FIRST EDITION

International Standard Book Number: 0-672-31104-6

Library of Congress Catalog Card Number: 97-66202

2000 99 98 97 4 3 2 1

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit, the number of the book's printing. For example, a printing code of 97-1 shows that the first printing of the book occurred in 1997.

Composed in AGaramond and MCPdigital by Macmillan Computer Publishing

Printed in the United States of America

DEFINITIONS

blank	A space or tab.
word	A sequence of characters considered as a single unit by the shell. Also known as a token.
name	A word consisting only of alphanumeric characters and underscores and beginning with an alphabetic character or an underscore. Also referred to as an identifier.
meta character	A character that, when unquoted, separates words. One of the following: |, &, ;, (,), <, >, space, tab
control operator	A token that performs a control function. It is one of the following symbols: ||, &, &&, ;, ;;, (,), |, <newline>

RESERVED WORDS

Reserved words are words that have a special meaning to the shell. The following words are recognized as reserved when unquoted and either the first word of a simple command (see "Shell Grammar," next) or the third word of a case or for command:

```
! case  do done elif else esac fi for function if in select then until while {  }
```

SHELL GRAMMAR

SIMPLE COMMANDS

A *simple command* is a sequence of optional variable assignments followed by words and redirections separated by blank and terminated by a control operator. The first word specifies the command to be executed. The remaining words are passed as arguments to the invoked command.

The return value of a simple command is its exit status, or `128+n` if the command is terminated by signal `n`.

PIPELINES

A *pipeline* is a sequence of one or more commands separated by the character |. The format for a pipeline is

```
[!]command [ ¦ command2 ... ]
```

The standard output of command is connected to the standard input of `command2`. This connection is performed before any redirections specified by the command. (See the "Redirection" section, later in this manual page.)

If the reserved word `!` precedes a pipeline, the exit status of that pipeline is the logical `NOT` of the exit status of the last command. Otherwise, the status of the pipeline is the exit status of the last command. The shell waits for all commands in the pipeline to terminate before returning a value.

Each command in a pipeline is executed as a separate process (that is, in a subshell).

LISTS

A *list* is a sequence of one or more pipelines separated by one of these operators: ;, `&`, `&&`, or ¦¦, and terminated by one of these: ;, `&`, or `<newline>`.

Of these list operators, `&&` and ¦¦ have equal precedence, followed by ; and `&`, which have equal precedence.

If a command is terminated by the control operator `&`, the shell executes the command in the background in a subshell. The shell does not wait for the command to finish, and the return status is `0`. Commands separated by a ; are executed sequentially; the shell waits for each command to terminate in turn. The return status is the exit status of the last command executed.

The control operators `&&` and ¦¦ denote `AND` lists and `OR` lists, respectively. An `AND` list has the form:

```
command && command2
```

`command2` is executed if, and only if, command returns an exit status of `Zero`.

BRACE EXPANSION

Brace expansion is a mechanism by which arbitrary strings may be generated. This mechanism is similar to pathname expansion, but the filenames generated need not exist. Patterns to be brace expanded take the form of an optional preamble, followed by a series of comma-separated strings between a pair of braces, followed by an optional postamble. The preamble is prepended to each string contained within the braces, and the postamble is then appended to each resulting string, expanding left to right.

Brace expansions may be nested. The results of each expanded string are not sorted; left to right order is preserved. For example, `a{d,c,b}e` expands into `ade ace abe`.

Brace expansion is performed before any other expansions, and any characters special to other expansions are preserved in the result. It is strictly textual. `bash` does not apply any syntactic interpretation to the context of the expansion or the text between the braces.

A correctly formed brace expansion must contain unquoted opening and closing braces, and at least one unquoted comma. Any incorrectly formed brace expansion is left unchanged.

This construct is typically used as shorthand when the common prefix of the strings to be generated is longer than in the preceding example, such as

```
mkdir /usr/local/src/bash/{old,new,dist,bugs}
```

or

```
chown root /usr/{ucb/{ex,edit},lib/ex?.?*,how_ex}}
```

Brace expansion introduces a slight incompatibility with traditional versions of `sh`, the Bourne shell. `sh` does not treat opening or closing braces specially when they appear as part of a word, and preserves them in the output. `bash` removes braces from words as a consequence of brace expansion. For example, a word entered to `sh` as `file{1,2}` appears identically in the output. The same word is output as `file1 file2` after expansion by `bash`. If strict compatibility with `sh` is desired, start `bash` with the `-nobraceexpansion` flag (see "Options," earlier in this manual page) or disable brace expansion with the `+o braceexpand` option to the `set` command. (See "Shell Built-in Commands.")

TILDE EXPANSION

If a word begins with a tilde character (`~`), all of the characters preceding the first slash (or all characters, if there is no slash) are treated as a possible login name. If this login name is the null string, the tilde is replaced with the value of the parameter `HOME`. If `HOME` is unset, the home directory of the user executing the shell is substituted instead.

If a `+` follows the tilde, the value of `PWD` replaces the tilde and `+` If a `-` follows, the value of `OLDPWD` is substituted. If the value following the tilde is a valid login name, the tilde and login name are replaced with the home directory associated with that name. If the name is invalid, or the tilde expansion fails, the word is unchanged.

Each variable assignment is checked for unquoted instances of tildes following `a : or =`. In these cases, tilde substitution is also performed. Consequently, one may use pathnames with tildes in assignments to `PATH`, `MAILPATH`, and `CDPATH`, and the shell assigns the expanded value.

PARAMETER EXPANSION

The `$` character introduces parameter expansion, command substitution, or arithmetic expansion. The parameter name or symbol to be expanded may be enclosed in braces, which are optional but serve to protect the variable to be expanded from characters immediately following it which could be interpreted as part of the name.

`${parameter}` The value of `parameter` is substituted. The braces are required when `parameter` is a positional parameter with more than one digit, or when `parameter` is followed by a character that is not to be interpreted as part of its name.

In each of the following cases, `word` is subject to tilde expansion, parameter expansion, command substitution, and arithmetic expansion. `bash` tests for a parameter that is unset or null; omitting the colon results in a test only for a parameter that is unset.

`${parameter:-word}`	Use default values. If `parameter` is unset or null, the expansion of `word` is substituted. Otherwise, the value of `parameter` is substituted.
`${parameter:=word}`	Assign default values. If `parameter` is unset or null, the expansion of `word` is assigned to parameter. The value of `parameter` is then substituted. Positional parameters and special parameters may not be assigned to in this way.
`${parameter:?word}`	Display Error if null or unset. If `parameter` is null or unset, the expansion of `word` (or a message to that effect if `word` is not present) is written to the standard error and the shell, if it is not interactive, exits. Otherwise, the value of `parameter` is substituted.
`${parameter:+word}`	Use Alternate Value. If `parameter` is null or unset, nothing is substituted; otherwise, the expansion of `word` is substituted.
`${#parameter}`	The length in characters of the value of `parameter` is substituted. If parameter is `*` or `@`, the length substituted is the length of `*` expanded within double quotes.
`${parameter#word}` `${parameter##word}`	The `word` is expanded to produce a pattern just as in pathname expansion. If the pattern matches the beginning of the value of `parameter`, then the expansion is the value of `parameter` with the shortest matching pattern deleted (the `#` case) or the longest matching pattern deleted (the `##` case).
`${parameter%word}` `${parameter%%word}`	The `word` is expanded to produce a pattern just as in pathname expansion. If the pattern matches a trailing portion of the value of `parameter`, then the expansion is the value of `parameter` with the shortest matching pattern deleted (the `%` case) or the longest matching pattern deleted (the `%%` case).

COMMAND SUBSTITUTION

Command substitution allows the output of a command to replace the command name.

There are two forms:

```
$(command )
```

or

```
‘command’
```

performs the expansion by executing `command` and replacing the command substitution with the standard output of the command, with any trailing newlines deleted.

When the old–style backquote form of substitution is used, backslash retains its literal meaning except when followed by `$`, `'`, or `\`. When using the `$(command)` form, all characters between the parentheses make up the command; none are treated specially.

Command substitutions may be nested. To nest when using the old form, escape the inner backquotes with backslashes.

If the substitution appears within double quotes, word splitting and pathname expansion are not performed on the results.

ARITHMETIC EXPANSION

Arithmetic expansion allows the evaluation of an arithmetic expression and the substitution of the result. There are two formats for arithmetic expansion:

```
$[expression]
$((expression))
```

The expression is treated as if it were within double quotes, but a double quote inside the braces or parentheses is not treated specially. All tokens in the expression undergo parameter expansion, command substitution, and quote removal. Arithmetic substitutions may be nested.

The evaluation is performed according to the rules listed under “Arithmetic Evaluation,” later in this section. If `expression` is `invalid`, `bash` prints a message indicating failure and no substitution occurs.

PROCESS SUBSTITUTION

Process substitution is supported on systems that support named pipes (FIFOs) or the `/dev/fd` method of naming open files. It takes the form of `<(list)` or `>(list)`. The process list is run with its input or output connected to a FIFO or some file in `/`

/dev/fd. The name of this file is passed as an argument to the current command as the result of the expansion. If the >(list) form is used, writing to the file will provide input for list. If the <(list) form is used, the file passed as an argument should be read to obtain the output of list.

On systems that support it, process substitution is performed simultaneously with parameter and variable expansion, command substitution, and arithmetic expansion.

WORD SPLITTING

The shell scans the results of parameter expansion, command substitution, and arithmetic expansion that did not occur within double quotes for word splitting.

The shell treats each character of IFS as a delimiter, and splits the results of the other expansions into words on these characters. If the value of IFS is exactly <space><tab><newline>, the default, then any sequence of IFS characters serves to delimit words. If IFS has a value other than the default, then sequences of the whitespace characters space and tab are ignored at the beginning and end of the word, as long as the whitespace character is in the value of IFS (an IFS whitespace character). Any character in IFS that is not IFS whitespace, along with any adjacent IFS whitespace characters, delimits a field. A sequence of IFS whitespace characters is also treated as a delimiter. If the value of IFS is null, no word splitting occurs. IFS cannot be unset.

Explicit null arguments ("" or '') are retained. Implicit null arguments, resulting from the expansion of parameters that have no values, are removed.

Note that if no expansion occurs, no splitting is performed.

PATHNAME EXPANSION

After word splitting, unless the -f option has been set, bash scans each word for the characters *, ?, and [. If one of these characters appears, then the word is regarded as a pattern and replaced with an alphabetically sorted list of pathnames matching the pattern. If no matching pathnames are found, and the shell variable allow_null_glob_expansion is unset, the word is left unchanged. If the variable is set, and no matches are found, the word is removed. When a pattern is used for pathname generation, the character (.) at the start of a name or immediately following a slash must be matched explicitly, unless the shell variable glob_dot_filenames is set. The slash character must always be matched explicitly. In other cases, the (.) character is not treated specially.

The special pattern characters have the following meanings:

*	Matches any string, including the null string.
?	Matches any single character.
[...]	Matches any one of the enclosed characters. A pair of characters separated by a minus sign denotes a range; any character lexically between those two characters, inclusive, is matched. If the first character following the [is a ! or a ^, then any character not enclosed is matched. A - or] may be matched by including it as the first or last character in the set.

QUOTE REMOVAL

After the preceding expansions, all unquoted occurrences of the characters \, ', and " are removed.

REDIRECTION

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. Redirection may also be used to open and close files for the current shell execution environment. The following redirection operators may precede or appear anywhere within a simple command or may follow a command. Redirections are processed in the order they appear, from left to right.

In the following descriptions, if the file descriptor number is omitted, and the first character of the redirection operator is <, the redirection refers to the standard input (file descriptor 0). If the first character of the redirection operator is >, the redirection refers to the standard output (file descriptor 1).

The word that follows the redirection operator in the following descriptions is subjected to brace expansion, tilde expansion, parameter expansion, command substitution, arithmetic expansion, quote removal, and pathname expansion. If it expands to more than one word, bash reports an error.

Note that the order of redirections is significant. For example, the command:

```
ls > dirlist 2>&1
```

directs both standard output and standard error to the file dirlist, while the command

```
ls 2>&1 > dirlist
```

directs only the standard output to file dirlist, because the standard error was duplicated as standard output before the standard output was redirected to dirlist.

REDIRECTING INPUT

Redirection of input causes the file whose name results from the expansion of word to be opened for reading on file descriptor n, or the standard input (file descriptor 0) if n is not specified.

The general format for redirecting input is

```
[n]<word
```

REDIRECTING OUTPUT

Redirection of output causes the file whose name results from the expansion of word to be opened for writing on file descriptor n, or the standard output (file descriptor 1) if n is not specified. If the file does not exist, it is created; if it does exist it is truncated to zero size.

The general format for redirecting output is

```
[n]>word
```

If the redirection operator is >|, then the value of the -C option to the set built-in command is not tested, and file creation is attempted. (See also the description of noclobber under "Shell Variables," earlier in this manual page.)

APPENDING REDIRECTED OUTPUT

Redirection of output in this fashion causes the file whose name results from the expansion of word to be opened for appending on file descriptor n, or the standard output (file descriptor 1) if n is not specified. If the file does not exist, it is created.

The general format for appending output is

```
[n]>>word
```

REDIRECTING STANDARD OUTPUT AND STANDARD ERROR

bash allows both the standard output (file descriptor 1) and the standard error output (file descriptor 2) to be redirected to the file whose name is the expansion of word with this construct.

There are two formats for redirecting standard output and standard error:

```
&>word
```

and

```
>&word
```

Of the two forms, the first is preferred. This is semantically equivalent to

```
>word 2>&1
```

HERE-DOCUMENTS

This type of redirection instructs the shell to read input from the current source until a line containing only word (with no trailing blanks) is seen. All of the lines read up to that point are then used as the standard input for a command.

is tested for aliases, but a word that is identical to an alias being expanded is not expanded a second time. This means that one may alias `ls` to `ls -F`, for instance, and `bash` does not try to recursively expand the replacement text. If the last character of the alias value is a blank, then the next command word following the alias is also checked for alias expansion.

Aliases are created and listed with the `alias` command, and removed with the `unalias` command.

There is no mechanism for using arguments in the replacement text, as in `csh`. If arguments are needed, a shell function should be used.

Aliases are not expanded when the shell is not interactive.

The rules concerning the definition and use of aliases are somewhat confusing. `bash` always reads at least one complete line of input before executing any of the commands on that line. Aliases are expanded when a command is read, not when it is executed. Therefore, an alias definition appearing on the same line as another command does not take effect until the next line of input is read. This means that the commands following the alias definition on that line are not affected by the new alias. This behavior is also an issue when functions are executed. Aliases are expanded when the function definition is read, not when the function is executed, because a function definition is itself a compound command. As a consequence, aliases defined in a function are not available until after that function is executed. To be safe, always put alias definitions on a separate line, and do not use alias in compound commands.

Note that for almost every purpose, aliases are superseded by shell functions.

JOB CONTROL

Job control refers to the ability to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point. A user typically employs this facility via an interactive interface supplied jointly by the system's terminal driver and `bash`.

The shell associates a job with each pipeline. It keeps a table of currently executing jobs, which may be listed with the jobs command. When `bash` starts a job asynchronously (in the background), it prints a line that looks like this:

```
[1] 25647
```

indicating that this job is job number 1 and that the process ID of the last process in the pipeline associated with this job is 25647. All of the processes in a single pipeline are members of the same job. `bash` uses the job abstraction as the basis for job control.

To facilitate the implementation of the user interface to job control, the system maintains the notion of a current terminal process group ID. Members of this process group (processes whose process group ID is equal to the current terminal process group ID) receive keyboard-generated signals such as `SIGINT`. These processes are said to be in the foreground. Background processes are those whose process group ID differs from the terminal's; such processes are immune to keyboard-generated signals. Only foreground processes are allowed to read from or write to the terminal. Background processes that attempt to read from (write to) the terminal are sent a `SIGTTIN` (`SIGTTOU`) signal by the terminal driver, which, unless caught, suspends the process.

If the operating system on which `bash` is running supports job control, `bash` allows you to use it. Typing the suspend character (typically `^Z`, `Control-Z`) while a process is running causes that process to be stopped and returns you to `bash`. Typing the delayed suspend character (typically `^Y`, `Control-Y`) causes the process to be stopped when it attempts to read input from the terminal, and control to be returned to `bash`. You may then manipulate the state of this job, using the `bg` command to continue it in the background, the `fg` command to continue it in the foreground, or the `kill` command to kill it. A Ctrl+Z takes effect immediately, and has the additional side effect of causing pending output and typeahead to be discarded.

There are a number of ways to refer to a job in the shell. The character `%` introduces a job name. Job number `n` may be referred to as `%n`. A job may also be referred to using a prefix of the name used to start it, or using a substring that appears in its command line. For example, `%ce` refers to a stopped `ce` job. If a prefix matches more than one job, `bash` reports an error. Using `%?ce`, on the other hand, refers to any job containing the string `ce` in its command line. If the substring matches more than one job, `bash` reports an error. The symbols `%%` and `%+` refer to the shell's notion of the current job, which is the last job

parameter and variable expansion (see "Expansion," earlier in this manual page) but after history expansion is performed, subject to the values of the shell variables `command_oriented_history` and `HISTCONTROL`. On startup, the history is initialized from the file named by the variable `HISTFILE` (default `~/.bash_history`). HISTFILE is truncated, if necessary, to contain no more than `HISTFILESIZE` lines. The built-in command `fc` (see Shell Built-in Commands, later in this manual page) may be used to list or edit and re-execute a portion of the history list. The `history` builtin can be used to display the history list and manipulate the history file. When using the command-line editing, search commands are available in each editing mode that provide access to the history list. When an interactive shell exits, the last `HISTSIZE` lines are copied from the history list to `HISTFILE`. If `HISTFILE` is unset, or if the history file is unwritable, the history is not saved.

