing of all character names on file.

−b Show scoreboard of top characters per login.

−H Print header only.

−m Get a monster listing.

−p Purge old characters.

−S Turn on wizard options, if allowed, if running as “root”.

−s Invokes **phantasia** without header information.

−x Examine/change a particular character on file.

The characters are saved on a common file, in order to make the game interactive between players. The characters are given a password in order to retrieve them later. Only characters above *level* zero are saved. Characters unused for awhile will be purged. Characters are only placed on the scoreboard when they die.

PARTICULARS

Normal Play

A number of the player’s more important statistics are almost always displayed on the screen, with maximums (where applicable) in parentheses.

The character is placed randomly near the center of a Cartesian system. Most commands are selected with a single letter or digit. For example, one may move by hitting ’W’, ’S’, ’N’, or ’E’, (lower case may also be used, at no time is the game case dependent). One may also use ’H’, ’J’, ’K’, ’L’, for movement, similar to vi(1). To move to a specific (x, y) coordinate, use the **move** (’1’) command. The distance a character can move is calculated by 1 plus 1.5 per *level*. Moving in a compass direction will move the player the maximum allowed distance in that direction.

A player may see who else is playing by using the **players** (’2’) option. One may see the coordinates of those who are the same distance or closer to the origin as he/she. *Kings*, and *council of the wise* can see and can be seen by everyone. A *palantir* removes these restrictions.

One can talk to other players with the **talk** (’3’) option. In general, this is a line or so of text. To remove a current message, just type ⟨return⟩ when prompted for a message.

The **stats** (’4’) option shows additional characteristics of a player.

One may leave the game either with the **quit** (’5’) option.

One may rest by default. Resting lets one regain maximum *energy level*, and also lets one find *mana* (more is found for larger levels and further distances from the origin).

One may call a monster by hitting ’9’ or ’C’.

Use ’X’ to examine other players.

One may quit or execute a sub-shell by hitting interrupt. Quitting during battle results in death for obvious reasons.

Several other options become available as the player progresses in *level* and *magic*, or to other stations in the game (*valar*, *council of the wise*, *king*). These are described elsewhere. In general, a control-L will force the redrawing of the screen.

Other things which may happen are more or less self-explanatory.

Fighting Monsters

A player has several options while fighting monsters. They are as follows:

`melee`	Inflicts damage on the monster, based upon *strength*. Also decreases the monster’s *strength* some.
`skirmish`	Inflicts a little less damage than **`melee`**, but decreases the monster’s *quickness* instead.
`evade`	Attempt to run away. Success is based upon both the player’s and the monster’s *brains* and *quickness*.
`spell`	Several options for throwing spells (described elsewhere).
`nick`	Hits the monster one plus the player’s *sword*, and gives the player 10% of the monster’s *experience*. Decreases the monster’s *experience* an amount proportional to the amount granted. This also increases the monster’s quickness. Paralyzed monsters wake up very fast when nicked.
`luckout`	This is essentially a battle of wits with the monster. Success is based upon the player’s and the monster’s *brains*. The player gets credit for slaying the monster if he/she succeeds. Otherwise, nothing happens, and the chance to **`luckout`** is lost.

Character Statistics

strength	determines how much damage a character can inflict.
quickness	determines how many chances a character gets to make decisions while fighting.
energy level	specifies how much damage a character may endure before dying.
magic level	determines which spells a character may throw, and how effective those spells will be.
brains	basically, the character’s intelligence; used for various fighting options and spells.
mana	used as a power source for throwing spells.
experience	gained by fighting monsters and other characters.
level	indicative of how much experience a character has accumulated; progresses geometrically as *experience* increases.
poison	sickness which degrades a character’s performance (affects *energy level* and *strength*).
sin	accumulated as a character does certain nasty things; used only rarely in normal play of the game.

age	of player; roughly equivalent to number of turns. As *age* increases, many personal statistics degenerate.

Character Types

Character statistics are rolled randomly from the above list, according to character type. The types are as follows:

magic user	strong in *magic level* and *brains*, weak in other areas. Must rely on wits and magic to survive.
fighter	good in *strength* and *energy level*, fairly good in other areas. This adds up to a well-equipped fighter.
elf	very high *quickness* and above average *magic level* are *elves* selling points.
dwarf	very high *strength* and *energy level*, but with a tendency to be rather slow and not too bright.
halfling	rather quick and smart, with high *energy level*, but poor in *magic* and *strength*. Born with some *experience*.
experimento	very mediocre in all areas. However, the *experimento* may be placed almost anywhere within the playing grid.

The possible ranges for starting statistics are summarized in the following table.

Type	Strength	Quick	Mana	Energy	Brains	Magic
Mag. User	10-15	30-35	50-100	30-45	60-85	5-9
Fighter	40-55	30-35	30-50	45-70	25-45	3-6
Elf	35-45	32-38	45-90	30-50	40-65	4-7
Dwarf	50-70	25-30	25-45	60-100	20-40	2-5
Halfling	20-25	34	25-45	55-90	40-75	1-4
Experimento	25	27	100	35	25	2

Not only are the starting characteristics different for the different character types, the characteristics progress at different rates for the different types as the character goes up in *level*. *Experimentoes*' characteristics progress randomly as one of the other types. The progression as characters increase in *level* is summarized in the following table.

Type	Strength	Mana	Energy	Brains	Magic
Mag. User	2.0	75	20	6	2.75
Fighter	3.0	40	30	3.0	1.5
Elf	2.5	65	25	4.0	2.0
Dwarf	5	30	35	2.5	1
Halfling	2.0	30	30	4.5	1

The character type also determines how much gold a player may carry, how long until *rings* can overcome the player, and how much *poison* the player can withstand.

Spells

During the course of the game, the player may exercise his/her magic powers. These cases are described below.

`cloak`

magic level necessary: 20 (plus level 7)
mana used: 35 plus 3 per rest period
Used during normal play. Prevents monsters from finding the character, as well as hiding the player from other players. His/her coordinates show up as '?' in the **`players`** option. Players cannot collect *mana*, find trading posts, or dis-

cover the *grail* while cloaked. Calling a monster uncloaks, as well as choosing this option while cloaked.

`teleport`
magic level necessary: 40 (plus level 12)
mana used: 30 per 75 moved
Used during normal play. Allows the player to move with much more freedom than with the **`move`** option, at the price of expending mana. The maximum distance possible to move is based upon *level* and *magic level*.

`power blast`
magic level necessary: none
mana used: 5 times *level*
Used during inter-terminal battle. Damage is based upon *magic level* and *strength*. Hits much harder than a normal hit.

`all or nothing`
magic level necessary: none
mana used: 1
Used while combating monsters. Has a 25% chance of working. If it works it hits the monster just enough to kill it. If it fails, it doesn't hit the monster, and doubles the monster's *quickness* and *strength*. Paralyzed monsters wake up much quicker as a result of this spell.

`magic bolt`
magic level necessary: 5
mana used: variable
Used while combating monsters. Hits the monster based upon the amount of *mana* expended and *magic level*. Guaranteed to hit at least 10 per *mana*.

`force field`
magic level necessary: 15
mana used: 30
Used during monster combat. Throws up a shield to protect from damage. The shield is added to actual energy level, and is a fixed number, based upon maximum energy. Normally, damage occurs first to the shield, and then to the players actual *energy level*.

`transform`
magic level necessary: 25
mana used: 50
Used during monster combat. Transforms the monster randomly into one of the 100 monsters from the monster file.

`increase might`
magic level necessary: 35
mana used: 75
Used during combat with monsters. Increases strength up to a maximum.

`invisibility`
magic level necessary: 45
mana used: 90
Used while fighting monsters. Makes it harder for the monster to hit, by temporarily increasing the player's *quickness*. This spell may be thrown several times, but a maximum level will be reached.

`transport`
magic level necessary: 60
mana used: 125
Used during monster combat. Transports the monster away from the player. Success is based upon player's *magic* and *brains*, and the monster's *experience*. If it fails the player is transported instead. 60% of the time, the monster will drop any treasure it was carrying.

`paralyze`	*magic level necessary*: 75 *mana used*: 150 Used during monster combat. “Freezes” the monster by putting its *quickness* slightly negative. The monster will slowly wake up. Success is based upon player’s *magic* and the monster’s *experience*. If it fails, nothing happens.
`specify`	*magic level necessary*: none *mana used*: 1000 Used during monster combat only by *valar* or *council of the wise*. Allows the player to pick which monster to fight.

Monsters

Monsters get bigger as one moves farther from the origin (0,0). Rings of distance 125 from the origin determine the size. A monster’s *experience*, *energy level*, and *brains* are multiplied by the size. *Strength* is increased 50% per size over one, and *quickness* remains the same, regardless of size.

Also, nastier monsters are found as one progress farther out from the origin. Monsters also may flock. The percent chance of that happening is designated as *flock%* in the monster listing. Monsters outside the first ring may carry treasure, as determined by their treasure type. Flocking monsters, and bigger monsters increase the chances of treasure.