HISTORY EXPANSION

The shell supports a history expansion feature that is similar to the history expansion in `csh`. This section describes what syntax features are available. This feature is enabled by default for interactive shells, and can be disabled using the `H` option to the `set` built-in command. (See "Shell Built-in Commands," later in this manual page.) Noninteractive shells do not perform history expansion.

History expansion is performed immediately after a complete line is read, before the shell breaks it into words. It takes place in two parts. The first is to determine which line from the previous history to use during substitution. The second is to select portions of that line for inclusion into the current one. The line selected from the previous history is the event, and the portions of that line that are acted upon are words. The line is broken into words in the same fashion as when reading input, so that several meta character–separated words surrounded by quotes are considered as one word. Only the backslash (`\`) and single quotes can quote the history escape character, which is `!` by default.

The shell allows control of the various characters used by the history expansion mechanism. (See the description of `histchars` under "Shell Variables," earlier in this manual page.)

EVENT DESIGNATORS

An event designator is a reference to a command line entry in the history list.

`!`	Start a history substitution, except when followed by a blank, newline, `=`, or `(`.
`!!`	Refer to the previous command. This is a synonym for `!-1`.
`!n`	Refer to command line `n`.
`!-n`	Refer to the current command line minus `n`.
`!string`	Refer to the most recent command starting with `string`.
`!?string[?]`	Refer to the most recent command containing `string`.
`^string1^string2^`	Quick substitution. Repeat the last command, replacing `string1` with `string2`. Equivalent to `!!:s/string1/string2/`. (See "Modifiers," later in this manual page.)
`!#`	The entire command line typed so far.

WORD DESIGNATORS

A colon (`:`) separates the event specification from the word designator. It can be omitted if the word designator begins with a `^`, `$`, `*`, or `%`. Words are numbered from the beginning of the line, with the first word being denoted by a `0` (zero).

`0` (zero)	The zeroth word. For the shell, this is the command word.
`n`	The *n*th word.
`^`	The first argument. That is, word 1.
`$`	The last argument.
`%`	The word matched by the most recent `?string?` search.
`x-y`	A range of words; `-y` abbreviates `0-y`.
`*`	All of the words but the zeroth. This is a synonym for `1-$`. It is not an error to use `*` if there is just one word in the event; the empty string is returned in that case.
`x*`	Abbreviates `x-$`.
`x-`	Abbreviates `x-$` like `x*`, but omits the last word.

bdftopcf

bdftopcf—Convert X font from Bitmap Distribution Format to Portable Compiled Format

SYNOPSIS

```
bdftopcf [ -pn ][-un ][-m ][-l ][-M ][-L ][-t ][-i ][-o outputfile ] fontfile.bdf
```

DESCRIPTION

bdftopcf is a font compiler for the X server and font server. Fonts in Portable Compiled Format can be read by any architecture, although the file is structured to allow one particular architecture to read them directly without reformatting. This allows fast reading on the appropriate machine, but the files are still portable (but read more slowly) on other machines.

OPTIONS

-pn	Sets the font glyph padding. Each glyph in the font will have each scanline padded in to a multiple of n bytes, where n is 1, 2, 4, or 8.
-u*n*	Sets the font scanline unit. When the font bit order is different from the font byte order, the scanline unit n describes what unit of data (in bytes) are to be swapped. The unit i can be 1, 2, or 4 bytes.
-m	Sets the font bit order to MSB (most significant bit) first. Bits for each glyph will be placed in this order; that is, the leftmost bit on the screen will be in the highest valued bit in each unit.
-l	Sets the font bit order to LSB (least significant bit) first. The leftmost bit on the screen will be in the lowest valued bit in each unit.
-M	Sets the font byte order to MSB first. All multibyte data in the file (metrics, bitmaps, and everything else) will be written most significant byte first.
-L	Sets the font byte order to LSB first. All multibyte data in the file (metrics, bitmaps, and everything else) will be written least significant byte first.
-t	When this option is specified, bdftopcf will convert fonts into terminal fonts when possible. A *terminal font* has each glyph image padded to the same size; the X server can usually render these types of fonts more quickly.
-i	This option inhibits the normal computation of ink metrics. When a font has glyph images that do not fill the bitmap image (that is, the “on” pixels don’t extend to the edges of the metrics), bdftopcf computes the actual ink metrics and places them in the PCF file; the -t option inhibits this behavior.
-o output-file-name	By default bdftopcf writes the PCF file to standard output; this option gives the name of a file to be used instead.

SEE ALSO

X(1)

AUTHOR

Keith Packard, MIT X Consortium

X Version 11 Release 6

beforelight

beforelight—Screen saver

SYNOPSIS

```
beforelight [ -toolkitoption ... ]
```

DESCRIPTION

The `beforelight` program is a sample implementation of a screen saver for `X` servers supporting the `MIT-SCREEN-SAVER` extension.

AUTHORS

Keith Packard (MIT X Consortium)

X Version 11 Release 6

biff

`biff`—Be notified if mail arrives and who it is from

SYNOPSIS

```
biff [ny]
```

DESCRIPTION

`biff` informs the system whether you want to be notified when mail arrives during the current terminal session.

Options supported by `biff`:

`n` Disables notification

`y` Enables notification

When mail notification is enabled, the header and first few lines of the message will be printed on your screen whenever mail arrives. A

```
biff y
```

command is often included in the file `.login` or `.profile` to be executed at each login.

Biff operates asynchronously. For synchronous notification use the `MAIL` variable of `sh`(1) or the mail variable of `csh`(1).

SEE ALSO

`csh`(1), `mail`(1), `sh`(1), `comsat`(8)

HISTORY

The `biff` command appeared in BSD 4.0.

BSD 4, 14 March 1991

bioradtopgm

`bioradtopgm`—Convert a Biorad confocal file into a portable graymap

SYNOPSIS

```
bioradtopgm [-image#][imagedata]
```

DESCRIPTION

Reads a Biorad confocal file as input. Produces a portable graymap as output. If the resulting image is upside down, run it through `pnmflip -tb`.

```
Toggle point
Toggle curve
Toggle line
Toggle rectangle
Toggle filledRectangle
Toggle circle
Toggle filledCircle
Toggle floodFill
Toggle setHotSpot
Command clearHotSpot
Command undo
```

COLORS

If you would like bitmap to be viewable in color, include the following in the `#ifdef COLOR` section of the file you read with `xrdb`:

```
*customization: -color
```

This will cause bitmap to pick up the colors in the app-defaults color customization file:

```
<XRoot>/lib/X11/app-defaults/Bitmap-color
```

where `<XRoot>` refers to the root of the X11 install tree.

BITMAP WIDGET

Bitmap widget is a standalone widget for editing raster images. It is not designed to edit large images, although it may be used in that purpose as well. It can be freely incorporated with other applications and used as a standard editing tool. The following are the resources provided by the bitmap widget:

Header file	`Bitmap.h`
Class	`bitmapWidgetClass`
Class Name	`Bitmap`
Superclass	`Bitmap`

All the Simple Widget resources plus…

Name	*Class*	*Type*	*Default Value*
foreground	Foreground	Pixel	XtDefaultForeground
highlight	Highlight	Pixel	XtDefaultForeground
framing	Framing	Pixel	XtDefaultForeground
gridTolerance	GridTolerance	Dimension	8
size	Size	String	32x32
dashed	Dashed	Boolean	True
grid	Grid	Boolean	True
stippled	Stippled	Boolean	True
proportional	Proportional	Boolean	True
axes	Axes	Boolean	False
squareWidth	SquareWidth	Dimension	16
squareHeight	SquareHeight	Dimension	16
margin	Margin	Dimension	16
xHot	XHot	Position	NotSet (–1)
yHot	YHot	Position	NotSet (–1)
button1Function	Button1Function	DrawingFunction	Set

Name	*Class*	*Type*	*Default Value*
button2Function	Button2Function	DrawingFunction	Invert
button3Function	Button3Function	DrawingFunction	Clear
button4Function	Button4Function	DrawingFunction	Invert
button5Function	Button5Function	DrawingFunction	Invert
filename	Filename	String	None (“”)
basename	Basename	String	None (“”)

AUTHOR

Davor Matic (MIT X Consortium)

X Version 11 Release 6

bmptoppm

`bmptoppm`—Convert a BMP file into a portable pixmap

SYNOPSIS

```
bmptoppm [bmpfile]
```

DESCRIPTION

`bmptoppm` reads a Microsoft Windows or OS/2 BMP file as input and produces a portable pixmap as output.

SEE ALSO

`ppmtobmp`(1), `ppm`(5)

AUTHOR

26 October 1992

brushtopbm

`brushtopbm`—Convert a doodle brush file into a portable bitmap

SYNOPSIS

```
brushtopbm [brushfile]
```

DESCRIPTION

`brushtopbm` reads a Xerox doodle brush file as input and produces a portable bitmap as output.

Note that there is currently no `pbmtobrush` tool.

SEE ALSO

`pbm`(5)

AUTHOR

28 August 1988

colcrt

`colcrt`—Filter `nroff` output for CRT previewing

SYNOPSIS

```
colcrt [-] [-2] [file ...]
```

DESCRIPTION

`colcrt` provides virtual half-line and reverse-line feed sequences for terminals without such capability, and on which overstriking is destructive. Half-line characters and underlining (changed to dashing -) are placed on new lines in between the normal output lines.

Available options:

`-` Suppress all underlining. This option is especially useful for previewing all boxed tables from `tbl`(1).

`-2` Causes all half-lines to be printed, effectively double spacing the output. Normally, a minimal space output format is used which will suppress empty lines. The program never suppresses two consecutive empty lines, however. The `-2` option is useful for sending output to the line printer when the output contains superscripts and subscripts that would otherwise be invisible.

EXAMPLES

A typical use of `colcrt` would be

```
tbl exum2.n | nroff -ms | colcrt - | more
```

SEE ALSO

`nroff`(1), `troff`(1), `col`(1), `more`(1), `ul`(1)

BUGS

Should fold underlines onto blanks even with the `-` option so that a true underline character would show.

Can't back up more than 102 lines.

General overstriking is lost; as a special case ¦ overstruck with `''` or underline becomes +. Lines are trimmed to 132 characters.

Some provision should be made for processing superscripts and subscripts in documents that are already double-spaced.

HISTORY

The `colcrt` command appeared in BSD 3.0.

BSD 3, 30 June 1993

colrm

`colrm`—Remove columns from a file

SYNOPSIS

```
colrm [startcol [endcol]]
```

DESCRIPTION

`colrm` removes selected columns from a file. Input is taken from standard input. Output is sent to standard output.

If called with one parameter, the columns of each line will be removed starting with the specified column. If called with two parameters, the columns from the first column to the last column will be removed.

Column numbering starts with column 1.

`-V, —version-control {numbered,existing,simple}`

The type of backups made can be set with the `VERSION_CONTROL` environment variable, which can be overridden by this option. If `VERSION_CONTROL` is not set and this option is not given, the default backup type is `existing`. The value of the `VERSION_CONTROL` environment variable and the argument to this option are like the GNU `emacs version- control` variable; they also recognize synonyms that are more descriptive. The valid values are (unique abbreviations are accepted) the following:

`t` or `numbered`	Always make numbered backups
`nil` or `existing`	Make numbered backups of files that already have them, simple backups of the others
`never` or `simple`	Always make simple backups

cccp, cpp

`cccp, cpp`—The GNU C-compatible compiler preprocessor

SYNOPSIS

```
cccp    [-$][-A predicate [( value )]] [ -C ][-D name [= definition ]]
        [-dD][-dM][-I\ directory ][-H ][-I- ][-imacros file ][-
        include file ][-idirafter dir ][-iprefix prefix ][-iwithprefix dir
        [ -lang-c][-lang-c++][-lang-objc][-lang-objc++ ][-lint ][
        M[-MG ]] [ -MM[-MG ]] [ -MMD file ][-MMD file ][-nostdinc ]
        [ -P][-pedantic ][-pedantic-errors ][-traditional ]
        [ -trigraphs ][-U name ][-undef ][-Wtrigraphs ][-Wcomment ]
        [ -Wall ][-Wtraditional ]
        [ infile ¦- ][ outfile ¦- ]
```


DESCRIPTION

The C preprocessor is a macro processor that is used automatically by the C compiler to transform your program before actual compilation. It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs.

The C preprocessor provides four separate facilities that you can use as you see fit:

- Inclusion of header files. These are files of declarations that can be substituted into your program.
- Macro expansion. You can define macros, which are abbreviations for arbitrary fragments of C code, and then the C preprocessor will replace the macros with their definitions throughout the program.
- Conditional compilation. Using special preprocessing directives, you can include or exclude parts of the program according to various conditions.
- Line control. If you use a program to combine or rearrange source files into an intermediate file which is then compiled, you can use line control to inform the compiler of where each source line originally came from.

C preprocessors vary in some details. For a full explanation of the GNU C preprocessor, see the `info` file `cpp.info`, or the manual *The C Preprocessor* . Both of these are built from the same documentation source file, `cpp.texinfo`. The GNU C preprocessor provides a superset of the features of ANSI Standard C.

ANSI Standard C requires the rejection of many harmless constructs commonly used by today's C programs. Such incompatibility would be inconvenient for users, so the GNU C preprocessor is configured to accept these constructs by default. Strictly speaking, to get ANSI Standard C, you must use the options `-trigraphs`, `-undef`, and `-pedantic`, but in practice the consequences of having strict ANSI Standard C make it undesirable to do this.

When you use the C preprocessor, you will usually not have to invoke it explicitly: the C compiler will do so automatically. However, the preprocessor is sometimes useful individually.

When you call the preprocessor individually, either name (`cpp` or `cccp`) will do; they are completely synonymous.

the -m option, thus suppressing the editor invocation, or use the -F option to specify that the argument file contains the log message.

The -r option can be used to commit to a particular symbolic or numeric revision within the RCS file. For example, to bring all your files up to the RCS revision 3.0 (including those that haven't changed), you might use

```
example% cvs commit -r3.0
```

cvs will only allow you to commit to a revision that is on the main trunk (a revision with a single dot). However, you can also commit to a branch revision (one that has an even number of dots) with the -r option. To create a branch revision, one typically use the -b option of the rtag or tag commands. Then, either checkout or update can be used to base your sources on the newly created branch. From that point on, all commit changes made within these working sources will be automatically added to a branch revision, thereby not perturbing mainline development in any way. For example, if you had to create a patch to the 1.2 version of the product, even though the 2.0 version is already under development, you might use this:

```
example% cvs rtag -b -rFCS1_2 FCS1_2 Patch product_module
example% cvs checkout -rFCS1_2_Patch product module
example% cd product module
[[ hack away ]]
example% cvs commit
```

Say you have been working on some extremely experimental software, based on whatever revision you happened to checkout last week. If others in your group would like to work on this software with you, but without disturbing mainline development, you could commit your changes to a new branch. Others can then check out your experimental stuff and utilize the full benefit of cvs conflict resolution. The scenario might look like this:

```
example% cvs tag -b EXPR1
example% cvs update -rEXPR1
[[ hack away ]]
example% cvs commit
```

Others would simply do `cvs checkout -rEXPR1 whatever_module` to work with you on the experimental change.

- `diff [-kl][rcsdiff_options][[-r rev1 ¦ -D date1][-r rev2 ¦ -D date2]] [files...]`
  Requires: Working directory, repository
  Changes: Nothing

You can compare your working files with revisions in the source repository, with the `cvs diff` command. If you don't specify a particular revision, your files are compared with the revisions they were based on. You can also use the standard cvs command option -r to specify a particular revision to compare your files with. Finally, if you use -r twice, you can see differences between two revisions in the repository. You can also specify -D options to diff against a revision in the past. The -r and -D options can be mixed together with at most two options ever specified.

See rcsdiff(1) for a list of other accepted options.

If you don't specify any files, diff will display differences for all those files in the current directory (and its subdirectories, unless you use the standard option -l) that differ from the corresponding revision in the source repository (that is, files that you have changed), or that differ from the revision specified.

- `export [-f lNnQq] -r rev ¦ -D date [-d dir][-k kflag] module...`
  Requires: Repository
  Changes: Current directory

This command is a variant of `cvs checkout`; use it when you want a copy of the source for `module` without the cvs administrative directories. For example, you might use `cvs export` to prepare source for shipment off-site. This command requires that you specify a date or tag (with -D or -r), so that you can count on reproducing the source you ship to others.

The only nonstandard options are `-d dir` (write the source into directory `dir`) and -N (don't shorten module paths). These have the same meanings as the same options in `cvs checkout`.

The -kv option is useful when export is used. This causes any RCS keywords to be expanded such that an import done at some other site will not lose the keyword revision information. Other kflag options may be used with cvs export and are described in co(1).

- history [-report][-flags][-options args][files...]

  Requires: The file $CVSROOT/CVSROOT/history

  Changes: Nothing

cvs keeps a history file that tracks each use of the checkout, commit, rtag, update, and release commands. You can use cvs history to display this information in various formats.

WARNING

cvs history uses -f, -l, -n, and -p in ways that conflict with the descriptions in "Common Command Options" earlier in this manual page.

Several options (shown as [-report] in the preceding bulleted code) control what kind of report is generated:

-c Report on each time commit was used (that is, each time the repository was modified).

-m module Report on a particular module. (You can meaningfully use -m more than once on the command line.)

-o Report on checked-out modules.

-T Report on all tags.

-x type Extract a particular set of record types X from the cvs history. The types are indicated by single letters, which you may specify in combination. Certain commands have a single record type: check-out (type O), release (type F), and rtag (type T). One of four record types may result from an update: W, when the working copy of a file is deleted during update (because it was gone from the repository); U, when a working file was copied from the repository; G, when a merge was necessary and it succeeded; and C, when a merge was necessary but collisions were detected (requiring manual merging). Finally, one of three record types results from commit: M, when a file was modified; A, when a file is first added; and R, when a file is removed.

-e Everything (all record types); equivalent to specifying -xMACFROGWUT.

-z zone Use time zone zone when outputting history records. The zone name LT stands for local time; numeric offsets stand for hours and minutes ahead of UTC. For example, +0530 stands for 5 hours and 30 minutes ahead of (that is, east of) UTC.

The options shown as -flags constrain the report without requiring option arguments:

-a Show data for all users. (The default is to show data only for the user executing cvs history.)

-l Show last modification only.

-w Show only the records for modifications done from the same working directory where cvs history is executing.

The options shown as -options args constrain the report based on an argument:

-b str Show data back to a record containing the string str in either the module name, the filename, or the repository path.

-D date Show data since date.

-p repository Show data for a particular source repository (you can specify several -p options on the same command line).

-r rev Show records referring to revisions since the revision or tag named rev appears in individual RCS files. Each RCS file is searched for the revision or tag.

-t tag Show records since tag tag was last added to the history file. This differs from the -r flag in that it reads only the history file, not the RCS files, and is much faster.

-u name Show records for username.

`C file` A conflict was detected while trying to merge your changes to `file` with changes from the source repository. `file` (the copy in your working directory) is now the output of the `rcsmerge`(1) command on the two versions; an unmodified copy of your file is also in your working directory, with the name `.#file.version`, where `version` is the RCS revision that your modified file started from. (Note that some systems automatically purge files that begin with `.#` if they have not been accessed for a few days. If you intend to keep a copy of your original file, it is a very good idea to rename it.)

`? file` `file` is in your working directory, but does not correspond to anything in the source repository, and is not in the list of files for `cvs` to ignore; see the description of the `-I` option.

Use the `-A` option to reset any sticky tags, dates, or `-k` options. (If you get a working copy of a file by using one of the `-r`, `-D`, or `-k` options, `cvs` remembers the corresponding `tag`, `date`, or `kflag` and continues using it on future updates; use the `-A` option to make `cvs` forget these specifications, and retrieve the `head` version of the file).

The `-jbranch` option merges the changes made between the resulting revision and the revision that it is based on. (For example, if the tag refers to a branch, `cvs` will merge all changes made in that branch into your working file.)

With two `-j` options, `cvs` will merge in the changes between the two respective revisions. This can be used to "remove" a certain delta from your working file. For example, if the file `foo.c` is based on revision 1.6 and I want to remove the changes made between 1.3 and 1.5, I might do this:

```
example% cvs update -j1.5 -j1.3 foo.c # note the order!
```

In addition, each `-j` option can contain an optional date specification which, when used with branches, can limit the chosen revision to one within a specific date. An optional date is specified by adding a colon (:) to the tag:

```
-jSymbolic_Tag:Date_Specifier
```

Use the `-d` option to create any directories that exist in the repository if they're missing from the working directory. (Normally, `update` acts only on directories and files that were already enrolled in your working directory.) This is useful for updating directories that were created in the repository since the initial checkout; but it has an unfortunate side effect. If you deliberately avoided certain directories in the repository when you created your working directory (either through use of a module name or by listing explicitly the files and directories you wanted on the command line), then updating with `-d` will create those directories, which may not be what you want.

Use `-I name` to ignore files whose names match `name` (in your working directory) during the update. You can specify `-I` more than once on the command line to specify several files to ignore. By default, `update` ignores files whose names match any of the following:

```
RCSLOG RCS SCCS
CVS* cvslog.*
tags TAGS
.make.state .nse_depinfo
~ #* .#* ,
.old *.bak *.BAK *.orig *.rej .del-*
.a *.o *.so *.Z *.elc *.ln core
```

Use `-I!` to avoid ignoring any files at all.

The standard `cvs` command options `-f`, `-k`, `-l`, `-P`, `-p`, and `-r` are also available with `update`.

FILES

For more detailed information on `cvs` supporting files, see `cvs`(5).

Files in home directories:

`.cvsrc` The `cvs` initialization file. Lines in this file can be used to specify default options for each cvs command. For example, the line `diff -c` will ensure that `cvs diff` is always passed the `-c` option in addition to any other options passed on the command line.

`.cvswrappers` Specifies wrappers to be used in addition to those specified in the `CVSROOT/cvswrappers` file in the repository.

Argument	*Sample output*	*Explanation*
%e	8	Day of month (leading zero blanked)
%h	Mar	Abbreviated month name*
%H	14	24-hour-clock hour (two digits)
%I	02	12-hour-clock hour (two digits)
%j	067	Julian day number (three digits)
%k	2	12-hour-clock hour (leading zero blanked)
%l	14	24-hour-clock hour (leading zero blanked)
%m	03	Month number (two digits)
%M	54	Minute (two digits)
%n	nn	Newline character
%p	PM	AM/PM designation
%r	02:54:40 PM	Hour:minute:second AM/PM designation
%R	14:54	Hour:minute
%S	40	Second (two digits)
%t	nt	Tab character
%T	14:54:40	Hour:minute:second
%U	10	Sunday-based week number (two digits)
%w	3	Day number (one digit, Sunday is 0)
%W	10	Monday-based week number (two digits)
%x	03/08/89	Date*
%X	14:54:40	Time*
%y	89	Last two digits of year
%Y	1989	Year in full
%Z	EST	Time zone abbreviation
%+	Wed Mar 8 14:54:40 EST 1989	Default output format*

* The exact output depends on the locale.

If a character other than one of those shown in the preceding table appears after a percent sign in the format, that following character is output. All other characters in the format are copied unchanged to the output; a newline character is always added at the end of the output.

In Sunday-based week numbering, the first Sunday of the year begins week 1; days preceding it are part of week 0. In Monday-based week numbering, the first Monday of the year begins week 1.

To set the date, use a command-line argument with one of the following forms:

1454	24-hour-clock hours (first two digits) and minutes
081454	Month day (first two digits), hours, and minutes
03081454	Month (two digits, January is 01), month day, hours, minutes
8903081454	Year, month, month day, hours, minutes
0308145489	Month, month day, hours, minutes, year (on System V-compatible systems)
030814541989	Month, month day, hours, minutes, four-digit year
198903081454	Four-digit year, month, month day, hours, minutes

SEE ALSO

lsmod(1), kerneld(8), ksyms(1), modules(2)

REQUIRED UTILITIES

insmod(1), nm(1) rmmod(1)

NOTES

The pattern supplied to modprobe will often be escaped to ensure that it is evaluated in the proper context.

AUTHOR

Jacques Gelinas (jack@solucorp.qc.ca), Bjorn Ekwall (bj0rn@blox.se)

BUGS

Naah…

Linux, 14 May 1995

df

df—Summarize free disk space

SYNOPSIS

```
df [-aikPv] [-t fstype] [-x fstype] [—all] [—inodes] [—type=fstype]
[—exclude-type=fstype] [—kilobytes] [—portability] [—print-type]
[—help] [—version] [filename...]
```

DESCRIPTION

This manual page documents the GNU version of df. df displays the amount of disk space available on the filesystem containing each filename argument. If no filename is given, the space available on all currently mounted filesystems is shown. Disk space is shown in 1K blocks by default, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used.

If an argument is the absolute filename of a disk device node containing a mounted filesystem, df shows the space available on that filesystem rather than on the filesystem containing the device node (which is always the root filesystem). This version of df cannot show the space available on unmounted filesystems, because on most kinds of systems doing so requires very nonportable, intimate knowledge of filesystem structures.

OPTIONS

-a, —all	Include in the listing filesystems that have 0 blocks, which are omitted by default. Such filesystems are typically special-purpose pseudo-filesystems, such as automounter entries. On some systems, filesystems of type ignore or auto are also omitted by default and included in the listing by this option.
-i, —inodes	List inode usage information instead of block usage. An inode (short for “index node”) is a special kind of disk block that contains information about a file, such as its owner, permissions, timestamps, and location on the disk.
-k, —kilobytes	Print sizes in 1K blocks instead of 512-byte blocks. This overrides the environment variable POSIXLY_CORRECT.
-P, —portability	Use the POSIX output format. This is like the default format except that the information about each filesystem is always printed on exactly one line; a mount device is never put on a line by itself. This means that if the mount device name is more than 20 characters long (as for some network mounts), the columns are misaligned.

FILES

`/etc/resolv.conf`	Initial domain name and name server addresses

ENVIRONMENT

`LOCALRES` file to use in place of `/etc/resolv.conf`

`LOCALDEF` default environment file

AUTHOR

Steve Hotz (`hotz@isi.edu`)

ACKNOWLEDGMENTS

`dig` uses functions from `nslookup`(8) authored by Andrew Cherenson.

BUGS

`dig` has a serious case of "creeping featurism," the result of considering several potential uses during its development. It would probably benefit from a rigorous diet. Similarly, the print flags and brevity of the items they specify make evident their rather ad hoc genesis.

`dig` does not consistently exit nicely (with appropriate status) when a problem occurs somewhere in the resolver (Most of the common exit cases are handled.) This is particularly annoying when running in batch mode. If it exits abnormally (and is not caught), the entire batch file aborts; when such an event is trapped, `dig` simply continues with the next query.

SEE ALSO

`named`(8), `resolver`(3), `resolver`(5), `nslookup`(8)

30 August 1990

dnsquery

`dnsquery` – Query domain name servers using resolver

SYNOPSIS

```
dnsquery [-n nameserver] [-t type] [-c class] [-r retry] [-p retry period]
[-d] [-s] [-v] host
```

DESCRIPTION

The `dnsquery` program is a general interface to nameservers via `BIND` resolver library calls. The program supports queries to the nameserver with an opcode of `QUERY`. This program is intended to be a replacement or supplement to programs like `nstest`, `nsquery`, and `nslookup`. All arguments except for `host` and `ns` are treated without case-sensitivity.

OPTIONS

`-n` The nameserver to be used in the query. Nameservers can appear as either Internet addresses of the form `w.x.y.z` or can appear as domain names. (default: as specified in `/etc/resolv.conf`)

`-t` The type of resource record of interest. Types include:

`A`	Address
`NS`	Nameserver
`CNAME`	Canonical name
`PTR`	Domain name pointer
`SOA`	Start of authority

BUGS

Queries of a class other than `IN` can have interesting results since ordinarily a nameserver only has a list of root nameservers for class `IN` resource records.

Query uses a call to `inet_addr()` to determine if the argument for the `-n` option is a valid Internet address. Unfortunately, `inet_addr()` seems to cause a segmentation fault with some (bad) addresses (for example, `1.2.3.4.5`).

AUTHOR

Bryan Beecher

10 March 1990

domainname

`domainname`—Set or print domain of current host

SYNOPSIS

```
domainname [ name ]
```

DESCRIPTION

`domainname` prints the domain name of the current host, from the `getdomainname`(3) library call. If an argument is present and the effective UID is 0, `domainname` changes the name of the host, with the `setdomainname`(2) system call. This is usually done at boot time in the `/etc/rc.local` script.

FILES

`/etc/rc.local`

SEE ALSO

`getdomainname`(3), `setdomainname`(2), `uname`(1), `uname`(2)

AUTHOR

Lars Wirzenius by substituting in `hostname.c`

Linux 0.98, 26 December 1992

dsplit

`dsplit`—Split a large file into pieces

SYNOPSIS

```
dsplit [ -size nnn ][input_file [ output_base ]]
```

DESCRIPTION

`dsplit` splits binary files into smaller chunks so that they may be placed on floppy disks.

OPTIONS

`-size nnn`	Specifies the size of each output file, in bytes. The default is `1457000`, which is enough to will a 1.44MB floppy disk.
`input_file`	Specifies the name of the file to split up. A `-` indicates standard input. The default is standard input.

`output_base` Specifies the name of the output files to be written. `dsplit` will append `000`, `001`, ..., to the `output_base`. The default is `dsplit`.

AUTHOR'S NOTES

Submitted by: David Arnstein (`arnstein@netcom.com`)

Posting number: Volume 40, Issue 51

Archive name: `dsplit/part01`

Environment: MS-DOS, UNIX

Here is a portable binary file splitting program. It reads a binary file and splits it into pieces. I use this program to put large binary files on floppy disks. For this reason, the default size of the output files is 1,457,000 bytes, which just about fills up a 1.44MB floppy disk.

Unlike other binary split programs I have seen, `dsplit` does not `malloc` a huge block of memory. `dsplit` is suitable for use under MS-DOS and other primitive operating systems.

(The program came from `gatekeeper.dec.com:/pub/usenet/comp.sources.misc/volume40/dsplit`).

Linux 1.1, 5 July 1994

du

`du`—Summarize disk usage

SYNOPSIS

```
du [-abcklsxDLS] [—all] [—total] [—count-links] [—summarize] [—bytes]
[—kilobytes] [—one-file-system] [—separate-dirs] [—dereference]
[—dereference-args] [—help] [—version] [filename...]
```

DESCRIPTION

This manual page documents the GNU version of `du`. `du` displays the amount of disk space used by each argument and for each subdirectory of directory arguments. The space is measured in 1K blocks by default, unless the environment variable `POSIXLY_CORRECT` is set, in which case 512-byte blocks are used.

OPTIONS

`-a, —all` Display counts for all files, not just directories.

`-b, —bytes` Print sizes in bytes.

`-c, —total` Write a grand total of all of the arguments after all arguments have been processed. This can be used to find out the disk usage of a directory, with some files excluded.

`-k, —kilobytes` Print sizes in kilobytes. This overrides the environment variable `POSIXLY_CORRECT`.

`-l, —count-links` Count the size of all files, even if they have appeared already in another hard link.

`-s, —summarize` Display only a total for each argument.

`-x, —one-file-system` Skip directories that are on different filesystems from the one that the argument being processed is on.

`-D, —dereference-args` Dereference symbolic links that are command-line arguments. Does not affect other symbolic links. This is helpful for finding out the disk usage of directories like `/usr/tmp` where they are symbolic links.

`-L, —dereference` Dereference symbolic links (show the disk space used by the file or directory that the link points to instead of the space used by the link).

`-S, —separate-dirs` Count the size of each directory separately, not including the sizes of subdirectories.

`—help` Print a usage message on standard output and exit successfully.

`—version` Print version information on standard output, then exit successfully.

SHELL	Optional. The SHELL variable sets the default value for the shell option, which determines which shell program is used to perform wildcard expansion in filenames, and also which is used to execute filters or external programs. The default value on UNIX systems is /bin/sh. Note: Under MS-DOS, this variable is called COMSPEC instead of SHELL.
HOME	This variable should be set to the name of your home directory. elvis looks for its initialization file there; if HOME is unset, then the initialization file will not be executed.
TAGPATH	Optional. This variable is used by the ref program, which is invoked by the shift-K, control-], and :tag commands. See ref for more information.
TMP, TEMP	These optional environment variables are only used in non-UNIX versions of elvis. They allow you to supply a directory name to be used for storing temporary files.

SEE ALSO

ctags(1), ref(1), elvprsv(1), elvrec(1)

Elvis—A Clone of Vi/Ex, the complete elvis documentation.

BUGS

There is no Lisp support. Certain other features are missing, too.

Auto-indent mode is not quite compatible with the real vi. Among other things, ^D and ^^D don't do what you might expect.

Long lines are displayed inefficiently. The real vi wraps long lines onto multiple rows of the screen, but elvis scrolls sideways.

AUTHOR

Steve Kirkendall (kirkenda@cs.pdx.edu)

Many other people have worked to port elvis to various operating systems. To see who deserves credit, run the :version command from within elvis, or look in the system-specific section of the complete documentation.

elvprsv

elvprsv—Preserve the modified version of a file after a crash

SYNOPSIS

```
elvprsv ["-why elvis died"] /tmp/filename...
elvprsv -R /tmp/filename...
```

DESCRIPTION

elvprsv preserves your edited text after elvis dies. The text can be recovered later, via the elvprsv program.

For UNIX-like systems, you should never need to run this program from the command line. It is run automatically when elvis is about to die, and it should be run (via /etc/rc) when the computer is booted. THAT'S ALL!

For non-UNIX systems such as MS-DOS or VMS, you can either use elvprsv the same way as under UNIX systems (by running it from your AUTOEXEC.BAT file), or you can run it separately with the -R flag to recover the files in one step.

If you're editing a file when elvis dies (due to a bug, system crash, power failure, and so on), then elvprsv will preserve the most recent version of your text. The preserved text is stored in a special directory; it does not overwrite your text file automatically. (If the preservation directory hasn't been set up correctly, then elvprsv will simply send you a mail message describing how to manually run elvprsv.)

elvprsv will send mail to any user whose work it preserves, if your operating system normally supports mail.