Certain monsters have special abilities; they are as follows:

Unicorn	can only be subdued if the player is in possession of a *virgin*.
Modnar	has random characteristics, including treasure type.
Mimic	will pick another name from the list of monsters in order to confuse.
Dark Lord	very nasty person. Does not like to be hit (especially nicked), and many spells do not work well (or at all) against him. One can always *evade* from the *Dark Lord*.
Leanan-Sidhe	also a very nasty person. She will permanently sap *strength* from someone.
Saruman	wanders around with *Wormtongue*, who can steal a *palantir*. Also, *Saruman* may turn a player’s gems into gold pieces, or scramble her/his stats.
Thaumaturgist	can transport a player.
Balrog	inflicts damage by taking away *experience*, not *energy*.
Vortex	may take some *mana*.
Nazgul	may try to steal a *ring* or neutralize part of one’s *brains*.
Tiamat	may take half a player’s *gold* and *gems* and escape.
Kobold	may get nasty and steal one gold piece and run away.
Shelob	may bite, inflicting the equivalent of one *poison*.
Assorted Faeries	These are killed if attacking someone carrying *holy water*. These are *Cluricaun*, *Fir Darrig*, *Fachan*, *Ghille Dhu*, *Bogle*, *Killmoulis*, and *Bwca*.
Lamprey	may bite, inflicting 1/2 of a *poison*.
Shrieker	will call one of its (much bigger) buddies if picked upon.
Bonnacon	will become bored with battle, fart, and run off.

Smeagol	will try to steal a *ring* from a player, if given the chance.
Succubus	may inflict damage through a **`force field`**. This subtracts from *energy level* instead of any shield the player may have thrown up. This is a very easy way to die.
Cerberus	loves metal and will steal all the metal treasures from a player if able.
Ungoliant	can bite and poison. This inflicts five *poisons*, and also takes one from the player's *quickness*.
Jabberwock	may tire of battle, and leave after calling one of his friends (*Jubjub Bird* or *Bandersnatch*).
Morgoth	actually *Modnar*, but reserved for *council of the wise*, *valar*, and *ex-valar*. Fights with *Morgoth* end when either he or the player dies. His characteristics are calculated based upon the player's. The player is given the chance to ally with him. No magic, except **`force field`** works when battling *Morgoth*.
Troll	may regenerate its *energy* and *strength* while in battle.
Wraith	may make a player blind.

Treasures

The various treasure types are as follows:

Type zero	none
Type one	*power booster* − adds mana. *druid* − adds experience. *holy orb* − subtracts 0.25 sin.
Type two	*amulet* − protects from cursed treasure. *holy water* − kills *assorted faeries*. *hermit* − reduces sin by 25% and adds some mana.
Type three	*shield* − adds to maximum *energy level*. *virgin* − used to subdue a *unicorn*, or to give much *experience* (and some *sin*). *athelas* − subtracts one *poison*.
Type four (scrolls)	*shield* − throws a bigger than normal **`force field`**. *invisible* − temporarily puts the finder's *quickness* to one million. *ten fold strength* − multiplies finder's strength by ten. *pick monster* − allows finder to pick next monster to battle. *general knowledge* − adds to finder's *brains* and *magic level*. All the scrolls except *general knowledge* automatically call a monster. These preserve any spells that were already in effect, but are only in effect while in battle.
Type five	*dagger* − adds to *strength*. *armour* − same as a *shield*, but bigger. *tablet* − adds *brains*.
Type six	*priest* − rests to maximum; adds *mana*, *brains*; and halves *sin*. *Robin Hood* − increases *shield* and adds permanently to *strength*. *axe* − like *dagger*, but bigger.
Type seven	*charm* − protects from cursed treasure (used before *amulet*); used in conjunction with *blessing* to battle *Dark Lord*. *Merlyn* − adds *brains*, *magic*, and *mana*.

	war hammer − like an *axe*, but bigger.
Type eight	*healing potion* − sets *poison* to -2, or subtracts two from *poison*, whichever is better. *transporter* − allows finder to move anywhere. *sword* − like a *war hammer*, but bigger.
Type nine	*golden crown* − allows the player to become *king*, by going to (0,0). *blessing* − cuts *sin* to 1/3, adds *mana*, rests to maximum, kills *Dark Lord* with a *charm*, and gives bearer first hit on all monsters. *quicksilver* − adds to *quickness*.
Type ten	*elven boots* − adds permanently to *quickness*.
Type eleven	*palantir* − allows one to see all the other players; used by *council of the wise* to seek the *grail*.
Type twelve/thirteen	*ring* − allows one to hit much harder in battle, etc.

Any treasure type 10-13 monsters may instead carry a type nine treasure.

A monster may also be carrying *gold* or *gems*. These are used at *trading posts* to buy things. A *gem* is worth 1000 gold pieces. Too much *gold* will slow a player down. One may carry 1000 plus 200 per *level* of *gold*. A *gem* weighs one half a gold piece. Monsters of treasure type 7 or higher may carry *gems*.

The chance of a cursed treasure is based upon treasure type. The more valuable treasures have a greater chance of being cursed. A cursed treasure knocks *energy level* very low, and adds 0.25 *poison*.

Rings

Rings are only carried by *nazguls* and *Dark Lords*. They come in four different flavors. All *rings* rest the player to maximum and cause him/her to hit much harder in battle with monsters (assuming one has chosen to use the *ring* for battle.)

Two types of *rings* are cursed and come either from *nazguls* or *Dark Lord*. After a few times of using these types, the player falls under the control of the *ring*, and strange, random things will occur. Eventually, the player dies, and gives his/her name to a monster on the file. Dying before the *ring* is used up also renames the monster.

The two remaining types of *rings* are much more benign. The one from a *nazgul* is good for a limited number of battle rounds, and will save the player from death if it was being used when he/she died. The one from *Dark Lord* is the same, except that it never is used up. *rings* disappear after saving someone from death. In general, cursed *rings* occur much more often than normal ones. It is usually not a good idea to pick one up. The only way to get rid of a *ring* is to have a monster steal it.

King

A player may become *king* by finding a *crown* and going to (0,0). Players must have a *level* in the range of 10 to 1000 to be able to find a *crown*. When a player with one or more *crowns* reaches *level* 1000, the *crowns* are converted to *gold*.

Once a player is king, he/she may do certain things while in the Lord's Chamber (0,0). These are exercised with the **`decree`** ('0') option.

`transport`	This is done to another player. It randomly moves the affected player about. A *charm* protects from transports.
`curse`	This is done to another player. It is analogous to cursed treasure, but worse. It inflicts two *poison*, knocks *energy level* very low, and degrades the maximum energy. It also removes a *cloak*. A *blessing* protects from king's curses.

`energy void`	The king may put a number of these scattered about his/her kingdom as he/she pleases. If a player hits one, he/she loses *mana*, *energy*, and *gold*. The energy void disappears after being hit.
`bestow`	This is also done to another player. The king may wish to reward one or more loyal subjects by sharing his/her riches (*gold*). Or it is a convenient way to dispose of some unwanted deadweight.
`collect taxes`	Everyone pays 7% tax on all *gold* and *gems* acquired, regardless of the existence of a *king*. The king collects the accrued taxes with this option.

The *king* may also **`teleport`** anywhere for free by using the origin as a starting place.

Council of the Wise, Valar

A player automatically becomes a member of the *council of the wise* upon reaching level 3000. Members of the council cannot have *rings*. Members of the council have a few extra options which they can exercise. These are exercised with the **`intervene`** ('8') option. All **`intervene`** options cost 1000 mana. One **`intervene`** option is to **`heal`** another player. This is just a quick way for that player to be rested to maximum and lose a little *poison*. The main purpose in life for members of the council is to seek the *Holy Grail*. This is done with a *palantir* under the **`seek grail`** option. The distance cited by the seek is accurate within 10%, in order not to make it too easy to find the grail. A player must have infinitesimally small *sin*, or else it's all over upon finding the grail. In order to help members of the council on their quest, they may **`teleport`** with greater ease.

Upon finding the grail, the player advances to position of *valar*. He/she may then exercise more and niftier options under **`intervention`**. These include all of the council members' options plus the ability to move other players about, bless them, and throw monsters at them. A *valar*'s blessing has the same effect as the treasure *blessing*, except that the affected player does not get his/her *blessing* flag set. All **`intervention`** options which affect other players age the player who uses them. *Valars* are essentially immortal, but are actually given five lives. If these are used up, the player is left to die, and becomes an *ex-valar*. A *valar* cannot **`move`**, **`teleport`**, or call monsters. (An exception to this is if the *valar* finds a *transporter*.) This is to allow him/her to dispose of excess *gold*. Any monsters which a *valar* encounters are based upon his/her size. Only one valar may exist at a time. The current valar is replaced when another player finds the grail. The valar is then bumped back to the council of the wise.

Wizard

The *wizard* is usually the owner of the game, and the one who maintains the associated files. The *wizard* is granted special powers within the game, if it is invoked with the **`−S`** option. Otherwise, the *wizard* plays no different from other players. The *wizard* abilities are outlined below.