FILES

`/tmp/elv*`	The temporary file that `elvis` was using when it died.
`/usr/preserve/p*`	The text that is preserved by `elvprsv`.
`/usr/preserve/Index`	A text file which lists the names of all preserved files, and the names of the `/usr/preserve/p*` files that contain their preserved text.

BUGS

Due to the permissions on the `/usr/preserve` directory, on UNIX systems `elvprsv` must be run as superuser. This is accomplished by making the `elvprsv` executable be owned by `root` and turning on its "set user id" bit.

If you're editing a nameless buffer when `elvis` dies, then `elvprsv` will pretend that the file was named `foo`.

AUTHOR

Steve Kirkendall (`kirkenda@cs.pdx.edu`)

elvrec

`elvrec`— Recover the modified version of a file after a crash

SYNOPSIS

`elvrec [preservedfile [newfile]]`

DESCRIPTION

If you're editing a file when `elvis` dies, the system crashes, or power fails, the most recent version of your text will be preserved. The preserved text is stored in a special directory; it does not overwrite your text file automatically.

The `elvrec` program locates the preserved version of a given file, and writes it over the top of your text file—or to a new file, if you prefer. The recovered file will have nearly all of your changes.

To see a list of all recoverable files, run `elvrec` with no arguments.

NOTE

If you haven't set up a directory for file preservation, you'll have to manually run the `elvprsv` program instead of `elvrec`.

FILES

`/usr/preserve/p*`	The text that was preserved when `elvis` died.
`/usr/preserve/Index`	A text file that lists the names of all preserved files, and the names of the `/usr/preserve/p*` files that contain their preserved text.

BUGS

`elvrec` is very picky about filenames. You must tell it to recover the file using exactly the same pathname as when you were editing it. The simplest way to do this is to go into the same directory that you were editing, and invoke `elvrec` with the same filename as `elvis`. If that doesn't work, then try running `elvrec` with no arguments, to see exactly which pathname it is using for the desired file.

Due to the permissions on the `/usr/preserve` directory, on UNIX systems `elvrec` must be run as `superuser`. This is accomplished by making the `elvrec` executable be owned by `root` and setting its "set user id" bit.

If you're editing a nameless buffer when `elvis` dies, then `elvrec` will pretend that the file was named `foo`.

AUTHOR

Steve Kirkendall (kirkenda@cs.pdx.edu)

emacs

emacs—GNU project emacs

SYNOPSIS

```
emacs [ command-line switches ] [ files ... ]
```

DESCRIPTION

GNU emacs is a version of emacs, written by the author of the original (PDP-10) emacs, Richard Stallman.

The primary documentation of GNU emacs is in the *GNU Emacs Manual*, which you can read online using info, a subsystem of emacs. Please look there for complete and up-to-date documentation. This man page is updated only when someone volunteers to do so; the emacs maintainers' priority goal is to minimize the amount of time this man page takes away from other more useful projects.

The user functionality of GNU emacs encompasses everything other emacs editors do, and it is easily extensible since its editing commands are written in Lisp.

emacs has an extensive interactive help facility, but the facility assumes that you know how to manipulate emacs windows and buffers. Ctrl+h (backspace or Ctrl+h) enters the Help facility. Help Tutorial (Ctrl+h t) requests an interactive tutorial that can teach beginners the fundamentals of emacs in ten minutes. Help Apropos (Ctrl+h a) helps you find a command given its functionality, Help Character (Ctrl+h c) describes a given character's effect, and Help Function (Ctrl+h f) describes a given Lisp function specified by name.

emacs's Undo can undo several steps of modification to your buffers, so it is easy to recover from editing mistakes.

GNU emacs's many special packages handle mail reading (RMail) and sending (Mail), outline editing (Outline), compiling (Compile), running subshells within emacs windows (Shell), running a Lisp read-eval-print loop (Lisp-Interaction-Mode), and automated psychotherapy (Doctor).

There is an extensive reference manual, but users of other emacses should have little trouble adapting even without a copy. Users new to emacs will be able to use basic features fairly rapidly by studying the tutorial and using the self-documentation features.

OPTIONS

The following options are of general interest:

file	Edit file.
+number	Go to the line specified by number (do not insert a space between the + sign and the number).
-q	Do not load an init file.
-u user	Load user's init file.
-t file	Use specified file as the terminal instead of using stdin/stdout. This must be the first argument specified in the command line.

The following options are Lisp-oriented (these options are processed in the order encountered):

-f function	Execute the Lisp function function.
-l file	Load the Lisp code in the file file.

The following options are useful when running emacs as a batch editor:

-batch	Edit in batch mode. The editor will send messages to stdout. This option must be the first in the argument list. You must use -l and -f options to specify files to execute and functions to call.
-kill	Exit emacs while in batch mode.

`-v`	Verbose option forces `ftp` to show all responses from the remote server, as well as report on data transfer statistics.
`-n`	Restrains `ftp` from attempting auto-login upon initial connection. If auto-login is enabled, `ftp` will check the (see below) file in the user's home directory for an entry describing an account on the remote machine. If no entry exists, `ftp` will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login.
`-i`	Turns off interactive prompting during multiple file transfers.
`-d`	Enables debugging.
`-g`	Disables filename globbing.

The client host with which `ftp` is to communicate may be specified on the command line. If this is done, `ftp` will immediately attempt to establish a connection to an FTP server on that host; otherwise, `ftp` will enter its command interpreter and await instructions from the user. When `ftp` is awaiting commands from the user, the prompt

```
ftp>
```

is provided to the user. The following commands are recognized by `ftp`:

`! [command] [args]`	Invoke an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.
`$ macro-name [args]`	Execute the macro `macro-name` that was defined with the `macdef` command. Arguments are passed to the macro unglobbed.
`account [passwd]`	Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a nonechoing input mode.
`append local-file [remote-file]`	Append a local file to a file on the remote machine. If `remote-file` is left unspecified, the local filename is used in naming the remote file after being altered by any ntrans or nmap setting. File transfer uses the current settings for `type`, `format`, `mode`, and `structure`.
`ascii`	Set the file transfer type to network ASCII. This is the default type.
`bell`	Arrange that a bell be sounded after each file transfer command is completed.
`binary`	Set the file transfer type to support binary image transfer.
`bye`	Terminate the FTP session with the remote server and exit `ftp`. An end of file will also terminate the session and exit.
`case`	Toggle remote computer filename case mapping during `mget` commands. When case is on (default is `off`), remote computer filenames with all letters in upper case are written in the local directory with the letters mapped to lowercase.
`cd remote-directory`	Change the working directory on the remote machine to `remote-directory`.
`cdup`	Change the remote machine working directory to the parent of the current remote machine working directory.
`chmod mode file-name`	Change the permission modes of the file `file-name` on the remote system to `mode`.
`close`	Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.
`cr`	Toggle carriage return stripping during ASCII type file retrieval. Records are denoted by a carriage `return/linefeed` sequence during ASCII type file transfer. When `cr` is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ASCII type transfer is made, these linefeeds may be distinguished from a record delimiter only when `cr` is off.
`delete remote-file`	Delete the file `remote-file` on the remote machine.

`rmdir directory-name`	Delete a directory on the remote machine.
`runique`	Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a `get` or `mget` command, a `.1` is appended to the name. If the resulting name matches another existing file, a `.2` is appended to the original name. If this process continues up to `.99`, an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that `runique` will not affect local files generated from a shell command. The default value is `off`.
`send local-file [remote-file]`	A synonym for `put`.
`sendport`	Toggle the use of `PORT` commands. By default, `ftp` will attempt to use a `PORT` command when establishing a connection for each data transfer. The use of `PORT` commands can prevent delays when performing multiple file transfers. If the `PORT` command fails, `ftp` will use the default data port. When the use of `PORT` commands is disabled, no attempt will be made to use `PORT` commands for each data transfer. This is useful for certain FTP implementations which do ignore `PORT` commands but, incorrectly, indicate they've been accepted.
`site arg1 arg2...`	The arguments specified are sent, verbatim, to the remote FTP server as a `SITE` command.
`size file-name`	Return size of `file-name` on remote machine.
`status`	Show the current status of `ftp`.
`struct [struct-name]`	Set the file transfer structure to `struct-name`. By default `stream` structure is used.
`sunique`	Toggle storing of files on remote machine under unique filenames. Remote FTP server must support ftp protocol `STOU` command for successful completion. The remote server will report unique name. Default value is `off`.
`system`	Show the type of operating system running on the remote machine.
`tenex`	Set the file transfer type to that needed to talk to TENEX machines.
`trace`	Toggle packet tracing.
`type [type-name]`	Set the file transfer type to `type-name`. If no type is specified, the current type is printed. The default type is network `ASCII`.
`umask [newmask]`	Set the default umask on the remote server to `newmask`. If `newmask` is omitted, the current `umask` is printed.
`user user-name [password] [account]`	Identify yourself to the remote FTP server. If the password is not specified and the server requires it, `ftp` will prompt the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user will be prompted for it. If an account field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless `ftp` is invoked with auto-login disabled, this process is done automatically on initial connection to the FTP server.
`verbose`	Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.
`? [command]`	A synonym for `help`.

Command arguments which have embedded spaces may be quoted with quotation marks (").

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending an FTP protocol `ABOR` command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for `ABOR` processing.

If the remote server does not support the ABOR command, an ftp> prompt will not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence will be ignored when ftp has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the ABOR processing described earlier in this section, or from unexpected behavior by the remote server, including violations of the FTP protocol. If the delay results from unexpected remote server behavior, the local ftp program must be killed by hand.

FILE NAMING CONVENTIONS

Files specified as arguments to ftp commands are processed according to the following rules:

If the filename - is specified, the stdin (for reading) or stdout (for writing) is used.

If the first character of the filename is *j*, the remainder of the argument is interpreted as a shell command. ftp then forks a shell, using popen 3 with the argument supplied, and reads (writes) from the stdout (stdin). If the shell command includes spaces, the argument must be quoted; for example, ls -lt. A particularly useful example of this mechanism is: dir more.

Failing the preceding checks, if "globbing" is enabled, local filenames are expanded according to the rules used in the csh 1; c.f. the glob command. If the ftp command expects a single local file (for example, put), only the first filename generated by the "globbing" operation is used.

For mget commands and get commands with unspecified local filenames, the local filename is the remote filename, which may be altered by a case, ntrans, or nmap setting. The resulting filename may then be altered if runique is on.

For mput commands and put commands with unspecified remote filenames, the remote filename is the local filename, which may be altered by a ntrans or nmap setting. The resulting filename may then be altered by the remote server if sunique is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters that may affect a file transfer. The type may be one of ASCII, image (binary), ebcdic, and local byte size (for PDP Ns -10s and PDP Ns -20s mostly). ftp supports the ASCII and image types of file transfer, plus local byte size 8 for tenex mode transfers.

ftp supports only the default values for the remaining file transfer parameters: mode, form, and struct.

THE .netrc FILE

The file contains login and initialization information used by the auto-login process. It resides in the user's home directory. The following tokens are recognized; they may be separated by spaces, tabs, or newlines:

machine name	Identify a remote machine name. The auto-login process searches the file for a machine token that matches the remote machine specified on the ftp command line or as an open command argument. When a match is made, the subsequent tokens are processed, stopping when the end of file is reached or another machine or a default token is encountered.
default	This is the same as machine name except that default matches any name. There can be only one default token, and it must be after all machine tokens. This is normally used as default login anonymous password user@site, thereby giving the user automatic anonymous ftp login to machines not specified. This can be overridden by using the -n flag to disable auto-login.
login name	Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.
password string	Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the file for any user other than anonymous, ftp will abort the auto-login process if the is readable by anyone besides the user.
account string	Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

`-Wtemplate-debugging`	When using templates in a C++ program, warn if debugging is not yet fully available.
`-w`	Inhibit all warning messages.
`+eN`	Control how virtual function definitions are used, in a fashion compatible with `cfront 1.x`.

PRAGMAS

Two `#pragma` directives are supported for GNU C++, to permit using the same header file for two purposes: as a definition of interfaces to a given object class, and as the full definition of the contents of that object class.

`#pragma interface`	Use this directive in header files that define object classes, to save space in most of the object files that use those classes. Normally, local copies of certain information (backup copies of inline member functions, debugging information, and the internal tables that implement virtual functions) must be kept in each object file that includes class definitions. You can use this pragma to avoid such duplication. When a header file containing `#pragma interface` is compiled in a compilation, this auxiliary information will not be generated (unless the main input source file itself uses `#pragma implementation`). Instead, the object files will contain references to be resolved at link time.
`#pragma implementation` `#pragma implementation objects.h`	Use this pragma in a main input file, when you want full output from included header files to be generated (and made globally visible). The included header file, in turn, should use `#pragma interface`. Backup copies of inline member functions, debugging information, and the internal tables used to implement virtual functions are all generated in implementation files. If you use #pragma implementation with no argument, it applies to an include file with the same basename as your source file; for example, in `allclass.cc`, `#pragma implementation` by itself is equivalent to `#pragma implementation "allclass.h"`. Use the string argument if you want a single implementation file to include code from multiple header files. There is no way to split up the contents of a single header file into multiple implementation files.

FILES

`file.h`	C header (preprocessor) file
`file.i`	Preprocessed C source
`file file.C`	C++ source file
`file.cc`	C++ source file
`file.cxx`	C++ source file
`file.s`	Assembly language file
`file.o`	Object file
`a.out`	Link edited output
`TMPDIR/cc`	Temporary files
`LIBDIR/cpp`	Preprocessor
`LIBDIR/cc1plus`	Compiler
`LIBDIR/collect`	Linker front end needed on some machines
`LIBDIR/libgcc.a`	GCC subroutine library
`/lib/crt[01n].o`	Start-up routine
`LIBDIR/ccrt0`	Additional start-up routine for C++
`/lib/libc.a`	Standard C library; see `intro`(3)

When a string must be converted to a number, the conversion is accomplished using `atof`(3). A number is converted to a string by using the value of `CONVFMT` as a format string for `sprintf`(3), with the numeric value of the variable as the argument. However, even though all numbers in `awk` are floating-point, integral values are *always* converted as integers. Thus, given this:

```
CONVFMT = "%2.2f"
a =12
b =a""
```

the variable `b` has a string value of `12` and not `12.00`.

`gawk` performs comparisons as follows: If two variables are numeric, they are compared numerically. If one value is numeric and the other has a string value that is a "numeric string," then comparisons are also done numerically. Otherwise, the numeric value is converted to a string and a string comparison is performed. Two strings are compared, of course, as strings. According to the standard, even if two strings are numeric strings, a numeric comparison is performed. However, this is clearly incorrect, and `gawk` does not do this.

Uninitialized variables have the numeric value `0` and the string value `""` (the null, or empty, string).

PATTERNS AND ACTIONS

`awk` is a line-oriented language. The pattern comes first, and then the action. Action statements are enclosed in `and .BR`. Either the pattern may be missing, or the action may be missing, but, of course, not both. If the pattern is missing, the action will be executed for every single line of input. A missing action is equivalent to

```
{ print }
```

which prints the entire line.

Comments begin with the `#` character, and continue until the end of the line. Blank lines may be used to separate statements. Normally, a statement ends with a newline, however, this is not the case for lines ending in a ,, `{`, `?`, `:`, `&&`, or `||`. Lines ending in `do` or `else` also have their statements automatically continued on the following line. In other cases, a line can be continued by ending it with a `\`, in which case the newline will be ignored.

Multiple statements may be put on one line by separating them with a semicolon. This applies to both the statements within the action part of a pattern-action pair (the usual case), and to the pattern-action statements themselves.

PATTERNS

`awk` patterns may be one of the following:

```
BEGIN
END
/regular expression/
relational expression
pattern && pattern
pattern jj pattern
pattern ? pattern : pattern
(pattern)
! pattern
pattern1, pattern2
```

`BEGIN` and `END` are two special kinds of patterns that are not tested against the input. The action parts of all `BEGIN` patterns are merged as if all the statements had been written in a single `BEGIN` block. They are executed before any of the input is read. Similarly, all the `END` blocks are merged, and executed when all the input is exhausted (or when an `exit` statement is executed). `BEGIN` and `END` patterns cannot be combined with other patterns in pattern expressions. `BEGIN` and `END` patterns cannot have missing action parts.

For `/regular expression/` patterns, the associated statement is executed for each input line that matches the regular expression. Regular expressions are the same as those in `egrep`(1), and are summarized as follows:

%d	A decimal number (the integer part).
%i	Just like %d.
%e	A floating-point number of the form [-]d.dddddd[+-]dd.
%f	A floating-point number of the form [-]ddd.dddddd.
%g	Use e or f conversion, whichever is shorter, with nonsignificant zeros suppressed.
%o	An unsigned octal number (again, an integer).
%s	A character string.
%x	An unsigned hexadecimal number (an integer).
%X	Like %x, but using ABCDEF instead of abcdef.
%%	A single % character; no argument is converted.

There are optional, additional parameters that may lie between the % and the control letter:

-	The expression should be left-justified within its field.
width	The field should be padded to this width. If the number has a leading zero, then the field will be padded with zeros. Otherwise, it is padded with blanks. This applies even to the nonnumeric output formats.
.prec	A number indicating the maximum width of strings or digits to the right of the decimal point.

The dynamic width and prec capabilities of the C printf() routines are supported. A * in place of either the width or prec specifications will cause their values to be taken from the argument list to printf or sprintf().

SPECIAL FILENAMES

When doing I/O redirection from either print or printf into a file, or via getline from a file, gawk recognizes certain special filenames internally. These filenames allow access to open file descriptors inherited from gawk's parent process (usually the shell). Other special filenames provide access information about the running gawk process. The filenames are

/dev/pid	Reading this file returns the process ID of the current process, in decimal, terminated with a newline.
/dev/ppid	Reading this file returns the parent process ID of the current process, in decimal, terminated with a newline.
/dev/pgrpid	Reading this file returns the process group ID of the current process, in decimal, terminated with a newline.
/dev/user	Reading this file returns a single record terminated with a newline. The fields are separated with blanks. $1 is the value of the getuid(2) system call, $2 is the value of the geteuid(2) system call, $3 is the value of the getgid(2) system call, and $4 is the value of the getegid(2) system call. If there are any additional fields, they are the group IDs returned by getgroups(2). Multiple groups may not be supported on all systems.
/dev/stdin	The standard input.
/dev/stdout	The standard output.
/dev/stderr	The standard error output.
/dev/fd/n	The file associated with the open file descriptor n.

These are particularly useful for error messages. For example, you could use

```
print "You blew it!" > "/dev/stderr"
```

whereas you would otherwise have to use

```
print "You blew it!" j "cat 1>&2"
```

These filenames may also be used on the command line to name data files.

NUMERIC FUNCTIONS

awk has the following predefined arithmetic functions:

atan2(y, x)	Returns the arctangent of *y/x* in radians.
cos(expr)	Returns the cosine in radians.

OVERALL OPTIONS

`-x language` Specify explicitly the `language` for the following input files (rather than choosing a default based on the filename suffix). This option applies to all following input files until the next `-x` option. Possible values of `language` are `c`, `objective-c`, `c-header`, `c++`, `cpp-output`, `assembler`, and `assembler-with-cpp`.

`-x none` Turn off any specification of a language, so that subsequent files are handled according to their filename suffixes (as they are if `-x` has not been used at all).

If you want only some of the four stages (preprocess, compile, assemble, link), you can use `-x` (or filename suffixes) to tell `gcc` where to start, and one of the options `-c`, `-S`, or `-E` to say where `gcc` is to stop. Note that some combinations (for example, `-x cpp-output -E`) instruct `gcc` to do nothing at all.

`-c` Compile or assemble the source files, but do not link. The compiler output is an object file corresponding to each source file.

By default, `gcc` makes the object filename for a source file by replacing the suffix `.c`, `.i`, `.s`, and so on, with `.o`. Use `-o` to select another name.

`gcc` ignores any unrecognized input files (those that do not require compilation or assembly) with the `-c` option.

`-S` Stop after the stage of compilation proper; do not assemble. The output is an assembler code file for each nonassembler input file specified.

By default, `gcc` makes the assembler filename for a source file by replacing the suffix `.c`, `.i`, and so on, with `.s`. Use `-o` to select another name.

`gcc` ignores any input files that don't require compilation.

`-E` Stop after the preprocessing stage; do not run the compiler proper. The output is preprocessed source code, which is sent to the standard output.

`gcc` ignores input files that don't require preprocessing.

`-o file` Place output in file `file`. This applies regardless to whatever sort of output `gcc` is producing, whether it be an executable file, an object file, an assembler file, or preprocessed C code.

Since only one output file can be specified, it does not make sense to use `-o` when compiling more than one input file, unless you are producing an executable file as output.

If you do not specify `-o`, the default is to put an executable file in `a.out`, the object file for `source.suffix` in `source.o`, its assembler file in `source.s`, and all preprocessed C source on standard output.

`-v` Print (on standard error output) the commands executed to run the stages of compilation. Also print the version number of the compiler driver program and of the preprocessor and the compiler proper.

`-pipe` Use pipes rather than temporary files for communication between the various stages of compilation. This fails to work on some systems where the assembler cannot read from a pipe; but the GNU assembler has no trouble.

LANGUAGE OPTIONS

The following options control the dialect of C that the compiler accepts:

`-ansi` Support all ANSI standard C programs.

This turns off certain features of GNU C that are incompatible with ANSI C, such as the `asm`, `inline`, and `typeof` keywords, and predefined macros such as `unix` and `vax` that identify the type of system you are using. It also enables the undesirable and rarely used ANSI trigraph feature, and disallows `$` as part of identifiers. The alternate keywords `__asm__`, `__extension__`, `__inline__`, and `__typeof__` continue to work despite `-ansi`. You would not want to use them in an ANSI C program, of course, but it is useful to put them in header files that might be included in compilations done with `-ansi`. Alternate predefined macros such as `__unix__` and `__vax__` are also available, with or without `-ansi`.

The `-ansi` option does not cause non-ANSI programs to be rejected gratuitously. For that, `-pedantic` is required in addition to `-ansi`.

The preprocessor predefines a macro `__STRICT_ANSI__` when you use the `-ansi` option. Some header files may notice this macro and refrain from declaring certain functions or defining certain macros that the ANSI standard doesn't call for; this is to avoid interfering with any programs that might use these names for other things.

`-fno-asm` Do not recognize `asm`, `inline`, or `typeof` as a keyword. These words may then be used as identifiers. You can use `__asm__`, `__inline__`, and `__typeof__` instead. `-ansi` implies `-fno-asm`.

`-fno-builtin` Don't recognize built-in functions that do not begin with two leading underscores. Currently, the functions affected include `_exit`, `abort`, `abs`, `alloca`, `cos`, `exit`, `fabs`, `labs`, `memcmp`, `memcpy`, `sin`, `sqrt`, `strcmp`, `strcpy`,and `strlen`.

The `-ansi` option prevents `alloca` and `_exit` from being built-in functions.

`-fno-strict-prototype` Treat a function declaration with no arguments, such as `int foo();`, as C would treat it—as saying nothing about the number of arguments or their types (C++ only). Normally, such a declaration in C++ means that the function `foo` takes no arguments.

`-trigraphs` Support ANSI C trigraphs. The `-ansi` option implies `-trigraphs`.

`-traditional` Attempt to support some aspects of traditional C compilers. For details, see the *GNU C Manual*; the duplicate list here has been deleted so that we won't get complaints when it is out of date.

But one note about C++ programs only (not C). `-traditional` has one additional effect for C++: assignment to `this` is permitted. This is the same as the effect of `-fthis-is-variable`.

`-traditional-cpp` Attempt to support some aspects of traditional C preprocessors. This includes the items that specifically mention the preprocessor previously, but none of the other effects of `-traditional`.

`-fdollars-in-identifiers` Permit the use of `$` in identifiers (C++ only). You can also use `-fno-dollars-in-identifiers` to explicitly prohibit use of `$`. (GNU C++ allows `$` by default on some target systems but not others.)

`-fenum-int-equiv` Permit implicit conversion of `int` to enumeration types (C++ only). Normally GNU C++ allows conversion of `enum` to `int`, but not the other way around.

`-fexternal-templates` Produce smaller code for template declarations, by generating only a single copy of each template function where it is defined (C++ only). To use this option successfully, you must also mark all files that use templates with either `#pragma implementation` (the definition) or `#pragma interface` (declarations).

When your code is compiled with `-fexternal-templates`, all template instantiations are external. You must arrange for all necessary instantiations to appear in the implementation file; you can do this with a `typedef` that references each instantiation needed. Conversely, when you compile using the default option `-fno-external-templates`, all template instantiations are explicitly internal.

`-fall-virtual` Treat all possible member functions as virtual, implicitly. All member functions (except for constructor functions and `new` or `delete` member operators) are treated as virtual functions of the class where they appear. This does not mean that all calls to these member functions will be made through the internal table of virtual functions. Under some circumstances, the compiler can determine that a call to a given virtual function can be made directly; in these cases, the calls are direct in any case.

`-fcond-mismatch` Allow conditional expressions with mismatched types in the second and third arguments. The value of such an expression is void.

`-fthis-is-variable` Permit assignment to `this` (C++ only). The incorporation of user-defined free store management into C++ has made assignment to `this` an anachronism. Therefore, by default it is invalid to assign to this within a class member function. However, for backwards compatibility, you can make it valid with `-fthis-is-variable`.

`-funsigned-char` Let the type `char` be unsigned, like unsigned `char`.

Each kind of machine has a default for what `char` should be. It is either like `unsigned char` by default or like `signed char` by default.

The directories searched include several standard system directories plus any that you specify with `-L`.

Normally, the files found this way are library files—archive files whose members are object files. The linker handles an archive file by scanning through it for members that define symbols that have so far been referenced but not defined. However, if the linker finds an ordinary object file rather than a library, the object file is linked in the usual fashion. The only difference between using an `-l` option and specifying a filename is that –l surrounds `library` with `lib` and `.a` and searches several directories.

`-lobjc` You need this special case of the `-l` option in order to link an Objective C program.

`-nostartfiles` Do not use the standard system startup files when linking. The standard libraries are used normally.

`-nostdlib` Don't use the standard system libraries and startup files when linking. Only the files you specify will be passed to the linker.

`-static` On systems that support dynamic linking, this prevents linking with the shared libraries. On other systems, this option has no effect.

`-shared` Produce a shared object which can then be linked with other objects to form an executable. Only a few systems support this option.

`-symbolic` Bind references to global symbols when building a shared object. Warn about any unresolved references (unless overridden by the link editor option `-Xlinker -z -Xlinker defs`). Only a few systems support this option.

`-Xlinker option` Pass `option` as an option to the linker. You can use this to supply system-specific linker options which GNU CC does not know how to recognize.

If you want to pass an option that takes an argument, you must use `-Xlinker` twice: once for the option and once for the argument. For example, to pass `-assert definitions`, you must write `-Xlinker -assert -Xlinker definitions`. It does not work to write `-Xlinker "-assert definitions"`, because this passes the entire string as a single argument, which is not what the linker expects.

`-Wl,option` Pass `option` as an option to the linker. If `option` contains commas, it is split into multiple options at the commas.

`-u symbol` Pretend the symbol `symbol` is undefined, to force linking of library modules to define it. You can use `-u` multiple times with different symbols to force loading of additional library modules.

DIRECTORY OPTIONS

These options specify directories to search for header files, for libraries, and for parts of the compiler:

`-Idir` Append directory `dir` to the list of directories searched for include files.

`-I-` Any directories you specify with `-I` options before the `-I-` option are searched only for the case of `#include "file"`; they are not searched for `#include <file>`.

If additional directories are specified with `-I` options after the `-I-`, these directories are searched for all `#include` directives. (Ordinarily *all* `-I` directories are used this way.)

In addition, the `-I-` option inhibits the use of the current directory (where the current input file came from) as the first search directory for `#include " file "`. There is no way to override this effect of `-I-`. With `-I.` you can specify searching the directory that was current when the compiler was invoked. That is not exactly the same as what the preprocessor does by default, but it is often satisfactory.

`-I-` does not inhibit the use of the standard system directories for header files. Thus, `-I-` and `-nostdinc` are independent.

`-Ldir` Add directory `dir` to the list of directories to be searched for `-l`.

`-Bprefix` This option specifies where to find the executables, libraries, and data files of the compiler itself.

The compiler driver program runs one or more of the subprograms `cpp`, `cc1` (or, for C++, `cc1plus`), `as`, and `ld`. It tries `prefix` as a prefix for each program it tries to run, both with and without `machine / version /`.

For each subprogram to be run, the compiler driver first tries the `-B` prefix, if any. If that name is not found, or if `-B` was not specified, the driver tries two standard prefixes, which are `/usr/lib/gcc/` and `/usr/local/lib/gcc-lib/`. If neither of those results in a filename that is found, the compiler driver searches for the unmodified program name, using the directories specified in your `PATH` environment variable.

The runtime support file `libgcc.a` is also searched for using the `-B` prefix, if needed. If it is not found there, the two standard prefixes (preceding paragraph) are tried, and that is all. The file is left out of the link if it is not found by those means. Most of the time, on most machines, `libgcc.a` is not actually necessary.

You can get a similar result from the environment variable `GCC_EXEC_PREFIX`; if it is defined, its value is used as a prefix in the same way. If both the `-B` option and the `GCC_EXEC_PREFIX` variable are present, the `-B` option is used first and the environment variable value second.

WARNING OPTIONS

Warnings are diagnostic messages that report constructions that are not inherently erroneous but are risky or suggest there may have been an error.

These options control the amount and kinds of warnings produced by GNU CC:

`-fsyntax-only` Check the code for syntax errors, but don't emit any output.

`-w` Inhibit all warning messages.

`-Wno-import` Inhibit warning messages about the use of `#import`.

`-pedantic` Issue all the warnings demanded by strict ANSI standard C; reject all programs that use forbidden extensions.

Valid ANSI standard C programs should compile properly with or without this option (though a rare few will require `-ansi`). However, without this option, certain GNU extensions and traditional C features are supported as well. With this option, they are rejected. There is no reason to use this option; it exists only to satisfy pedants.

`-pedantic` does not cause warning messages for use of the alternate keywords whose names begin and end with '__'. Pedantic warnings are also disabled in the expression that follows `extension`. However, only system header files should use these escape routes; application programs should avoid them.

`-pedantic-errors` Like `-pedantic`, except that errors are produced rather than warnings.

`-W` Print extra warning messages for these events:

A nonvolatile automatic variable might be changed by a call to `longjmp`. These warnings are possible only in optimizing compilation. The compiler sees only the calls to `setjmp`. It cannot know where `longjmp` will be called; in fact, a signal handler could call it at any point in the code. As a result, you may get a warning even when there is in fact no problem because `longjmp` cannot in fact be called at the place which would cause a problem.

A function can return either with or without a value. (Falling off the end of the function body is considered returning without a value.) For example, this function would evoke such a warning:

```
foo (a)
{
if (a > 0)
return a;
}
```

Spurious warnings can occur because GNU CC does not realize that certain functions (including `abort` and `longjmp`) will never return.

An expression-statement or the left side of a comma expression contains no side effects. To suppress the warning, cast the unused expression to void. For example, an expression such as `x[i,j]` will cause a warning, but `x[(void)i,j]` will not.

An unsigned value is compared against zero with `>` or `<=`.

`-mkernel` Generate code which is suitable for use in kernels. Specifically, avoid `add` instructions in which one of the arguments is the DP register; generate `addil` instructions instead. This avoids a rather serious bug in the HP-UX linker.

`-mshared-libs` Generate code that can be linked against HP-UX shared libraries. This option is not fully functioning yet, and is not on by default for any PA target. Using this option can cause incorrect code to be generated by the compiler.

`-mno-shared-libs` Don't generate code that will be linked against shared libraries. This is the default for all PA targets.

`-mlong-calls` Generate code which allows calls to functions greater than 256K away from the caller when the caller and callee are in the same source file. Do not turn this option on unless code refuses to link with branch out of range errors from the linker.

`-mdisable-fpregs` Prevent floating-point registers from being used in any manner. This is necessary for compiling kernels that perform lazy context switching of floating-point registers. If you use this option and attempt to perform floating-point operations, the compiler will abort.

`-mdisable-indexing` Prevent the compiler from using indexing address modes. This avoids some rather obscure problems when compiling MIG-generated code under MACH.

`-mtrailing-colon` Add a colon to the end of label definitions (for ELF assemblers).

These `-m` options are defined for the Intel 80960 family of computers:

`-mcpu-type` Assume the defaults for the machine type *cpu-type* for instruction and addressing-mode availability and alignment. The default *cpu-type* is `kb`; other choices are `ka`, `mc`, `ca`, `cf`, `sa`, and `sb`.

`-mnumerics` The `-mnumerics` option indicates that the processor does support floating-point instructions.

`-msoft-float` The `-msoft-float` option indicates that floating-point support should not be assumed.

`-mleaf-procedures` Do (or do not) attempt to alter leaf procedures to be callable with the `bal` instruction as well as `call`. This will result in more efficient code for explicit calls when the `bal` instruction can be substituted by the assembler or linker, but less efficient code in other cases, such as calls via function pointers, or using a linker that doesn't support this optimization.

`-mno-leaf-procedures`

`-mtail-call` Do (or do not) make additional attempts (beyond those of the machine-independent portions of the compiler) to optimize tail-recursive calls into branches. You may not want to do this because the detection of cases where this is not valid is not totally complete. The default is `-mno-tail-call`.

`-mno-tail-call`

`-mcomplex-addr` Assume (or do not assume) that the use of a complex addressing mode is a win on this implementation of the i960. Complex addressing modes may not be worthwhile on the K-series, but they definitely are on the C-series. The default is currently `-mcomplex-addr` for all processors except the CB and CC.

`-mno-complex-addr`

`-mcode-align` Align code to 8-byte boundaries for faster fetching (or don't bother). Currently turned on by default for C-series implementations only.

`-mno-code-align`

`-mic-compat` Enable compatibility with iC960 v2.0 or v3.0.

`-mic2.0-compat`

`-mic3.0-compat`

`-masm-compat` Enable compatibility with the iC960 assembler.

`-mintel-asm`

`-mstrict-align` Do not permit (do permit) unaligned accesses.