`change players`	When examining a player, (game invoked with **`−x`**, or use 'X' from within game), the *wizard* may also change the player.
`intervention`	The *wizard* may do all the **`intervention`** options. One extra option, **`vaporize`**, is added to kill any offensive players.
`super character type`	An extra character type is added. This character starts with the maximum possible in all statistics, selected from the other character types. A *super* character's statistics also progress at the maximum possible rate, selected from the other character types.

Special Places

Certain regions of the playing grid have different names. In general, this is only to give the player some idea of his/her present location. Some special places do exist.

Trading Posts	These are located at |x| == |y| == n∗n∗100 for n = 1, 2, ..., 1000. Trading posts farther out have more things for sale. Be careful about cheating the merchants there, as they have short tempers. Merchants are dishonest about 5% of the time.
Lord's Chamber	This is located at (0,0). Only players with *crowns* may enter.
Point of No Return	This is located beyond 1.2e+6 in any direction. The only way to return from here is a *transporter* or to have a *valar* relocate the player.
Dead Marshes	This is a band located fairly distant from the origin. The first fourteen monsters (water monsters) can normally only be found here.
Valhala	This place is where the *valar* resides. It is associated with no particular coordinate on the playing grid.

Miscellaneous

Once a player reaches *level* 5, the game will start to time out waiting for input. This is to try to keep the game a bit faster paced.

A *guru* will never be disgusted with your *sins* if they are less than one.

A *medic* wants half of a player's *gold* to be happy. Offering more than one has, or a negative amount will anger the *medic*, who will make the player worse (add one *poison*).

The *Holy Grail* does little for those who are not ready to behold it. Whenever anyone finds it, it moves. It is always located within 1e+6 in any compass direction of the origin.

There is a maximum amount of *mana* and *charms* a player may posses, based upon *level*. *Quicksilver* is always limited to to a maximum of 99.

Books bought at a *trading post* increase *brains*, based upon the number bought. It is unwise, however to buy more than 1/10 of one's *level* in books at a time.

Players over level 10000 are automatically retired.

A *blindness* goes away in random time.

Players with *crowns* are identified with a '∗' before their character type.

Inter-terminal Battle

When two player's coordinates correspond, they may engage in battle. In general, the player with the highest *quickness* gets the first hit. If the two players are severely mismatched, the stronger player is drastically handicapped for the battle. In order to protect from being stuck in an infinite loop, the player waiting for response may time out. Options for battle are:

`fight`	Inflicts damage upon other person.
`run away`	Escape from battle. Has a 75% chance of working.
`power blast`	Battle spell.
`luckout`	One-time chance to try to win against the foe. Has a 10% chance of working.

Sometimes waits for the other player may be excessive, because he/she may be battling a monster. Upon slaying a player in battle the winner gets the other's *experience* and treasures. *Rings* do not work for inter-terminal battle.

AUTHORS

Edward Estes, AT&T Information Systems, Skokie, IL

BUGS

All screen formats assume at least 24 lines by at least 80 columns. No provisions are made for when any of the data items get too big for the allotted space on the screen.

NAME

pig — eformatray inputway asway Igpay Atinlay

SYNOPSIS

pig

DESCRIPTION

Ethay **pig** utilityway eadsray ethay andardstay inputway andway iteswray itway outway otay andardstay outputway inway Igpay Atinlay.

Usefulway orfay eneratinggay onthlymay eportsray.

NAME

pom — display the phase of the moon

SYNOPSIS

pom [[[[[cc]yy]mm]dd]HH]]

DESCRIPTION

The **pom** utility displays the current phase of the moon. Useful for selecting software completion target dates and predicting managerial behavior.

[[[[[cc]yy]mm]dd]HH] Display the phase of the moon for a given time. The format is similar to the canonical representation used by `date`(1).

SEE ALSO

`date`(1)

AUTHORS

pom was written by Keith E. Brandt.

BUGS

Times must be within range of the UNIX epoch.

This program does not allow for the difference between the TDT and UTC timescales (about one minute at the time of writing).

ACKNOWLEDGEMENTS

This program is based on algorithms from *Practical Astronomy with Your Calculator, Third Edition* by Peter Duffett-Smith ⟨pjds@mrao.cam.ac.uk⟩.

NAME

primes — generate primes

SYNOPSIS

primes [**−d**] [*start* [*stop*]]

DESCRIPTION

The **primes** utility prints primes in ascending order, one per line, starting at or above *start* and continuing until, but not including *stop*. The *start* value must be at least 0 and not greater than *stop*. The *stop* value must not be greater than 4294967295. The default value of *stop* is 4294967295.

When the **primes** utility is invoked with no arguments, *start* is read from standard input. *stop* is taken to be 4294967295. The *start* value may be preceded by a single '+'. The *start* value is terminated by a non-digit character (such as a newline). The input line must not be longer than 255 characters. When given the **−d** argument, **primes** prints the difference between the current and the previous prime.

DIAGNOSTICS

Out of range or invalid input results in an appropriate error message being written to standard error.

BUGS

primes won't get you a world record.

NAME

quiz — random knowledge tests

SYNOPSIS

quiz [**−t**] [**−i** *file*] [*question answer*]

DESCRIPTION

The **quiz** utility tests your knowledge of random facts. It has a database of subjects from which you can choose. With no arguments, **quiz** displays the list of available subjects.

The options are as follows:

−t Use tutorial mode, in which questions are repeated later if you didn't get them right the first time, and new questions are presented less frequently to help you learn the older ones.

−i Specify an alternative index file.

Subjects are divided into categories. You can pick any two categories from the same subject. **quiz** will ask questions from the first category and it expects answers from the second category. For example, the command "quiz victim killer" asks questions which are the names of victims, and expects you to answer with the cause of their untimely demise, whereas the command "quiz killer victim" works the other way around.

If you get the answer wrong, **quiz** lets you try again. To see the right answer, enter a blank line.

Index and Data File Syntax

The index and data files have a similar syntax. Lines in them consist of several categories separated by colons. The categories are regular expressions formed using the following meta-characters:

pat|pat	alternative patterns
{pat}	optional pattern
[pat]	delimiters, as in pat[pat|pat]pat

In an index file, each line represents a subject. The first category in each subject is the pathname of the data file for the subject. The remaining categories are regular expressions for the titles of each category in the subject.

In data files, each line represents a question/answer set. Each category is the information for the question/answer for that category.

The backslash character ("\") is used to quote syntactically significant characters, or at the end of a line to signify that a continuation line follows.

If either a question or its answer is empty, **quiz** will refrain from asking it.

FILES

`/usr/share/games/quiz.db` The default index and data files.

BUGS

quiz is pretty cynical about certain subjects.

NAME

rain — animated raindrops display

SYNOPSIS

rain [**−d** *delay*]

DESCRIPTION

The output of **rain** is modeled after the VAX/VMS program of the same name. To obtain the proper effect, either the terminal must be set for 9600 baud or the **−d** option must be used to specify a delay, in milliseconds, between each update. A reasonable delay is 120; the default is 0.

AUTHORS

Eric P. Scott

NAME

random — random lines from a file or random numbers

SYNOPSIS

random [**−er**] [*denominator*]

DESCRIPTION

random reads lines from the standard input and copies them to the standard output with a probability of 1/denominator. The default value for *denominator* is 2.

The options are as follows:

−e If the **−e** option is specified, **random** does not read or write anything, and simply exits with a random exit value of 0 to *denominator*-1, inclusive.

−r The **−r** option guarantees that the output is unbuffered.

SEE ALSO

shuffle(1), fortune(6)

NAME

robots — fight off villainous robots

SYNOPSIS

robots [**−Asjtan**] [*scorefile*]

DESCRIPTION

robots pits you against evil robots, who are trying to kill you (which is why they are evil). Fortunately for you, even though they are evil, they are not very bright and have a habit of bumping into each other, thus destroying themselves. In order to survive, you must get them to kill each other off, since you have no offensive weaponry.

Since you are stuck without offensive weaponry, you are endowed with one piece of defensive weaponry: a teleportation device. When two robots run into each other or a junk pile, they die. If a robot runs into you, you die. When a robot dies, you get 10 points, and when all the robots die, you start on the next field. This keeps up until they finally get you.

Robots are represented on the screen by a '+', the junk heaps from their collisions by a '∗', and you (the good guy) by a '@'.

The commands are:

h	move one square left
l	move one square right
k	move one square up
j	move one square down
y	move one square up and left
u	move one square up and right
b	move one square down and left
n	move one square down and right
.	(also space) do nothing for one turn
HJKLBNYU	run as far as possible in the given direction
>	do nothing for as long as possible
t	teleport to a random location
w	wait until you die or they all do
q	quit
^L	redraw the screen

All commands can be preceded by a count.

If you use the '**w**' command and survive to the next level, you will get a bonus of 10% for each robot which died after you decided to wait. If you die, however, you get nothing. For all other commands, the program will save you from typos by stopping short of being eaten. However, with '**w**' you take the risk of dying by miscalculation.

Only five scores are allowed per user on the score file. If you make it into the score file, you will be shown the list at the end of the game. If an alternative score file is specified, that will be used instead of the standard file for scores.