`-mno-strict-align`

`-mold-align` Enable structure-alignment compatibility with Intel's `gcc` release version 1.3 (based on `gcc` 1.37). Currently this is buggy in that `#pragma align 1` is always assumed as well, and cannot be turned off.

```
e1 && e2
e1 || e2
e1 == e2
e1 != e2
e1 >= e2
e1 > e2
e1 <= e2
e1 < e2
"str1"=="str2"
"str1"!="str2"
```

String comparison expressions must be parenthesized in some contexts to avoid ambiguity.

OTHER CHANGES

A bare expression, `expr`, is acceptable as an attribute; it is equivalent to `dir expr`, where `dir` is the current direction. For example

```
line 2i
```

means draw a line 2 inches long in the current direction.

The maximum width and height of the picture are taken from the variables `maxpswid` and `maxpsht`. Initially, these have values 8.5 and 11.

Scientific notation is allowed for numbers. For example

```
x = 5e-2
```

Text attributes can be compounded. For example

```
"foo" above ljust
```

is legal.

There is no limit to the depth to which blocks can be examined. For example

```
[A: [B: [C: box ]]] with .A.B.C.sw at 1,2
circle at last [].A.B.C
```

is acceptable.

Arcs now have compass points determined by the circle of which the arc is a part.

Circles and arcs can be dotted or dashed. In mode, splines can be dotted or dashed.

Boxes can have rounded corners. The `rad` attribute specifies the radius of the quarter-circles at each corner. If no `rad` or `diam` attribute is given, a radius of `boxrad` is used. Initially, `boxrad` has a value of `0`. A box with rounded corners can be dotted or dashed.

The `.PS` line can have a second argument specifying a maximum height for the picture. If the width of zero is specified, the width will be ignored in computing the scaling factor for the picture. Note that GNU `pic` will always scale a picture by the same amount vertically as horizontally. This is different from the DWB 2.0 `pic`, which may scale a picture by a different amount vertically than horizontally if a height is specified.

Each text object has an invisible box associated with it. The compass points of a text object are determined by this box. The implicit motion associated with the object is also determined by this box. The dimensions of this box are taken from the width and height attributes; if the width attribute is not supplied, then the width will be taken to be `textwid`; if the height attribute is not supplied, then the height will be taken to be the number of text strings associated with the object times `textht`. Initially `textwid` and `textht` have a value of `0`.

In places where a quoted text string can be used, an expression of the form:

```
sprintf(format,arg,...)
```

space); the resulting label will also be a two-part label with the same first part as before merging, and so additional labels can be merged into it. Note that it is permissible for the first part to be empty; this may be desirable for expressions used in the `short-label` command.

`(expr)` The same as `expr`. Used for grouping.

The preceding expressions are listed in order of precedence (highest first); `&` and `¦` have the same precedence.

MACRO INTERFACE

Each reference starts with a call to the macro `]-`. The string `[F` will be defined to be the label for this reference, unless the `no-label-in-reference` command has been given. There then follows a series of string definitions, one for each field: string `[X` corresponds to field `X`. The number register `[P` is set to `1` if the `P` field contains a range of pages. The `[T`, `[A`, and `[O` number registers are set to `1` according as the `T`, `A`, and `O` fields end with one of the characters `.?!`. The `[E` number register will be set to 1 if the `[E` string contains more than one name. The reference is followed by a call to the `][` macro. The first argument to this macro gives a number representing the type of the reference. If a reference contains a `J` field, it will be classified as type 1; otherwise, if it contains a `B` field, it will be type 3; otherwise, if it contains a G or R field it will be type 4; otherwise if it contains an `I` field, it will be type 2; otherwise, it will be type 0. The second argument is a symbolic name for the type: `other`, `journal-article`, `book`, `article-in-book`, or `tech-report`. Groups of references that have been accumulated, as are produced by the bibliography command are preceded by a call to the `]<` macro and followed by a call to the `]>` macro.

FILES

`/usr/dict/papers/Ind` Default database

`file.i` Index file

SEE ALSO

`gindxbib`(1), `glookbib`(1), `lkbib`(1)

BUGS

In label expressions, `<>` expressions are ignored inside `.char` expressions.

Groff Version 1.09

grep, egrep, fgrep

`grep`, `egrep`, `fgrep`—Print lines matching a pattern

SYNOPSIS

```
grep [ -[[AB]]num ][-[CEFGVBchilnsvwx]][-e ] pattern j -ffile ][files... ]
```

DESCRIPTION

`grep` searches the named input `files` (or standard input if no files are named, or the filename `-` is given) for lines containing a match to the given `pattern`. By default, `grep` prints the matching lines.

There are three major variants of `grep`, controlled by the following options:

`-G` Interpret `pattern` as a basic regular expression (see the list following this one). This is the default.

`-E` Interpret `pattern` as an extended regular expression.

`-F` Interpret `pattern` as a list of fixed strings, separated by newlines, any of which is to be matched.

In addition, two variant programs, `egrep` and `fgrep`, are available. `egrep` is similar (but not identical) to `grepn-E`, and is compatible with the historical UNIX `egrep`. `Fgrep` is the same as `grepn-F`.

All variants of `grep` understand the following options:

The caret and the dollar sign are meta characters that respectively match the empty string at the beginning and end of a line. The symbols \< and \>, respectively, match the empty string at the beginning and end of a word. The symbol \b matches the empty string at the edge of a word, and \B matches the empty string provided it's *not* at the edge of a word.

A regular expression matching a single character may be followed by one of several repetition operators:

?	The preceding item is optional and matched at most once.
*	The preceding item will be matched zero or more times.
+	The preceding item will be matched one or more times.
n	The preceding item is matched exactly n times.
n,	The preceding item is matched n or more times.
,m	The preceding item is optional and is matched at most m times.
n,m	The preceding item is matched at least n times, but not more than m times.

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated subexpressions.

Two regular expressions may be joined by the infix operator |; the resulting regular expression matches any string matching either subexpression.

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole subexpression may be enclosed in parentheses to override these precedence rules.

The back reference \n, where n is a single digit, matches the substring previously matched by the nth parenthesized subexpression of the regular expression.

In basic regular expressions, the meta characters ?, +, (, and) lose their special meaning; instead use the backslashed versions \?, \+, \(, and \).

In egrep, the meta character { loses its special meaning; instead use \{.

DIAGNOSTICS

Normally, exit status is 0 if matches were found, and 1 if no matches were found. (The .B -v option inverts the sense of the exit status.) Exit status is 2 if there were syntax errors in the pattern, inaccessible input files, or other system errors.

BUGS

E-mail bug reports to bug-gnu-utils@prep.ai.mit.edu. Be sure to include the word grep somewhere in the "Subject:" field.

Large repetition counts in the m ,n construct may cause grep to use lots of memory. In addition, certain other obscure regular expressions require exponential time and space, and may cause grep to run out of memory.

Back references are very slow, and may require exponential time.

GNU Project, 10 September 1992

grephistory

grephistory—Display filenames from Usenet history file

SYNOPSIS

```
grephistory [ -f filename ][-e ][-n ][-q ][-l ][-i ][-s ][messageid ]
```

DESCRIPTION

grephistory queries the dbz(3) index into the history(5) file for an article having a specified Message ID.

If messageid cannot be found in the database, the program prints "Not found" and exits with a nonzero status. If messageid is in the database, the program prints the pathname and exits successfully. If no pathname exists, the program will print /dev/

In troff, the \N escape sequence can be used to access characters by their position in the corresponding tfm file; all characters in the tfm file can be accessed this way.

OPTIONS

-d Do not use tpic specials to implement drawing commands. Horizontal and vertical lines will be implemented by rules. Other drawing commands will be ignored.

-v Print the version number.

-wn Set the default line thickness to n thousandths of an em.

-Fdir Search directory dir/devdvi for font and device description files.

FILES

/usr/lib/groff/font/devdvi/DESC Device description file

/usr/lib/groff/font/devdvi/ F Font description file for font F

/usr/lib/groff/tmac/tmac.dvi Macros for use with grodvi

BUGS

dvi files produced by grodvi use a different resolution (57.81 units per inch) than those produced by TeX. Incorrectly written drivers that assume the resolution used by TeX, rather than using the resolution specified in the dvi file, will not work with grodvi.

When using the -d option with boxed tables, vertical and horizontal lines can sometimes protrude by one pixel. This is a consequence of the way TeX requires that the heights and widths of rules be rounded.

SEE ALSO

tfmtodit(1), groff(1), gtroff(1), geqn(1), groff_out(5), groff_font(5), groff_char(7)

Groff Version 1.09 14

groff

groff—Front end for the groff document formatting system

SYNOPSIS

```
groff [ -tpeszaivhblCENRVXZ][-wname ][-Wname ][-mname ][-Fdir ][-Tdev ] [ -ffam ][-Mdir ][-dcs ][-rcn ][-nnum ]
[-olist ][-Parg ][files ... ]
```

DESCRIPTION

groff is a front-end to the groff document formatting system. Normally, it runs the gtroff program and a postprocessor appropriate for the selected device. Available devices are

ps For PostScript printers and previewers

dvi For TeX dvi format

X75 For a 75 dpi X11 previewer

X100 For a 100dpi X11 previewer

ascii For typewriter-like devices

latin1 For typewriter-like devices using the ISO Latin-1 character set.

The postprocessor to be used for a device is specified by the postpro command in the device description file. This can be overridden with the -X option.

The default device is ps. It can optionally preprocess with any of gpic, geqn, gtbl, grefer, or gsoelim.

BUGS

Report bugs to bug-groff@prep.ai.mit.edu. Include a complete, self-contained example that will allow the bug to be reproduced, and say which version of groff you are using.

COPYRIGHT

AVAILABILITY

The most recent released version of groff is always available for anonymous ftp from prep.ai.mit.edu (18.71.0.38) in the directory pub/gnu.

SEE ALSO

grog(1), gtroff(1), gtbl(1), gpic(1), geqn(1), gsoelim(1), grefer(1), glookbib(1), grodvi(1), grotty(1), gxditview(1), groff_font(5), groff_out(5), groff_ms(7), groff_me(7), groff_char(7)

Groff Version 1.09, 29 October 1992

grog

grog—Guess options for groff command

SYNOPSIS

```
grog [ -option ...][files ... ]
```

DESCRIPTION

grog reads files and guesses which of the groff(1) options -e, -man, -me, -mm, -ms, -p, -s,and -t are required for printing files, and prints the groff command including those options on the standard output. A filename of - is taken to refer to the standard input. If no files are specified, the standard input will be read. Any specified options will be included in the printed command. No space is allowed between options and their arguments. For example,

```
'grog -Tdvi paper.ms'
```

will guess the appropriate command to print paper.ms and then run it after adding the -Tdvi option.

SEE ALSO

doctype(1), groff(1), gtroff(1), gtbl(1), gpic(1), geqn(1), gsoelim(1)

Groff Version 1.09 14

grops

grops—PostScript driver for groff

SYNOPSIS

```
grops [ -glv ][-bn ][-cn ][-wn ][-Fdir ][files ... ]
```

HNB	Helvetica-Narrow-Bold
HNBI	Helvetica-Narrow-BoldOblique
NR	NewCenturySchlbk-Roman
NI	NewCenturySchlbk-Italic
NB	NewCenturySchlbk-Bold
NBI	NewCenturySchlbk-BoldItalic
PR	Palatino-Roman
PI	Palatino-Italic
PB	Palatino-Bold
PBI	Palatino-BoldItalic
TR	Times-Roman
TI	Times-Italic
TB	Times-Bold
TBI	Times-BoldItalic

There is also the following font which is not a member of a family:

ZCMI	ZapfChancery-MediumItalic

There are also some special fonts called S and S. ZapfDingbats is available as ZD and a reversed version of ZapfDingbats (with symbols pointing in the opposite direction) is available as ZDR; most characters in these fonts are unnamed and must be accessed using N

grops understands various X commands produced using the \X escape sequence; grops will only interpret commands that begin with a ps: tag.

\X'ps:execcode'
: This executes the arbitrary PostScript commands in code. The PostScript currentpoint will be set to the position of the \nX command before executing code. The origin will be at the top left corner of the page, and y coordinates will increase down the page. A procedure u will be defined that converts groff units to the coordinate system in effect. For example,

```
\X'ps: exec \nx u 0 rlineto stroke'
```

will draw a horizontal line one inch long. code may make changes to the graphics state, but any changes will persist only to the end of the page. A dictionary containing the definitions specified by def and mdef will be on top of the dictionary stack. If your code adds definitions to this dictionary, you should allocate space for them using \X'psmdefn'. Any definitions will persist only until the end of the page. If you use the \Y escape sequence with an argument that names a macro, code can extend over multiple lines. For example,

```
.nr x 1i
.de y
ps: exec
\nx u 0 rlineto
stroke
..
\Yy
```

is another way to draw a horizontal line one inch long.

\X'ps:filename'
: This is the same as the exec command except that the PostScript code is read from file name.

\X'ps:defcode'
: Place a PostScript definition contained in code in the prologue. There should be at most one definition per \X command. Long definitions can be split over several \X commands; all the code arguments are simply joined together separated by newlines. The definitions are placed in a dictionary which is automatically pushed on the dictionary stack when an exec command is executed. If you use the \Y escape sequence with an argument that names a macro, code can extend over multiple lines.

```
.di
.x
```

will print `b`; if `trnt` is used instead of `tr`, it will print `a`.

`.troff` Make the `n` built-in condition false, and the `t` built-in condition true. This undoes the effect of the `nroff` request.

`.vptn` Enable vertical position traps if `n` is nonzero, disable them otherwise. Vertical position traps are traps set by the `wh` or `dt` requests. Traps set by the `it` request are not vertical position traps. The parameter that controls whether vertical position traps are enabled is global. Initially, vertical position traps are enabled.

`.warnn` Control warnings. `n` is the sum of the numbers associated with each warning that is to be enabled; all other warnings will be disabled. The number associated with each warning is listed in the "Warnings" subsection. For example, `.warn 0` will disable all warnings, and `.warn 1` will disable all warnings except that about missing characters. If `n` is not given, all warnings will be enabled.

`.whilecanything` While condition `c` is true, accept `anything` as input; `c` can be any conditional acceptable to an `if` request; `anything` can comprise multiple lines if the first line starts with `{` and the last line ends with `\}`. See also the `break` and `continue` requests.

`.writestreamanything` Write `anything` to the stream named `stream`. `stream` must previously have been the subject of an open request. `anything` is read in copy mode; a leading will be stripped.

EXTENDED REQUESTS

`.cffilename` When used in a diversion, this will embed in the diversion an object which, when reread, will cause the contents of `filename` to be transparently copied through to the output. In UNIX `troff`, the contents of `filename` are immediately copied through to the output regardless of whether there is a current diversion; this behavior is so anomalous that it must be considered a bug.

`.evxx` If `xx` is not a number, this will switch to a named environment called `xx`. The environment should be popped with a matching `ev` request without any arguments, just as for numbered environments. There is no limit on the number of named environments; they will be created the first time that they are referenced.

`.fpnf1f2` The `fp` request has an optional third argument. This argument gives the external name of the font, which is used for finding the font description file. The second argument gives the internal name of the font, which is used to refer to the font in `troff` after it has been mounted. If there is no third argument, then the internal name will be used as the external name. This feature allows you to use fonts with long names in compatibility mode.

`.ssmn` When two arguments are given to the `ss` request, the second argument gives the sentence space size. If the second argument is not given, the sentence space size will be the same as the word space size. Like the word space size, the sentence space is in units of one twelfth of the `spacewidth` parameter for the current font. Initially, both the word space size and the sentence space size are 12. The sentence space size is used in two circumstances: If the end of a sentence occurs at the end of a line in fill mode, then both an interword space and a sentence space will be added; if two spaces follow the end of a sentence in the middle of a line, then the second space will be a sentence space. Note that the behavior of UNIX `troff` will be exactly that exhibited by GNU `troff` if a second argument is never given to the `ss` request. In GNU `troff`, as in UNIX `troff`, you should always follow a sentence with either a newline or two spaces.

`.tan1n2...nnTr1r2...rn` Set tabs at positions `n1, n2,...,nn` and then set tabs at `nn+r1, nn+r2,...., nn+rn` and then at `nn+rn+r1, nn+rn+r2,..., nn+rn+rn`, and so on. For example, `.ta T .5i` will set tabs every half an inch.

DESCRIPTION

The `gzexe` utility enables you to compress executables in place and have them automatically uncompress and execute when you run them (at a penalty in performance). For example if you execute `gzexe /bin/cat`, it will create the following two files:

```
-r-xr-xr-x 1 root bin 9644 Feb 11 11:16 /bin/cat
-r-xr-xr-x 1 bin bin 24576 Nov 23 13:21 /bin/cat~
```

`/bin/cat~` is the original file and `/bin/cat` is the self-uncompressing executable file. You can remove `/bin/cat~` when you are sure that `/bin/cat` works properly.

This utility is most useful on systems with very small disks.

OPTIONS

`-d` Decompress the given executables instead of compressing them

SEE ALSO

`gzip`(1), `znew`(1), `zmore`(1), `zcmp`(1), `zforce`(1)

CAVEATS

The compressed executable is a shell script. This may cause some security holes. In particular, the compressed executable relies on the `PATH` environment variable to find `gzip` and some other utilities (`cat`, `chmod`, `ln`, `sleep`).