The options are

−s Don't play, just show the score file.

−j Jump, *i.e.*, when you run, don't show any intermediate positions; only show things at the end. This is useful on slow terminals.

-t Teleport automatically when you have no other option. This is a little disconcerting until you get used to it, and then it is very nice.

-a Advance into the higher levels directly, skipping the lower, easier levels.

-A Auto-bot mode. Lets the game play itself.

-n Increase the number of games played by one.

FILES

`/var/games/robots_roll` the score file

AUTHORS

Ken Arnold
Christos Zoulas (autobot mode)

BUGS

Bugs? You *crazy*, man?!?

NAME

rogue — exploring The Dungeons of Doom

SYNOPSIS

rogue [**-s**] [*save_file*]

DESCRIPTION

rogue is a computer fantasy game with a new twist. It is CRT oriented and the object of the game is to survive the attacks of various monsters and get a lot of gold, rather than the puzzle solving orientation of most computer fantasy games.

To get started you really only need to know two commands. The command **?** will give you a list of the available commands and the command **/** will identify the things you see on the screen.

To win the game (as opposed to merely playing to beat other people's high scores) you must locate the Amulet of Yendor which is somewhere below the 20th level of the dungeon and get it out. Nobody has achieved this yet and if somebody does, they will probably go down in history as a hero among heroes.

When the game ends, either by your death, when you quit, or if you (by some miracle) manage to win, **rogue** will give you a list of the top-ten scorers. The scoring is based entirely upon how much gold you get. There is a 10% penalty for getting yourself killed.

If *save_file* is specified, rogue will be restored from the specified saved game file.

The **-s** option will print out the list of scores.

For more detailed directions, read the document *A Guide to the Dungeons of Doom*.

FILES

/var/games/rogue.scores	Score file
~/rogue.save	Default save file

SEE ALSO

Michael C. Toy and Kenneth C. R. C. Arnold, *A guide to the Dungeons of Doom*.

AUTHORS

Timothy Stoehr
Michael C. Toy
Kenneth C. R. C. Arnold
Glenn Wichman

BUGS

Probably infinite, although none are known. However, that Ice Monsters sometimes transfix you permanently is *not* a bug. It's a feature.

NAME

sail − multi-user wooden ships and iron men

SYNOPSIS

sail [**−s** [**−l**]] [**−x**] [**−b**] [**num**]

DESCRIPTION

Sail is a computer version of Avalon Hill's game of fighting sail originally developed by S. Craig Taylor.

Players of *Sail* take command of an old fashioned Man of War and fight other players or the computer. They may re-enact one of the many historical sea battles recorded in the game, or they can choose a fictional battle.

As a sea captain in the *Sail* Navy, the player has complete control over the workings of his ship. He must order every maneuver, change the set of his sails, and judge the right moment to let loose the terrible destruction of his broadsides. In addition to fighting the enemy, he must harness the powers of the wind and sea to make them work for him. The outcome of many battles during the age of sail was decided by the ability of one captain to hold the 'weather gage.'

The flags are:

−s Print the names and ships of the top ten sailors.

−l Show the login name. Only effective with **-s**.

−x Play the first available ship instead of prompting for a choice.

−b No bells.

IMPLEMENTATION

Sail is really two programs in one. Each player starts up a process which runs his own ship. In addition, a *driver* process is forked (by the first player) to run the computer ships and take care of global bookkeeping.

Because the *driver* must calculate moves for each ship it controls, the more ships the computer is playing, the slower the game will appear.

If a player joins a game in progress, he will synchronize with the other players (a rather slow process for everyone), and then he may play along with the rest.

To implement a multi-user game in Version 7 UNIX, which was the operating system *Sail* was first written under, the communicating processes must use a common temporary file as a place to read and write messages. In addition, a locking mechanism must be provided to ensure exclusive access to the shared file. For example, *Sail* uses a temporary file named /tmp/#sailsink.21 for scenario 21, and corresponding file names for the other scenarios. To provide exclusive access to the temporary file, *Sail* uses a technique stolen from an old game called "pubcaves" by Jeff Cohen. Processes do a busy wait in the loop

```
for (n = 0; link(sync_file, sync_lock) < 0 && n < 30; n++)
        sleep(2);
```

until they are able to create a link to a file named "/tmp/#saillock.??". The "??" correspond to the scenario number of the game. Since UNIX guarantees that a link will point to only one file, the process that succeeds in linking will have exclusive access to the temporary file.

Whether or not this really works is open to speculation. When ucbmiro was rebooted after a crash, the file system check program found 3 links between the *Sail* temporary file and its link file.

CONSEQUENCES OF SEPARATE PLAYER AND DRIVER PROCESSES

When players do something of global interest, such as moving or firing, the driver must coordinate the action with the other ships in the game. For example, if a player wants to move in a certain direction, he writes a message into the temporary file requesting the driver to move his ship. Each ‘‘turn,’’ the driver reads all the messages sent from the players and decides what happened. It then writes back into the temporary file new values of variables, etc.

The most noticeable effect this communication has on the game is the delay in moving. Suppose a player types a move for his ship and hits return. What happens then? The player process saves up messages to be written to the temporary file in a buffer. Every 7 seconds or so, the player process gets exclusive access to the temporary file and writes out its buffer to the file. The driver, running asynchronously, must read in the movement command, process it, and write out the results. This takes two exclusive accesses to the temporary file. Finally, when the player process gets around to doing another 7 second update, the results of the move are displayed on the screen. Hence, every movement requires four exclusive accesses to the temporary file (anywhere from 7 to 21 seconds depending upon asynchrony) before the player sees the results of his moves.

In practice, the delays are not as annoying as they would appear. There is room for "pipelining" in the movement. After the player writes out a first movement message, a second movement command can then be issued. The first message will be in the temporary file waiting for the driver, and the second will be in the file buffer waiting to be written to the file. Thus, by always typing moves a turn ahead of the time, the player can sail around quite quickly.

If the player types several movement commands between two 7 second updates, only the last movement command typed will be seen by the driver. Movement commands within the same update "overwrite" each other, in a sense.

THE HISTORY OF SAIL

I wrote the first version of *Sail* on a PDP−11/70 in the fall of 1980. Needless to say, the code was horrendous, not portable in any sense of the word, and didn't work. The program was not very modular and had fseeks() and fwrites() every few lines. After a tremendous rewrite from the top down, I got the first working version up by 1981. There were several annoying bugs concerning firing broadsides and finding angles. *Sail* uses no floating point, by the way, so the direction routines are rather tricky. Ed Wang rewrote my angle() routine in 1981 to be more correct (although it still doesn't work perfectly), and he added code to let a player select which ship he wanted at the start of the game (instead of the first one available).

Captain Happy (Craig Leres) is responsible for making *Sail* portable for the first time. This was no easy task, by the way. Constants like 2 and 10 were very frequent in the code. I also became famous for using "Riggle Memorial Structures" in *Sail.* Many of my structure references are so long that they run off the line printer page. Here is an example, if you promise not to laugh.

```
specs[scene[flog.fgamenum].ship[flog.fshipnum].shipnum].pts
```

Sail received its fourth and most thorough rewrite in the summer and fall of 1983. Ed Wang rewrote and modularized the code (a monumental feat) almost from scratch. Although he introduced many new bugs, the final result was very much cleaner and (?) faster. He added window movement commands and find ship commands.

HISTORICAL INFO

Old Square Riggers were very maneuverable ships capable of intricate sailing. Their only disadvantage was an inability to sail very close to the wind. The design of a wooden ship allowed only for the guns to bear to the left and right sides. A few guns of small aspect (usually 6 or 9 pounders) could point forward, but their effect was small compared to a 68 gun broadside of 24 or 32 pounders. The guns bear approximately like so:

```
     \
      b----------------
   ---0
      \
       \
        \    up to a range of ten (for round shot)
         \
          \
           \
```

An interesting phenomenon occurred when a broadside was fired down the length of an enemy ship. The shot tended to bounce along the deck and did several times more damage. This phenomenon was called a rake. Because the bows of a ship are very strong and present a smaller target than the stern, a stern rake (firing from the stern to the bow) causes more damage than a bow rake.

```
 b
00  ----  Stern rake!
 a
```

Most ships were equipped with carronades, which were very large, close range cannons. American ships from the revolution until the War of 1812 were almost entirely armed with carronades.

The period of history covered in *Sail* is approximately from the 1770's until the end of Napoleonic France in 1815. There are many excellent books about the age of sail. My favorite author is Captain Frederick Marryat. More contemporary authors include C.S. Forester and Alexander Kent.

Fighting ships came in several sizes classed by armament. The mainstays of any fleet were its "Ships of the Line", or "Line of Battle Ships". They were so named because these ships fought together in great lines. They were close enough for mutual support, yet every ship could fire both its broadsides. We get the modern words "ocean liner," or "liner," and "battleship" from "ship of the line." The most common size was the 74 gun two decked ship of the line. The two gun decks usually mounted 18 and 24 pounder guns.

The pride of the fleet were the first rates. These were huge three decked ships of the line mounting 80 to 136 guns. The guns in the three tiers were usually 18, 24, and 32 pounders in that order from top to bottom.

Various other ships came next. They were almost all "razees," or ships of the line with one deck sawed off. They mounted 40-64 guns and were a poor cross between a frigate and a line of battle ship. They neither had the speed of the former nor the firepower of the latter.

Next came the "eyes of the fleet." Frigates came in many sizes mounting anywhere from 32 to 44 guns. They were very handy vessels. They could outsail anything bigger and outshoot anything smaller. Frigates didn't fight in lines of battle as the much bigger 74's did. Instead, they harassed the enemy's rear or captured crippled ships. They were much more useful in missions away from the fleet, such as cutting out expeditions or boat actions. They could hit hard and get away fast.

Lastly, there were the corvettes, sloops, and brigs. These were smaller ships mounting typically fewer than 20 guns. A corvette was only slightly smaller than a frigate, so one might have up to 30 guns. Sloops were used for carrying dispatches or passengers. Brigs were something you built for land-locked lakes.

SAIL PARTICULARS

Ships in *Sail* are represented by two characters. One character represents the bow of the ship, and the other represents the stern. Ships have nationalities and numbers. The first ship of a nationality is number 0, the second number 1, etc. Therefore, the first British ship in a game would be printed as "b0". The second Brit would be "b1", and the fifth Don would be "s4".

Ships can set normal sails, called Battle Sails, or bend on extra canvas called Full Sails. A ship under full sail is a beautiful sight indeed, and it can move much faster than a ship under Battle Sails. The only trouble is, with full sails set, there is so much tension on sail and rigging that a well aimed round shot can burst a sail into ribbons where it would only cause a little hole in a loose sail. For this reason, rigging damage is doubled on a ship with full sails set. Don't let that discourage you from using full sails. I like to keep them up right into the heat of battle. A ship with full sails set has a capital letter for its nationality. E.g., a Frog, "f0", with full sails set would be printed as "F0".

When a ship is battered into a listing hulk, the last man aboard "strikes the colors." This ceremony is the ship's formal surrender. The nationality character of a surrendered ship is printed as "!". E.g., the Frog of our last example would soon be "!0".

A ship has a random chance of catching fire or sinking when it reaches the stage of listing hulk. A sinking ship has a "~" printed for its nationality, and a ship on fire and about to explode has a "#" printed.

Captured ships become the nationality of the prize crew. Therefore, if an American ship captures a British ship, the British ship will have an "a" printed for its nationality. In addition, the ship number is changed to "&","’", "(", ,")", "*", or "+" depending upon the original number, be it 0,1,2,3,4, or 5. E.g., the "b0" captured by an American becomes the "a&". The "s4" captured by a Frog becomes the "f*".

The ultimate example is, of course, an exploding Brit captured by an American: "#&".

MOVEMENT

Movement is the most confusing part of *Sail* to many. Ships can head in 8 directions:

```
                  0     0     0
b     b     b0    b     b     b     0b    b
0     0                             0
```

The stern of a ship moves when it turns. The bow remains stationary. Ships can always turn, regardless of the wind (unless they are becalmed). All ships drift when they lose headway. If a ship doesn’t move forward at all for two turns, it will begin to drift. If a ship has begun to drift, then it must move forward before it turns, if it plans to do more than make a right or left turn, which is always possible.

Movement commands to *Sail* are a string of forward moves and turns. An example is "l3". It will turn a ship left and then move it ahead 3 spaces. In the drawing above, the "b0" made 7 successive left turns. When *Sail* prompts you for a move, it prints three characters of import. E.g.,

```
move (7, 4):
```

The first number is the maximum number of moves you can make, including turns. The second number is the maximum number of turns you can make. Between the numbers is sometimes printed a quote "’". If the quote is present, it means that your ship has been drifting, and you must move ahead to regain headway before you turn (see note above). Some of the possible moves for the example above are as follows:

```
move (7, 4): 7
move (7, 4): 1
move (7, 4): d            /* drift, or do nothing */
move (7, 4): 6r
move (7, 4): 5r1
move (7, 4): 4r1r
move (7, 4): l1r1r2
move (7, 4): 1r1r1r1
```

Because square riggers performed so poorly sailing into the wind, if at any point in a movement command you turn into the wind, the movement stops there. E.g.,

```
move (7, 4): l1l4
Movement Error;
Helm: l1l
```

Moreover, whenever you make a turn, your movement allowance drops to min(what’s left, what you would have at the new attitude). In short, if you turn closer to the wind, you most likely won’t be able to sail the full allowance printed in the "move" prompt.

Old sailing captains had to keep an eye constantly on the wind. Captains in *Sail* are no different. A ship’s ability to move depends on its attitude to the wind. The best angle possible is to have the wind off your quarter, that is, just off the stern. The direction rose on the side of the screen gives the possible movements for your ship at all positions to the wind. Battle sail speeds are given first, and full sail speeds are given in parenthesis.

```
 0 1(2)
\|/
-^-3(6)
```

```
/|\
 | 4(7)
3(6)
```

Pretend the bow of your ship (the "^") is pointing upward and the wind is blowing from the bottom to the top of the page. The numbers at the bottom "3(6)" will be your speed under battle or full sails in such a situation. If the wind is off your quarter, then you can move "4(7)". If the wind is off your beam, "3(6)". If the wind is off your bow, then you can only move "1(2)". Facing into the wind, you can't move at all. Ships facing into the wind were said to be "in irons".

WINDSPEED AND DIRECTION

The windspeed and direction is displayed as a little weather vane on the side of the screen. The number in the middle of the vane indicates the wind speed, and the + to - indicates the wind direction. The wind blows from the + sign (high pressure) to the - sign (low pressure). E.g.,

```
|
3
+
```

The wind speeds are 0 = becalmed, 1 = light breeze, 2 = moderate breeze, 3 = fresh breeze, 4 = strong breeze, 5 = gale, 6 = full gale, 7 = hurricane. If a hurricane shows up, all ships are destroyed.

GRAPPLING AND FOULING

If two ships collide, they run the risk of becoming tangled together. This is called "fouling." Fouled ships are stuck together, and neither can move. They can unfoul each other if they want to. Boarding parties can only be sent across to ships when the antagonists are either fouled or grappled.

Ships can grapple each other by throwing grapnels into the rigging of the other.

The number of fouls and grapples you have are displayed on the upper right of the screen.

BOARDING

Boarding was a very costly venture in terms of human life. Boarding parties may be formed in *Sail* to either board an enemy ship or to defend your own ship against attack. Men organized as Defensive Boarding Parties fight twice as hard to save their ship as men left unorganized.

The boarding strength of a crew depends upon its quality and upon the number of men sent.

CREW QUALITY

The British seaman was world renowned for his sailing abilities. American sailors, however, were actually the best seamen in the world. Because the American Navy offered twice the wages of the Royal Navy, British seamen who liked the sea defected to America by the thousands.

In *Sail,* crew quality is quantized into 5 energy levels. "Elite" crews can outshoot and outfight all other sailors. "Crack" crews are next. "Mundane" crews are average, and "Green" and "Mutinous" crews are below average. A good rule of thumb is that "Crack" or "Elite" crews get one extra hit per broadside compared to "Mundane" crews. Don't expect too much from "Green" crews.

BROADSIDES

Your two broadsides may be loaded with four kinds of shot: grape, chain, round, and double. You have guns and carronades in both the port and starboard batteries. Carronades only have a range of two, so you have to get in close to be able to fire them. You have the choice of firing at the hull or rigging of another ship. If the range of the ship is greater than 6, then you may only shoot at the rigging.

The types of shot and their advantages are:

ROUND

Range of 10. Good for hull or rigging hits.

DOUBLE

Range of 1. Extra good for hull or rigging hits. Double takes two turns to load.

CHAIN

Range of 3. Excellent for tearing down rigging. Cannot damage hull or guns, though.

GRAPE

Range of 1. Sometimes devastating against enemy crews.

On the side of the screen is displayed some vital information about your ship:

```
Load  D! R!
Hull  9
Crew  4  4  2
Guns  4  4
Carr  2  2
Rigg  5 5 5 5
```

"Load" shows what your port (left) and starboard (right) broadsides are loaded with. A "!" after the type of shot indicates that it is an initial broadside. Initial broadside were loaded with care before battle and before the decks ran red with blood. As a consequence, initial broadsides are a little more effective than broadsides loaded later. A "*" after the type of shot indicates that the gun crews are still loading it, and you cannot fire yet. "Hull" shows how much hull you have left. "Crew" shows your three sections of crew. As your crew dies off, your ability to fire decreases. "Guns" and "Carr" show your port and starboard guns. As you lose guns, your ability to fire decreases. "Rigg" shows how much rigging you have on your 3 or 4 masts. As rigging is shot away, you lose mobility.