BUGS

`gzexe` attempts to retain the original file attributes on the compressed executable, but you may have to fix them manually in some cases, using `chmod` or `chown`.

head

`head`—Output the first part of files

SYNOPSIS

```
head [-c N[bkm]] [-n N] [-qv] [--bytes=N[bkm]] [--lines=N] [--quiet] [--silent]
[--verbose] [--help] [--version] [file...]

head [-Nbcklmqv] [file...]
```

DESCRIPTION

This manual page documents the GNU version of `head`. `head` prints the first part (10 lines by default) of each given file; it reads from standard input if no files are given or when a filename of `-` is encountered. If more than one file is given, it prints a header consisting of the file's name enclosed in `==>` and `<==` before the output for each file.

OPTIONS

`head` accepts two option formats: the new one, in which numbers are arguments to the option letters; and the old one, in which the number precedes any option letters.

`-c N`, `--bytes N` Print first `N` bytes. `N` is a nonzero integer, optionally followed by one of the following characters to specify a different unit.

- `b` 512-byte blocks.
- `k` 1-kilobyte blocks.
- `m` 1-megabyte blocks.

`-l`, `-n N`, `--lines N` Print first `N` lines.

`-q`, `--quiet`, `--silent` Never print filename headers.

The option `-t` allows you to specify a particular type of information to be looked up. The arguments are defined in the man page for `named`. Currently supported types are `a`, `ns`, `md`, `mf`, `cname`, `soa`, `mb`, `mg`, `mr`, `null`, `wks`, `ptr`, `hinfo`, `minfo`, `mx`, `uinfo`, `uid`, `gid`, `unspec`, and the wildcard, which may be written as either `any` or `*`. Types must be given in lowercase. Note that the default is to look first for `a`, and then `mx`, except that if the verbose option is turned on, the default is only `a`.

The option `-a` (for "all") is equivalent to `-v -t any`.

The option `-l` causes a listing of a complete domain. For example,

```
host -l rutgers.edu
```

will give a listing of all hosts in the `rutgers.edu` domain. The `-t` option is used to filter what information is presented, as you would expect. The default is address information, which also include PTR and NS records. The command host:

```
-l -v -t any rutgers.edu
```

will give a complete download of the zone data for `rutgers.edu`, in the official master file format. (However the SOA record is listed twice, for arcane reasons.)

NOTE

`-l` is implemented by doing a complete zone transfer and then filtering out the information you have asked for. This command should be used only if it is absolutely necessary.

CUSTOMIZING HOSTNAME LOOKUP

In general, if the name supplied by the user does not have any dots in it, a default domain is appended to the end. This domain can be defined in `/etc/resolv.conf`, but is normally derived by taking the local hostname after its first dot. The user can override this, and specify a different default domain, using the environment variable `LOCALDOMAIN`. In addition, the user can supply his own abbreviations for hostnames. They should be in a file consisting of one line per abbreviation. Each line contains an abbreviation, a space, and then the full hostname. This file must be pointed to by an environment variable `HOSTALIASES`, which is the name of the file.

SEE ALSO

`named`(8)

BUGS

Unexpected effects can happen when you type a name that is not part of the local domain. Please always keep in mind that the local domain name is tacked onto the end of every name, unless it ends in a dot. Only if this fails is the name used unchanged.

The `-l` option only tries the first name server listed for the domain that you have requested. If this server is dead, you may need to specify a server manually. For example, to get a listing of `foo.edu`, you could try `host -t ns foo.edu` to get a list of all the name servers for `foo.edu`, and then try `host -l foo.edu xxx` for all `xxx` on the list of name servers, until you find one that works.

hostid

`hostid`—Set or print system's host ID.

SYNTAX

```
hostid [-v] [ decimal-id ]
```

OPTIONS

The following command-line options may be passed to `imake`:

`-Ddefine` This option is passed directly to `cpp`. It is typically used to set directory-specific variables. For example, the X Window System uses this flag to set `TOPDIR` to the name of the directory containing the top of the core distribution and `CURDIR` to the name of the current directory, relative to the top.

`-Idirectory` This option is passed directly to `cpp`. It is typically used to indicate the directory in which the `imake` template and configuration files may be found.

`-Ttemplate` This option specifies the name of the master template file (which is usually located in the directory specified with `-I`) used by `cpp`. The default is `Imake.tmpl`.

`-f filename` This option specifies the name of the per-directory input file. The default is `Imakefile`.

`-C filename` This option specifies the name of the `.c` file that is constructed in the current directory. The default is `Imakefile.c`.

`-s filename` This option specifies the name of the `make` description file to be generated but `make` should not be invoked. If the `filename` is a hyphen (-), the output is written to `stdout`. The default is to generate, but not execute, a `Makefile`.

`-e` This option indicates the `imake` should execute the generated `Makefile`. The default is to leave this to the user.

`-v` This option indicates that `imake` should print the `cpp` command line that it is using to generate the `Makefile`.

HOW IT WORKS

`Imake` invokes `cpp` with any `-I` or `-D` flags passed on the command line and passes the name of a file containing the following three lines:

```
#define IMAKE_TEMPLATE "Imake.tmpl"
#define INCLUDE_IMAKEFILE <Imakefile>
#include IMAKE_TEMPLATE
```

where `Imake.tmpl` and `Imakefile` may be overridden by the `-T` and `-f` command options, respectively.

The `IMAKE_TEMPLATE` typically reads in a file containing machine-dependent parameters (specified as `cpp` symbols), a site-specific parameters file, a file defining variables, a file containing `cpp` macro functions for generating `make` rules, and finally the `Imakefile` (specified by `INCLUDE_IMAKEFILE`) in the current directory. The `Imakefile` uses the macro functions to indicate what targets should be built; `imake` takes care of generating the appropriate rules.

`Imake` configuration files contain two types of variables, `imake` variables and `make` variables. The `imake` variables are interpreted by `cpp` when `imake` is run. By convention they are mixed case. The make variables are written into the `Makefile` for later interpretation by `make`. By convention `make` variables are uppercase.

The rules file (usually named `Imake.rules` in the configuration directory) contains a variety of `cpp` macro functions that are configured according to the current platform. `Imake` replaces any occurrences of the string `@@` with a newline to allow macros that generate more than one line of `make` rules. For example, when called with `program_target(foo, foo1.o foo2.o)`, the macro:

```
#define program_target(program, objlist) @@\
program: objlist @@\
$(CC) -o $@ objlist $(LDFLAGS)
```

will expand to

```
foo: foo1.o foo2.o
$(CC) -o $@ foo1.o foo2.o $(LDFLAGS)
```

`imake` also replaces any occurrences of the word `XCOMM` with the character `#` to permit placing comments in the `Makefile` without causing `invalid directive` errors from the preprocessor.

DESCRIPTION

Inews reads a Usenet news article (perhaps with headers) from the named file or standard input if no file is given. It adds some headers and performs some consistency checks. If the article does not meet these checks (for example, too much quoting of old articles, or posting to nonexistent newsgroups), then the article is rejected. If it passes the checks, inews sends the article to the local news server as specified in the inn.conf(5) file for distribution.

In the standard mode of operation, the input consists of the article headers, a blank line, and the message body. For compatibility with older software, the -h flag must be used. If there are no headers in the message, then this flag may be omitted.

Several headers may be specified on the command line, shown in the synopsis above as header flags. Each of these flags takes a single parameter; if the value is more than one word (for example, almost all Subject lines) then quotes must be used to prevent the shell from splitting it into multiple words. The options, and their equivalent headers, are as follows:

a	Approved
c	Control
d	Distribution
e	Expires
f	From
w	Followup-To
n	Newsgroups
r	Reply-To
t	Subject
F	References
o	Organization
x	Path prefix

The Path header is built according to the following rules. If the –x flag is used, then its value will be the start of the header. Any other host will see the site in the header, and therefore not offer the article to that site. If the pathhost configuration parameter is specified in the inn.conf(5) file, then it will be added to the Path. Otherwise, if the server configuration parameter is specified, then the full domain name of the local host will be added to the Path. The Path will always end not-for-mail.

The default Organization header will be provided if none is present in the article or if the -o flag is not used. To prevent adding the default, use the -O flag.

As a debugging aide, if the -D flag is used, the consistency checks will be performed, and the article will be sent to the standard output, rather then sent to the server.

For compatibility with C News, inews accepts, but ignores, the -A, -V, and -W flags. The C News -N flag is treated as the -D flag.

If a file named .signature exists in the user's home directory, inews will try to append it to the end of the article. If the file cannot be read, or if it is too long (for example, more than four lines or one standard I/O buffer), or if some other problem occurs, then the article will not be posted. To suppress this action, use the -S flag.

If the -R flag is used then inews will reject any attempts to post control messages.

If an unapproved posting is made to a moderated newsgroup, inews will try to mail the article to the moderator for posting. It uses the moderators(5) file to determine the mailing address. If no address is found, it will use the inn.conf file to determine a "last-chance" host to try.

If the NNTP server needs to authenticate the client, inews will use the NNTPsendpass-word(3) routine to authenticate itself. In order to do this, the program will need read access to the passwd.nntp(5) file. This is typically done by having the file group-readable and making inews run setgid to that group.

Inews exits with a zero status if the article was successfully posted or mailed, or with a nonzero status if the article could not be delivered.

When in the -a mode, ispell will also accept lines of single words prefixed with any of the following: *, &, @, +, -, ~, #, !, %, or ^. A line starting with * tells ispell to insert the word into the user's dictionary (similar to the I command). A line starting with & tells ispell to insert an all-lowercase version of the word into the user's dictionary (similar to the U command). A line starting with @ causes ispell to accept this word in the future (similar to the A command). A line starting with +, followed immediately by tex or nroff, will cause ispell to parse future input according the syntax of that formatter. A line consisting solely of a + will place ispell in TeX/LaTeX mode (similar to the -t option) and - returns ispell to nroff/troff mode (but these commands are obsolete). However, string character type is not changed; the ~ command must be used to do this. A line starting with ~ causes ispell to set internal parameters (in particular, the default string character type) based on the filename given in the rest of the line. (A file suffix is sufficient, but the period must be included. Instead of a filename or suffix, a unique name, as listed in the language affix file, may be specified.) However, the formatter parsing is not changed; the + command must be used to change the formatter. A line prefixed with # will cause the personal dictionary to be saved. A line prefixed with ! will turn on terse mode (explained later in this subsection), and a line prefixed with % will return ispell to normal (non-terse) mode. Any input following the prefix characters +, -, #, !, or % is ignored, as is any input following the filename on a ~ line. To allow spell checking of lines beginning with these characters, a line starting with ^ has that character removed before it is passed to the spell checking code. It is recommended that programmatic interfaces prefix every data line with an up arrow to protect themselves against future changes in ispell.

To summarize these:

*	Add to personal dictionary
@	Accept word, but leave out of dictionary
#	Save current personal dictionary
~	Set parameters based on filename
+	Enter TeX mode
-	Exit TeX mode
!	Enter terse mode
%	Exit terse mode
^	Spell check rest of line

In terse mode, ispell will not print lines beginning with *, +, or -, all of which indicate correct words. This significantly improves running speed when the driving program is going to ignore correct words anyway.

The -s option is only valid in conjunction with the -a or -A options, and only on BSD-derived systems. If specified, ispell will stop itself with a SIGTSTP signal after each line of input. It will not read more input until it receives a SIGCONT signal. This may be useful for handshaking with certain text editors.

The -f option is only valid in conjunction with the -a or -A options. If -f is specified, ispell will write its results to the given file, rather than to standard output.

The -v option causes ispell to print its current version identification on the standard output and exit. If the switch is doubled, ispell will also print the options that it was compiled with.

The -c, -e[1-4], and -D options of ispell are primarily intended for use by the munchlist shell script. The -c switch causes a list of words to be read from the standard input. For each word, a list of possible root words and affixes will be written to the standard output. Some of the root words will be illegal and must be filtered from the output by other means; the munchlist script does this. As an example, the command

```
echo BOTHER ¦ ispell -c
```

produces

```
BOTHER BOTHE/R BOTH/R
```

The -e switch is the reverse of -c; it expands affix flags to produce a list of words. For example, the command

```
echo BOTH/R ¦ ispell -e
```

produces

```
BOTH BOTHER
```

This version of ld uses the general-purpose BFD libraries to operate on object files. This allows ld to read, combine, and write object files in many different formats, for example, COFF or a.out. Different formats may be linked together to produce any available kind of object file. You can use objdump -i to get a list of formats supported on various architectures; see objdump(1).

Aside from its flexibility, the GNU linker is more helpful than other linkers in providing diagnostic information. Many linkers abandon execution immediately upon encountering an error; whenever possible, ld continues executing, allowing you to identify other errors (or, in some cases, to get an output file in spite of the error).

The GNU linker ld is meant to cover a broad range of situations, and to be as compatible as possible with other linkers. As a result, you have many choices to control its behavior through the command line, and through environment variables.

OPTIONS

The plethora of command-line options may seem intimidating, but in actual practice few of them are used in any particular context. For instance, a frequent use of ld is to link standard UNIX object files on a standard, supported UNIX system. On such a system, this line links a file hello.o :

```
$ ld -o output /lib/crt0.o hello.o -lc
```

This tells ld to produce a file called output as the result of linking the file /lib/crt0.o with hello.o and the library libc.a, which will come from the standard search directories.

The command-line options to ld may be specified in any order, and may be repeated at will. For the most part, repeating an option with a different argument will either have no further effect or override prior occurrences (those further to the left on the command line) of that option.

The exceptions—which may meaningfully be used more than once—are -A, -b (or its synonym -format), -defsym, -L, -l, -R, and -u.

The list of object files to be linked together, shown as objfile, may follow, precede, or be mixed in with command-line options, except that an objfile argument may not be placed between an option flag and its argument.

Usually the linker is invoked with at least one object file, but other forms of binary input files can also be specified with -l, -R, and the script command language. If no binary input files at all are specified, the linker does not produce any output, and issues the message No input files.

Option arguments must either follow the option letter without intervening whitespace or be given as separate arguments immediately following the option that requires them.

-Aarchitecture

In the current release of ld, this option is useful only for the Intel 960 family of architectures. In that ld configuration, the architecture argument is one of the two-letter names identifying members of the 960 family; the option specifies the desired output target and warns of any incompatible instructions in the input files. It also modifies the linker's search strategy for archive libraries to support the use of libraries specific to each particular architecture, by including in the search loop names suffixed with the string identifying the architecture.

For example, if your ld command line included -ACA as well as -ltry, the linker would look (in its built-in search paths, and in any paths you specify with -L) for a library with the names

```
try
libtry.a
tryca
libtryca.a
```

The first two possibilities would be considered in any event; the last two are due to the use of -ACA.

Future releases of ld may support similar functionality for other architecture families.

lispmtopgm

lispmtopgm—Convert a Lisp Machine bitmap file into PGM format

SYNOPSIS

```
lispmtopgm [lispmfile]
```

DESCRIPTION

lispmtopgm reads a Lisp machine bitmap as input and produces a portable graymap as output.

This is the file format written by the tv:write-bit-array-file function on TI Explorer and Symbolics Lisp machines.

Multiplane bitmaps on Lisp machines are color; but the lispm image file format does not include a colormap, so it must be treated as a graymap instead. This is unfortunate.

SEE ALSO

pgmtolispm(1), pgm(5)

BUGS

The lispm bitmap file format is a bit quirky; Usually the image in the file has its width rounded up to the next higher multiple of 32, but not always. If the width is not a multiple of 32, we don't deal with it properly, but because of the lispm microcode, such arrays are probably not image data anyway.

Also, the lispm code for saving bitmaps has a bug in that if you are writing a bitmap that is not mod32 across, the file may be up to seven bits too short. They round down instead of up, and we don't handle this bug gracefully.

No color.

AUTHOR

6 March 1990

lkbib

lkbib—Search bibliographic databases

SYNOPSIS

```
lkbib [ -v ][-ifields ][-pfilename ][-tn ] key ...
```

DESCRIPTION

lkbib searches bibliographic databases for references that contain the keys key... and prints any references found on the standard output. lkbib will search any databases given by -p options, and then a default database. The default database is taken from the REFER environment variable if it is set, otherwise it is

```
/usr/dict/papers/Ind.
```

For each database filename to be searched, if an index filename.i created by gindxbib(1) exists, then it will be searched instead; each index can cover multiple databases.

OPTIONS

-v	Print the version number.
-pfilename	Search filename. Multiple -p options can be used.
-istring	When searching files for which no index exists, ignore the contents of fields whose names are in string.
-tn	Only require the first n characters of keys to be given. Initially n is 6.

ENVIRONMENT

`REFER` Default database

FILES

`/usr/dict/papers/Ind`	Default database to be used if the REFER environment variable is not set.
`filename.i`	Index files.

SEE ALSO

`grefer`(1), `glookbib`(1), `gindxbib`(1)

Groff Version 1.09, 6 August 1992

ln

`ln`—Make links between files

SYNOPSIS

```
ln [options] source [dest]
ln [options] source... directory
```

Options:

```
[-bdfinsv] [-S backup-suffix] [-V {numbered,existing,simple}]
[--version-control={numbered,existing,simple}] [--backup] [--directory]
[--force] [--interactive] [--no-dereference] [--symbolic] [--verbose]
[--suffix=backup-suffix] [--help] [--version]
```

DESCRIPTION

This manual page documents the GNU version of `ln`. If the last argument names an existing directory, `ln` links each other given file into a file with the same name in that directory. If only one file is given, it links that file into the current directory. Otherwise, if only two files are given, it links the first onto the second. It is an error if the last argument is not a directory and more than two files are given. It makes hard links by default. By default, it does not remove existing files.