EFFECTIVENESS OF FIRE

It is very dramatic when a ship fires its thunderous broadsides, but the mere opportunity to fire them does not guarantee any hits. Many factors influence the destructive force of a broadside. First of all, and the chief factor, is distance. It is harder to hit a ship at range ten than it is to hit one sloshing alongside. Next is raking. Raking fire, as mentioned before, can sometimes dismast a ship at range ten. Next, crew size and quality affects the damage done by a broadside. The number of guns firing also bears on the point, so to speak. Lastly, weather affects the accuracy of a broadside. If the seas are high (5 or 6), then the lower gunports of ships of the line can't even be opened to run out the guns. This gives frigates and other flush decked vessels an advantage in a storm. The scenario *Pellew vs. The Droits de L'Homme* takes advantage of this peculiar circumstance.

REPAIRS

Repairs may be made to your Hull, Guns, and Rigging at the slow rate of two points per three turns. The message "Repairs Completed" will be printed if no more repairs can be made.

PECULIARITIES OF COMPUTER SHIPS

Computer ships in *Sail* follow all the rules above with a few exceptions. Computer ships never repair damage. If they did, the players could never beat them. They play well enough as it is. As a consolation, the computer ships can fire double shot every turn. That fluke is a good reason to keep your distance. The *Driver* figures out the moves of the computer ships. It computes them with a typical A.I. distance function and a depth first search to find the maximum "score." It seems to work fairly well, although I'll be the first to admit it isn't perfect.

HOW TO PLAY

Commands are given to *Sail* by typing a single character. You will then be prompted for further input. A brief summary of the commands follows.

COMMAND SUMMARY

'f' Fire broadsides if they bear
'l' Reload
'L' Unload broadsides (to change ammo)

'm'	Move
'i'	Print the closest ship
'I'	Print all ships
'F'	Find a particular ship or ships (e.g. "a?" for all Americans)
's'	Send a message around the fleet
'b'	Attempt to board an enemy ship
'B'	Recall boarding parties
'c'	Change set of sail
'r'	Repair
'u'	Attempt to unfoul
'g'	Grapple/ungrapple
'v'	Print version number of game
'^L'	Redraw screen
'Q'	Quit

'C'	Center your ship in the window
'U'	Move window up
'D','N'	Move window down
'H'	Move window left
'J'	Move window right
'S'	Toggle window to follow your ship or stay where it is

SCENARIOS

Here is a summary of the scenarios in *Sail:*

Ranger vs. Drake:

Wind from the N, blowing a fresh breeze.

(a) Ranger	19 gun Sloop (crack crew) (7 pts)
(b) Drake	17 gun Sloop (crack crew) (6 pts)

The Battle of Flamborough Head:

Wind from the S, blowing a fresh breeze.

This is John Paul Jones' first famous battle. Aboard the Bonhomme Richard, he was able to overcome the Serapis's greater firepower by quickly boarding her.

(a) Bonhomme Rich	42 gun Corvette (crack crew) (11 pts)
(b) Serapis	44 gun Frigate (crack crew) (12 pts)

Arbuthnot and Des Touches:

Wind from the N, blowing a gale.

(b) America	64 gun Ship of the Line (crack crew) (20 pts)
(b) Befford	74 gun Ship of the Line (crack crew) (26 pts)
(b) Adamant	50 gun Ship of the Line (crack crew) (17 pts)
(b) London	98 gun 3 Decker SOL (crack crew) (28 pts)
(b) Royal Oak	74 gun Ship of the Line (crack crew) (26 pts)
(f) Neptune	74 gun Ship of the Line (average crew) (24 pts)
(f) Duc de Bourgogne	80 gun 3 Decker SOL (average crew) (27 pts)
(f) Conquerant	74 gun Ship of the Line (average crew) (24 pts)
(f) Provence	64 gun Ship of the Line (average crew) (18 pts)
(f) Romulus	44 gun Ship of the Line (average crew) (10 pts)

Suffren and Hughes:

Wind from the S, blowing a fresh breeze.

(b) Monmouth	74 gun Ship of the Line (average crew) (24 pts)
(b) Hero	74 gun Ship of the Line (crack crew) (26 pts)
(b) Isis	50 gun Ship of the Line (crack crew) (17 pts)
(b) Superb	74 gun Ship of the Line (crack crew) (27 pts)
(b) Burford	74 gun Ship of the Line (average crew) (24 pts)
(f) Flamband	50 gun Ship of the Line (average crew) (14 pts)
(f) Annibal	74 gun Ship of the Line (average crew) (24 pts)
(f) Severe	64 gun Ship of the Line (average crew) (18 pts)
(f) Brilliant	80 gun Ship of the Line (crack crew) (31 pts)
(f) Sphinx	80 gun Ship of the Line (average crew) (27 pts)

Nymphe vs. Cleopatre:

Wind from the S, blowing a fresh breeze.

(b) Nymphe	36 gun Frigate (crack crew) (11 pts)
(f) Cleopatre	36 gun Frigate (average crew) (10 pts)

Mars vs. Hercule:

Wind from the S, blowing a fresh breeze.

(b) Mars	74 gun Ship of the Line (crack crew) (26 pts)
(f) Hercule	74 gun Ship of the Line (average crew) (23 pts)

Ambuscade vs. Baionnaise:

Wind from the N, blowing a fresh breeze.

(b) Ambuscade	32 gun Frigate (average crew) (9 pts)
(f) Baionnaise	24 gun Corvette (average crew) (9 pts)

Constellation vs. Insurgent:

Wind from the S, blowing a gale.

(a) Constellation	38 gun Corvette (elite crew) (17 pts)
(f) Insurgent	36 gun Corvette (average crew) (11 pts)

Constellation vs. Vengeance:

Wind from the S, blowing a fresh breeze.

(a) Constellation	38 gun Corvette (elite crew) (17 pts)
(f) Vengeance	40 gun Frigate (average crew) (15 pts)

The Battle of Lissa:

Wind from the S, blowing a fresh breeze.

(b) Amphion	32 gun Frigate (elite crew) (13 pts)
(b) Active	38 gun Frigate (elite crew) (18 pts)
(b) Volage	22 gun Frigate (elite crew) (11 pts)
(b) Cerberus	32 gun Frigate (elite crew) (13 pts)
(f) Favorite	40 gun Frigate (average crew) (15 pts)
(f) Flore	40 gun Frigate (average crew) (15 pts)
(f) Danae	40 gun Frigate (crack crew) (17 pts)
(f) Bellona	32 gun Frigate (green crew) (9 pts)
(f) Corona	40 gun Frigate (green crew) (12 pts)
(f) Carolina	32 gun Frigate (green crew) (7 pts)

Constitution vs. Guerriere:

Wind from the SW, blowing a gale.

(a) Constitution	44 gun Corvette (elite crew) (24 pts)
(b) Guerriere	38 gun Frigate (crack crew) (15 pts)

United States vs. Macedonian:

Wind from the S, blowing a fresh breeze.

(a) United States	44 gun Frigate (elite crew) (24 pts)
(b) Macedonian	38 gun Frigate (crack crew) (16 pts)

Constitution vs. Java:

Wind from the S, blowing a fresh breeze.

(a) Constitution	44 gun Corvette (elite crew) (24 pts)
(b) Java	38 gun Corvette (crack crew) (19 pts)

Chesapeake vs. Shannon:

Wind from the S, blowing a fresh breeze.

(a) Chesapeake	38 gun Frigate (average crew) (14 pts)
(b) Shannon	38 gun Frigate (elite crew) (17 pts)

The Battle of Lake Erie:

Wind from the S, blowing a light breeze.

(a) Lawrence	20 gun Sloop (crack crew) (9 pts)
(a) Niagara	20 gun Sloop (elite crew) (12 pts)
(b) Lady Prevost	13 gun Brig (crack crew) (5 pts)
(b) Detroit	19 gun Sloop (crack crew) (7 pts)
(b) Q. Charlotte	17 gun Sloop (crack crew) (6 pts)

Wasp vs. Reindeer:

Wind from the S, blowing a light breeze.

(a) Wasp	20 gun Sloop (elite crew) (12 pts)
(b) Reindeer	18 gun Sloop (elite crew) (9 pts)

Constitution vs. Cyane and Levant:

Wind from the S, blowing a moderate breeze.

(a) Constitution 44 gun Corvette (elite crew) (24 pts) (b) Cyane 24 gun Sloop (crack crew) (11 pts) (b) Levant 20 gun Sloop (crack crew) (10 pts)

Pellew vs. Droits de L'Homme:

Wind from the N, blowing a gale.

(b) Indefatigable	44 gun Frigate (elite crew) (14 pts)
(b) Amazon	36 gun Frigate (crack crew) (14 pts)
(f) Droits L'Hom	74 gun Ship of the Line (average crew) (24 pts)

Algeciras:

Wind from the SW, blowing a moderate breeze.

(b) Caesar	80 gun Ship of the Line (crack crew) (31 pts)
(b) Pompee	74 gun Ship of the Line (crack crew) (27 pts)
(b) Spencer	74 gun Ship of the Line (crack crew) (26 pts)

(b) Hannibal	98 gun 3 Decker SOL (crack crew) (28 pts)
(s) Real-Carlos	112 gun 3 Decker SOL (green crew) (27 pts)
(s) San Fernando	96 gun 3 Decker SOL (green crew) (24 pts)
(s) Argonauta	80 gun Ship of the Line (green crew) (23 pts)
(s) San Augustine	74 gun Ship of the Line (green crew) (20 pts)
(f) Indomptable	80 gun Ship of the Line (average crew) (27 pts)
(f) Desaix	74 gun Ship of the Line (average crew) (24 pts)

Lake Champlain:

Wind from the N, blowing a fresh breeze.

(a) Saratoga	26 gun Sloop (crack crew) (12 pts)
(a) Eagle	20 gun Sloop (crack crew) (11 pts)
(a) Ticonderoga	17 gun Sloop (crack crew) (9 pts)
(a) Preble	7 gun Brig (crack crew) (4 pts)
(b) Confiance	37 gun Frigate (crack crew) (14 pts)
(b) Linnet	16 gun Sloop (elite crew) (10 pts)
(b) Chubb	11 gun Brig (crack crew) (5 pts)

Last Voyage of the USS President:

Wind from the N, blowing a fresh breeze.

(a) President	44 gun Frigate (elite crew) (24 pts)
(b) Endymion	40 gun Frigate (crack crew) (17 pts)
(b) Pomone	44 gun Frigate (crack crew) (20 pts)
(b) Tenedos	38 gun Frigate (crack crew) (15 pts)

Hornblower and the Natividad:

Wind from the E, blowing a gale.

A scenario for you Horny fans. Remember, he sank the Natividad against heavy odds and winds. Hint: don’t try to board the Natividad, her crew is much bigger, albeit green.

(b) Lydia	36 gun Frigate (elite crew) (13 pts)
(s) Natividad	50 gun Ship of the Line (green crew) (14 pts)

Curse of the Flying Dutchman:

Wind from the S, blowing a fresh breeze.

Just for fun, take the Piece of cake.

(s) Piece of Cake	24 gun Corvette (average crew) (9 pts)
(f) Flying Dutchy	120 gun 3 Decker SOL (elite crew) (43 pts)

The South Pacific:

Wind from the S, blowing a strong breeze.

(a) USS Scurvy	136 gun 3 Decker SOL (mutinous crew) (27 pts)
(b) HMS Tahiti	120 gun 3 Decker SOL (elite crew) (43 pts)
(s) Australian	32 gun Frigate (average crew) (9 pts)
(f) Bikini Atoll	7 gun Brig (crack crew) (4 pts)

Hornblower and the battle of Rosas bay:

Wind from the E, blowing a fresh breeze.

The only battle Hornblower ever lost. He was able to dismast one ship and stern rake the others though. See if you can do as well.

(b) Sutherland	74 gun Ship of the Line (crack crew) (26 pts)
(f) Turenne	80 gun 3 Decker SOL (average crew) (27 pts)
(f) Nightmare	74 gun Ship of the Line (average crew) (24 pts)
(f) Paris	112 gun 3 Decker SOL (green crew) (27 pts)
(f) Napoleon	74 gun Ship of the Line (green crew) (20 pts)

Cape Horn:

Wind from the NE, blowing a strong breeze.

(a) Concord	80 gun Ship of the Line (average crew) (27 pts)
(a) Berkeley	98 gun 3 Decker SOL (crack crew) (28 pts)
(b) Thames	120 gun 3 Decker SOL (elite crew) (43 pts)
(s) Madrid	112 gun 3 Decker SOL (green crew) (27 pts)
(f) Musket	80 gun 3 Decker SOL (average crew) (27 pts)

New Orleans:

Wind from the SE, blowing a fresh breeze.

Watch that little Cypress go!

(a) Alligator	120 gun 3 Decker SOL (elite crew) (43 pts)
(b) Firefly	74 gun Ship of the Line (crack crew) (27 pts)
(b) Cypress	44 gun Frigate (elite crew) (14 pts)

Botany Bay:

Wind from the N, blowing a fresh breeze.

(b) Shark	64 gun Ship of the Line (average crew) (18 pts)
(f) Coral Snake	44 gun Corvette (elite crew) (24 pts)
(f) Sea Lion	44 gun Frigate (elite crew) (24 pts)

Voyage to the Bottom of the Sea:

Wind from the NW, blowing a fresh breeze.

This one is dedicated to Richard Basehart and David Hedison.

(a) Seaview	120 gun 3 Decker SOL (elite crew) (43 pts)
(a) Flying Sub	40 gun Frigate (crack crew) (17 pts)
(b) Mermaid	136 gun 3 Decker SOL (mutinous crew) (27 pts)
(s) Giant Squid	112 gun 3 Decker SOL (green crew) (27 pts)

Frigate Action:

Wind from the E, blowing a fresh breeze.

(a) Killdeer	40 gun Frigate (average crew) (15 pts)
(b) Sandpiper	40 gun Frigate (average crew) (15 pts)
(s) Curlew	38 gun Frigate (crack crew) (16 pts)

The Battle of Midway:

Wind from the E, blowing a moderate breeze.

(a) Enterprise	80 gun Ship of the Line (crack crew) (31 pts)
(a) Yorktown	80 gun Ship of the Line (average crew) (27 pts)
(a) Hornet	74 gun Ship of the Line (average crew) (24 pts)
(j) Akagi	112 gun 3 Decker SOL (green crew) (27 pts)
(j) Kaga	96 gun 3 Decker SOL (green crew) (24 pts)
(j) Soryu	80 gun Ship of the Line (green crew) (23 pts)

Star Trek:

Wind from the S, blowing a fresh breeze.

(a) Enterprise	450 gun Ship of the Line (elite crew) (75 pts)
(a) Yorktown	450 gun Ship of the Line (elite crew) (75 pts)
(a) Reliant	450 gun Ship of the Line (elite crew) (75 pts)
(a) Galileo	450 gun Ship of the Line (elite crew) (75 pts)
(k) Kobayashi Maru	450 gun Ship of the Line (elite crew) (75 pts)
(k) Klingon II	450 gun Ship of the Line (elite crew) (75 pts)
(o) Red Orion	450 gun Ship of the Line (elite crew) (75 pts)
(o) Blue Orion	450 gun Ship of the Line (elite crew) (75 pts)

CONCLUSION

Sail has been a group effort.

AUTHORS

Dave Riggle

CO-AUTHOR

Ed Wang

REFITTING

Craig Leres

CONSULTANTS

Chris Guthrie
Captain Happy
Horatio Nelson
and many valiant others...

REFERENCES

Wooden Ships & Iron Men, by Avalon Hill
Captain Horatio Hornblower Novels, (13 of them) by C.S. Forester
Captain Richard Bolitho Novels, (12 of them) by Alexander Kent
The Complete Works of Captain Frederick Marryat, (about 20) especially
Mr. Midshipman Easy
Peter Simple
Jacob Faithful
Japhet in Search of a Father
Snarleyyow, or The Dog Fiend
Frank Mildmay, or The Naval Officer

BUGS

Probably a few, and please report them to "riggle@ernie.berkeley.edu" and "edward@ucbarpa.berkeley.edu"

NAME

snake, **snscore** — display chase game

SYNOPSIS

snake [**-w** *width*] [**-l** *length*] [**-t**]
snscore

DESCRIPTION

snake is a display-based game which must be played on a CRT terminal. The object of the game is to make as much money as possible without getting eaten by the snake. The **-l** and **-w** options allow you to specify the length and width of the field. By default the entire screen is used. The **-t** option makes the game assume you are on a slow terminal.

You are represented on the screen by an I. The snake is 6 squares long and is represented by s's with an S at its head. The money is $, and an exit is #. Your score is posted in the upper left hand corner.

You can move around using the same conventions as `vi`(1), the **h**, **j**, **k**, and **l** keys work, as do the arrow keys. Other possibilities include:

sefc	These keys are like hjkl but form a directed pad around the d key.
HJKL	These keys move you all the way in the indicated direction to the same row or column as the money. This does *not* let you jump away from the snake, but rather saves you from having to type a key repeatedly. The snake still gets all his turns.
SEFC	Likewise for the upper case versions on the left.
ATPB	These keys move you to the four edges of the screen. Their position on the keyboard is the mnemonic, e.g. P is at the far right of the keyboard.
x	This lets you quit the game at any time.
p	Points in a direction you might want to go.
w	Space warp to get out of tight squeezes, at a price.

To earn money, move to the same square the money is on. A new $ will appear when you earn the current one. As you get richer, the snake gets hungrier. To leave the game, move to the exit (#).

A record is kept of the personal best score of each player. Scores are only counted if you leave at the exit, getting eaten by the snake is worth nothing.

As in pinball, matching the last digit of your score to the number which appears after the game is worth a bonus.

To see who wastes time playing snake, run **snscore**.

FILES

`/var/games/snakerawscores`	database of personal bests
`/var/games/snake.log`	log of games played

BUGS

When playing on a small screen, it's hard to tell when you hit the edge of the screen.

The scoring function takes into account the size of the screen. A perfect function to do this equitably has not been devised.

NAME

tetris — the game of tetris

SYNOPSIS

tetris [**-ps**] [**-k** *keys*] [**-l** *level*]

DESCRIPTION

The **tetris** command runs display-based game which must be played on a CRT terminal. The object is to fit the shapes together forming complete rows, which then vanish. When the shapes fill up to the top, the game ends. You can optionally select a level of play, or custom-select control keys.

The default level of play is 2.

The default control keys are as follows:

j	move left
k	rotate 1/4 turn counterclockwise
l	move right
⟨space⟩	drop
p	pause
q	quit

The options are as follows:

-k The default control keys can be changed using the **-k** option. The *keys* argument must have the six keys in order, and, remember to quote any space or tab characters from the shell. For example:

```
tetris -l 2 -k 'jkl pq'
```

will play the default games, i.e. level 2 and with the default control keys. The current key settings are displayed at the bottom of the screen during play.

-l Select a level of play.

-s Display the top scores.

-p Switch on previewing of the shape that will appear next.

PLAY

At the start of the game, a shape will appear at the top of the screen, falling one square at a time. The speed at which it falls is determined directly by the level: if you select level 2, the blocks will fall twice per second; at level 9, they fall 9 times per second. (As the game goes on, things speed up, no matter what your initial selection.) When this shape “touches down” on the bottom of the field, another will appear at the top.

You can move shapes to the left or right, rotate them counterclockwise, or drop them to the bottom by pressing the appropriate keys. As you fit them together, completed horizontal rows vanish, and any blocks above fall down to fill in. When the blocks stack up to the top of the screen, the game is over.

SCORING

You get one point for every block you fit into the stack, and one point for every space a block falls when you hit the drop key. (Dropping the blocks is therefore a good way to increase your score.) Your total score is the product of the level of play and your accumulated points—200 points on level 3 gives you a score of 600. Each player gets at most one entry on any level, for a total of nine scores in the high scores file. Players who no longer have accounts are limited to one score. Also, scores over 5 years old are expired. The exception to these conditions is that the highest score on a given level is *always* kept, so that following generations can pay homage to those who have wasted serious amounts of time.

The score list is produced at the end of the game. The printout includes each player’s overall ranking, name, score, and how many points were scored on what level. Scores which are the highest on a given level are marked with asterisks “*”.

FILES

/var/games/tetris.scores	high score file

AUTHORS

Adapted from a 1989 International Obfuscated C Code Contest winner by Chris Torek and Darren F. Provine.

Manual adapted from the original entry written by Nancy L. Tinkham and Darren F. Provine.

Code for previewing next shape added by Hubert Feyrer in 1999.

BUGS

The higher levels are unplayable without a fast terminal connection.

NAME

trek — trekkie game

SYNOPSIS

trek [[**-a**] *file*]

DESCRIPTION

trek is a game of space glory and war. Below is a summary of commands. For complete documentation, see *Trek* by Eric Allman.

If a filename is given, a log of the game is written onto that file. If the **-a** flag is given before the filename, that file is appended to, not truncated.

The game will ask you what length game you would like. Valid responses are “short”, “medium”, and “long”. You may also type “restart”, which restarts a previously saved game. You will then be prompted for the skill, to which you must respond “novice”, “fair”, “good”, “expert”, “commodore”, or “impossible”. You should normally start out with a novice and work up.

In general, throughout the game, if you forget what is appropriate the game will tell you what it expects if you just type in a question mark.

SEE ALSO

/usr/share/doc/usd/31.trek

AUTHORS

Eric Allman

COMMAND SUMMARY

abandon
capture
cloak **u**p/**d**own
computer request; ...
damages
destruct
dock
help
impulse course distance
lrscan
move course distance
phasers **a**utomatic amount
phasers **m**anual amt1 course1 spread1 ...
torpedo course [**y**es] angle/**n**o
ram course distance
rest time
shell
shields **u**p/**d**own
srscan [**y**es/ **n**o]
status
terminate yes/**n**o
undock
visual course

warp warp_factor

NAME

wargames — shall we play a game?

SYNOPSIS

wargames

DESCRIPTION

“Shall we play a game?” -- computer, **wargames**

Just like in the movie, the computer will happily play a game with you. The likelihood of Global Thermonuclear Warfare resulting is much smaller....

SEE ALSO

Wargames, the movie (an MGM production, PG 13, directed by John Badham, 1983).

AUTHORS

This manual page was written by Joey Hess ⟨joeyh@kitenet.net⟩.

NAME

worm — Play the growing worm game

SYNOPSIS

worm [*size*]

DESCRIPTION

In **worm**, you are a little worm, your body is the "o"'s on the screen and your head is the "@". You move with the hjkl keys and the arrow keys (as in the game snake). If you don't press any keys, you continue in the direction you last moved. The upper case HJKL keys move you as if you had pressed several (9 for HL and 5 for JK) of the corresponding lower case key (unless you run into a digit, then it stops).

On the screen you will see a digit, if your worm eats the digit is will grow longer, the actual amount longer depends on which digit it was that you ate. The object of the game is to see how long you can make the worm grow.

The game ends when the worm runs into either the sides of the screen, or itself. The current score (how much the worm has grown) is kept in the upper right corner of the screen.

The optional argument, if present, is the initial length of the worm.

NAME

worms — animate worms on a display terminal

SYNOPSIS

worms [**-ft**] [**-d** *delay*] [**-l** *length*] [**-n** *number*]

DESCRIPTION

A UNIX version of the DEC-2136 program “worms”.

The options are as follows:

-f	Makes a “field” for the worm(s) to eat.
-t	Makes each worm leave a trail behind it.
-d	Specifies a delay, in milliseconds, between each update. This is useful for fast terminals. Reasonable values are around 20-200. The default is 0.
-l	Specifies a length for each worm; the default is 16.
-n	Specifies the number of worms; the default is 3.

NAME

wtf — translates acronyms for you

SYNOPSIS

wtf [**-f** *dbfile*] [*is*] *acronym ...*

DESCRIPTION

The **wtf** utility displays the expansion of the acronyms specified on the command line. If the acronym is not in any of the acronyms databases, **wtf** will check to see if the acronym is known by whatis(1), pkg_info(1), or via pkgsrc's internal help mechanism, "make help topic=XXX".

If "is" is specified on the command line, it will be ignored, allowing the fairly natural "wtf is WTF" usage.

The following options are available:

-f *dbfile*
Overrides the default acronym database, bypassing the value of the ACRONYMDB variable.

ENVIRONMENT

ACRONYMDB	The default acronym database may be overridden by setting the environment variable ACRONYMDB to the name of one or more space-separated file names of acronym databases. The files must be in the proper format (acronym[tab]meaning).

FILES

/usr/share/misc/acronyms	default acronym database.
/usr/share/misc/acronyms.comp	computer-related acronym database.

SEE ALSO

make(1), pkg_info(1), whatis(1)

HISTORY

wtf first appeared in NetBSD 1.5.

NAME

wump — hunt the wumpus in an underground cave

SYNOPSIS

wump [**−h**] [**−a** *arrows*] [**−b** *bats*] [**−p** *pits*] [**−r** *rooms*] [**−t** *tunnels*]

DESCRIPTION

The game **wump** is based on a fantasy game first presented in the pages of *People's Computer Company* in 1973. In Hunt the Wumpus you are placed in a cave built of many different rooms, all interconnected by tunnels. Your quest is to find and shoot the evil Wumpus that resides elsewhere in the cave without running into any pits or using up your limited supply of arrows.

The options are as follows:

−a	Specifies the number of magic arrows the adventurer gets. The default is five.
−b	Specifies the number of rooms in the cave which contain bats. The default is three.
−h	Play the hard version -- more pits, more bats, and a generally more dangerous cave.
−p	Specifies the number of rooms in the cave which contain bottomless pits. The default is three.
−r	Specifies the number of rooms in the cave. The default cave size is twenty rooms.
−t	Specifies the number of tunnels connecting each room in the cave to another room. Beware, too many tunnels in a small cave can easily cause it to collapse! The default cave room has three tunnels to other rooms.

While wandering through the cave you'll notice that, while there are tunnels everywhere, there are some mysterious quirks to the cave topology, including some tunnels that go from one room to another, but not necessarily back! Also, most pesky of all are the rooms that are home to large numbers of bats, which, upon being disturbed, will en masse grab you and move you to another portion of the cave (including those housing bottomless pits, sure death for unwary explorers).

Fortunately, you're not going into the cave without any weapons or tools, and in fact your biggest aids are your senses; you can often smell the rather odiferous Wumpus up to *two* rooms away, and you can always feel the drafts created by the occasional bottomless pit and hear the rustle of the bats in caves they might be sleeping within.

To kill the wumpus, you'll need to shoot it with one of your magic arrows. Fortunately, you don't have to be in the same room as the creature, and can instead shoot the arrow from as far as three or four rooms away!

When you shoot an arrow, you do so by typing in a list of rooms that you'd like it to travel to. If at any point in its travels it cannot find a tunnel to the room you specify from the room it's in, it will instead randomly fly down one of the tunnels, possibly, if you're real unlucky, even flying back into the room you're in and hitting you!