OPTIONS

`-b, --backup`	Make backups of files that are about to be removed.
`-d, -F, --directory`	Allow the superuser to make hard links to directories.
`-f, --force`	Remove existing destination files.
`-i, --interactive`	Prompt whether to remove existing destination files.
`-n, --no-dereference`	When the specified destination is a symbolic link to a directory, attempt to replace the symbolic link rather than dereferencing it to create a link in the directory to which it points. This option is most useful in conjunction with `--force`.
`-s, --symbolic`	Make symbolic links instead of hard links. This option produces an error message on systems that do not support symbolic links.
`-v, --verbose`	Print the name of each file before linking it.
`--help`	Print a usage message on standard output and exit successfully.
`--version`	Print version information on standard output then exit successfully.
`-S, --suffix backup-suffix`	The suffix used for making simple backup files can be set with the `SIMPLE_BACKUP_SUFFIX` environment variable, which can be overridden by this option. If neither of those is given, the default is ~, as it is in Emacs.

locate

`locate`—List files in databases that match a pattern

SYNOPSIS

```
locate [-d path] [--database=path] [--version] [--help] pattern...
```

DESCRIPTION

This manual page documents the GNU version of `locate`. For each given pattern, `locate` searches one or more databases of filenames and displays the filenames that contain the pattern. Patterns can contain shell-style meta characters: `*`, `?`, and `[]`. The meta characters do not treat `/` or `.` specially. Therefore, a pattern `foo*bar` can match a filename that contains `foo3/bar`, and a pattern `*duck*` can match a filename that contains `lake/.ducky`. Patterns that contain meta characters should be quoted to protect them from expansion by the shell.

If a pattern is a plain string—it contains no meta characters—`locate` displays all filenames in the database that contain that string anywhere. If a pattern does contain meta characters, `locate` only displays filenames that match the pattern exactly. As a result, patterns that contain meta characters should usually begin with `*` and will most often end with one as well. The exceptions are patterns that are intended to explicitly match the beginning or end of a filename.

The filename databases contain lists of files that were on the system when the databases were last updated. The system administrator can choose the filename of the default database, the frequency with which the databases are updated, and the directories for which they contain entries; see `updatedb`(1L).

OPTIONS

`-d path, --database=path`
: Instead of searching the default filename database, search the filename databases in `path`, which is a colon-separated list of database filenames. You can also use the environment variable `LOCATE_PATH` to set the list of database files to search. The option overrides the environment variable if both are used.

  The filename database format changed starting with GNU `find` and `locate` version 4.0 to allow machines with different byte orderings to share the databases. This version of `locate` can automatically recognize and read databases produced for older versions of GNU `locate` or UNIX versions of `locate` or `find`.

`--help`
: Print a summary of the options to `locate` and exit.

`--version`
: Print the version number of `locate` and exit.

ENVIRONMENT

`LOCATE_PATH` Colon-separated list of databases to search

SEE ALSO

`find`(1L), `locatedb`(5L), `updatedb`(1L), `xargs`(1L), *Finding Files* (online in `info`, or printed)

logger

`logger`—Make entries in the system log

SYNOPSIS

```
logger [-is] [-f file] [-p pri] [-t tag] [message ...]
```

DESCRIPTION

`logger` provides a shell command interface to the `syslog`(3) system log module.

SEE ALSO

picttoppm(1), pbmtomacp(1), pbm(5)

AUTHOR

29 March 1989

make

make—GNU make utility to maintain groups of programs

SYNOPSIS

```
make [ -f makefile ] [ option ] ... target ...
```

WARNING

This man page is an extract of the documentation of GNU make. It is updated only occasionally because the GNU project does not use nroff. For complete, current documentation, refer to the Info file make or the DVI file make.dvi, which are made from the texinfo source file make.texinfo.

DESCRIPTION

The purpose of the make utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them. This manual page describes the GNU implementation of make, which was written by Richard Stallman and Roland McGrath. Our examples show C programs because they are most common, but you can use make with any programming language whose compiler can be run with a shell command. In fact, make is not limited to programs. You can use it to describe any task where some files must be updated automatically from others whenever the others change.

To prepare to use make, you must write a file called the makefile that describes the relationships among files in your program and states the commands for updating each file. In a program, typically, the executable file is updated from object files, which are in turn made by compiling source files.

Once a suitable makefile exists, each time you change some source files, this simple shell command:

```
make
```

suffices to perform all necessary recompilations. The make program uses the makefile database and the last-modification times of the files to decide which of the files need to be updated. For each of those files, it issues the commands recorded in the database.

make executes commands in the makefile to update one or more target names, where name is typically a program. If no -f option is present, make will look for the makefiles GNU-makefile, makefile, and Makefile, in that order.

Normally you should call your makefile either makefile or Makefile. (We recommend Makefile because it appears prominently near the beginning of a directory listing, right near other important files such as README.) The first name checked, GNUmakefile, is not recommended for most makefiles. You should use this name if you have a makefile that is specific to GNU make, and will not be understood by other versions of make. If makefile is -, the standard input is read.

make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist.

`-S, --suffix backup-suffix`
: The suffix used for making simple backup files can be set with the `SIMPLE_BACKUP_SUFFIX` environment variable, which can be overridden by this option. If neither of those is given, the default is `~`, as it is in Emacs.

`-V, --version-control {numbered,existing,simple}`
: The type of backups made can be set with the `VERSION_CONTROL` environment variable, which can be overridden by this option. If `VERSION_CONTROL` is not set and this option is not given, the default backup type is `existing`. The value of the `VERSION_CONTROL` environment variable and the argument to this option are like the GNU Emacs `version-control` variable; they also recognize synonyms that are more descriptive.

The valid values are the following (unique abbreviations are accepted):

`t` or `numbered`--Always make numbered backups.

`nil` or `existing`--Make numbered backups of files that already have them, simple backups of the others.

`never` or `simple`—Always make simple backups.

GNU File Utilities

mwrite

`mwrite`—Low-level write (copy) a UNIX file to MS-DOS

SYNOPSIS

```
mwrite [ -tnvmoOsSrRA ] unixfile msdosfile
```

```
mwrite [ -tnvmoOsSrRA ] unixfile [ unixfiles... ] msdosdirectory
```

DESCRIPTION

This command is obsolete and only supplied for backward compatibility reasons with old scripts. Use `mcopy` instead.

SEE ALSO

`mcopy`(1), `mtools`(1)

Local

namei

`namei`—Follow a pathname until a terminal point is found

SYNOPSIS

```
namei [-mx] pathname [ pathname ... ]
```

DESCRIPTION

`namei` uses its arguments as pathnames to any type of UNIX file (symlinks, files, directories, and so forth). `namei` then follows each pathname until a terminal point is found (a file, directory, char device, and so on). If it finds a symbolic link, the user shows the link, and starts following it, indenting the output to show the context.

This program is useful for finding `too many levels of symbolic links` problems.

For each line output, `namei` outputs the following characters to identify the file types found:

`f:` The pathname the user is currently trying to resolve

`d` Directory

`l` Symbolic link (both the link and its contents are output)

`s` Socket

AUTHORS

Copyright © 1988 by Jef Poskanzer and Michael Haberler. `-float` and `-noreset` options added by Wim Lewis

29 August 1988

pbmtoln03

`pbmtoln03`—Convert portable bitmap to DEC LN03+ Sixel output

SYNOPSIS

```
pbmtoln03 [-rltbf] pbmfile
```

DESCRIPTION

`pbmtoln03` reads a portable bitmap as input and produces a DEC LN03+ Sixel output file.

OPTIONS

`-l nn` Use `nn` as value for left margin (default `0`).

`-r nn` Use `nn` as value for right margin (default `3400`).

`-t nn` Use `nn` as value for top margin (default `0`).

`-b nn` Use `nn` as value for bottom margin (default `3400`).

`-f nn` Use `nn` as value for form length (default `3400`).

SEE ALSO

`pbm`(5)

AUTHOR

Tim Cook, 26 February 1992

7 May 1993

pbmtolps

`pbmtolps`—Convert a portable bitmap to PostScript

SYNOPSIS

```
pbmtolps [ -dpi n ] [ pbmfile ]
```

DESCRIPTION

`pbmtolps` reads a portable bitmap as input, and outputs PostScript. The output PostScript uses lines instead of the image operator to generate a (device-dependent) picture that will be imaged much faster.

The PostScript path length is constrained to be less that 1000 points so that no limits are overrun on the Apple Laserwriter and (presumably) no other printers.

SEE ALSO

`pgmtops`(1), `ppmtops`(1), `pbm`(5)

AUTHOR

George Phillips (<phillips@cs.ubc.ca>)

pbmtopgm

pbmtopgm—Convert portable bitmap to portable graymap by averaging areas

SYNOPSIS

```
pbmtopgm <width><height> [pbmfile]
```

DESCRIPTION

pbmtopgm reads a portable bitmap as input and outputs a portable graymap created by averaging the number of pixels within a sample area of width by height around each point. pbmtopgm is similar to a special case of ppmconvol. A ppmsmooth step may be needed after pbmtopgm.

pbmtopgm has the effect of antialiasing bitmaps that contain distinct line features.

SEE ALSO

pbm(5)

AUTHOR

NOTES

pbmtopgm works best with odd sample widths and heights.

pbmtopi3

pbmtopi3—Convert a portable bitmap into an Atari Degas PI3 file

SYNOPSIS

```
pbmtopi3 [pbmfile]
```

DESCRIPTION

pbmtopi3 reads a portable bitmap as input and produces an Atari Degas PI3 file as output.

SEE ALSO

pi3topbm(1), pbm(5), ppmtopi1(1), pi1toppm(1)

AUTHOR

11 March 1990

pbmtopk

pbmtopk—Convert a portable bitmap into a packed (PK) format font

SYNOPSIS

```
pbmtopk pkfile[.pk] tfmfile[.tfm] resolution [-s designsize] [-p num param...]
[-C cod-ingscheme] [-F family] [-f optfile] [-c num] [-W width] [-H height]
[-D depth] [-I ital] [-h horiz] [-v vert] [-x xoff] [-y yoff] [pbmfile]...
```

AUTHOR

John Walker
Autodesk SA Avenue des Champs-Montants 14b
CH-2074 MARIN
Suisse/Schweiz/Svizzera/Svizra/Switzerland

Usenet: kelvin@Autodesk.com
Fax: 038/33 88 15
Voice: 038/33 76 33

PLUGWARE! If you like this kind of stuff, you may also enjoy James Gleick’s “Chaos—The Software” for MS-DOS, available for $59.95 from your local software store or directly from Autodesk, Inc., Attn: Science Series, 2320 Marinship Way, Sausalito, CA 94965, USA. Telephone: 800-688-2344 toll-free or, outside the U.S. (415) 332-2344 Ext 4886. Fax: 415-289-4718. “Chaos—The Software” includes a more comprehensive fractal imagery generator that creates three-dimensional landscapes as well as clouds and planets, plus five more machines that explore other aspects of Chaos. The user guide of more than 200 pages includes an introduction by James Gleick and detailed explanations by Rudy Rucker of the mathematics and algorithms used by each program.

15 October 1991

pgmedge

pgmedge—Edge detect a portable graymap

SYNOPSIS

```
pgmedge [pgmfile]
```

DESCRIPTION

pgmedge reads a portable graymap as input, outlines the edges, and writes a portable graymap as output. Piping the result through pgmtopbm -threshold and playing with the threshold value will give a bitmap of the edges.

The edge detection technique used is to take the Pythagorean sum of two Sobel gradient operators at 90 degrees to each other. For more details see *Digital Image Processing* by Gonzalez and Wintz, Chapter 7.

SEE ALSO

pgmenhance(1), pgmtopbm(1), pgm(5), pbm(5)

AUTHOR

4 February 1990

pgmenhance

pgmenhance—Edge enhance a portable graymap

SYNOPSIS

```
pgmenhance [-N][pgmfile]
```

pgmtoppm

pgmtoppm—Colorize a portable graymap into a portable pixmap

SYNOPSIS

```
pgmtoppm colorspec [pgmfile]
pgmtoppm colorspec1-colorspec2 [pgmfile]
pgmtoppm -map mapfile [pgmfile]
```

DESCRIPTION

pgmtoppm reads a portable graymap as input, colorizes it by multiplying the gray values by specified color or colors, and produces a portable pixmap as output.

If only one color is specified, black in the PGM file stays black and white in the PGM file turns into the specified color in the PPM file. If two colors (separated by a hyphen) are specified, then black gets mapped to the first color and white gets mapped to the second.

The color can be specified in five ways:

- A name, assuming that a pointer to an X11-style color names file was compiled in.
- An X11-style hexadecimal specifier: `rgb:r/g/b`, where `r`, `g`, and `b` are each 1- to 4-digit hexadecimal numbers.
- An X11-style decimal specifier: `rgbi:r/g/b`, where `r`, `g`, and `b` are floating-point numbers between 0 and 1.
- For backwards compatibility, an old-X11-style hexadecimal number: `#rgb`, `#rrggbb`, `#rrrgggbbb`, or `#rrrrggggbbbb`.
- For backwards compatibility, a triplet of numbers separated by commas: `r,g,b`, where `r`, `g`, and `b` are floating-point numbers between 0 and 1. (This style was added before MIT came up with the similar `rgbi` style.)

Also, the `-map` flag lets you specify an entire colormap to be used. The mapfile is just a PPM file; it can be any shape, all that matters is the colors in it and their order. In this case, black gets mapped to the first color in the mapfile, and white gets mapped to the last.

SEE ALSO

rgb3toppm(1), ppmtopgm(1), ppmtorgb3(1), ppm(5), pgm(5)

AUTHOR

11 January 1991

pi1toppm

pi1toppm—Convert an Atari Degas PI1 into a portable pixmap

SYNOPSIS

```
pi1toppm [pi1file]
```

DESCRIPTION

pi1toppm reads an Atari Degas PI1 file as input and produces a portable pixmap as output.

SEE ALSO

ppmtopi1(1), ppm(5), pi3topbm(1), pbmtopi3(1)

AUTHORS

19 July 1990

fontdir FILE FORMAT

picttoppm has a built-in default font and your local installer probably provided adequate extra fonts. You can point picttoppm at more fonts that you specify in a font directory file. Each line in the file is either a comment line, which must begin with #, or font information. The font information consists of four whitespace separated fields. The first is the font number, the second is the font size in pixels, the third is the font style, and the fourth is the name of a BDF file containing the font. The BDF format is defined by the X Window System and is not described here.

The font number indicates the type face. Here is a list of known font numbers and their faces.

0	Chicago
1	Application font
2	New York
3	Geneva
4	Monaco
5	Venice
6	London
7	Athens
8	San Francisco
9	Toronto
11	Cairo
12	Los Angeles
20	Times Roman
21	Helvetica
22	Courier
23	Symbol
24	Taliesin

The font style indicates a variation on the font. Multiple variations may apply to a font and the font style is the sum of the variation numbers, which are

1	Boldface
2	Italic
4	Underlined
8	Outlined
16	Shadow
32	Condensed
64	Extended

Obviously, the font definitions are strongly related to the Macintosh. More font numbers and information about fonts can be found in Macintosh documentation.

SEE ALSO

Inside Macintosh volumes 1 and 5, ppmtopict(1), ppm(5)

AUTHOR

29 November 1991

DESCRIPTION

`pnmdepth` reads a portable anymap as input, scales all the pixel values, and writes out the image with the new `maxval`. Scaling the colors down to a smaller `maxval` will result in some loss of information.

Be careful of off-by-one errors when choosing the new `maxval`. For instance, if you want the color values to be five bits wide, use a `maxval` of `31`, not `32`.

SEE ALSO

`pnm`(5), `ppmquant`(1), `ppmdither`(1)

AUTHOR

12 January 1991

pnmenlarge

`pnmenlarge`—Read a portable anymap and enlarge it N times

SYNOPSIS

```
pnmenlarge N [pnmfile]
```

DESCRIPTION

`pnmenlarge` reads a portable anymap as input, replicates its pixels `N` times, and produces a portable anymap as output.

`pnmenlarge` can only enlarge by integer factors. The slower but more general `pnmscale` can enlarge or reduce by arbitrary factors, and `pbmreduce` can reduce by integer factors, but only for bitmaps.

If you enlarge by a factor of 3 or more, you should probably add a `pnmsmooth` step; otherwise, you can see the original pixels in the resulting image.

SEE ALSO

`pbmreduce`(1), `pnmscale`(1), `pnmsmooth`(1), `pnm`(5)

AUTHOR

26 February 1989

pnmfile

`pnmfile`—Describe a portable anymap

SYNOPSIS

```
pnmfile [pnmfile] ...
```

DESCRIPTION

`pnmfile` reads one or more portable anymaps as input and writes out short descriptions of the image type, size, and so on. This is mostly for use in shell scripts, so the format is not particularly pretty.

SEE ALSO

`pnm`(5), `file`(1)

pnmtoddif

pnmtoddif—Convert a portable anymap to DDIF format

SYNTAX

```
pnmtoddif pnmtoddif [-resolution x y] [pnmfile [ddiffile]]
```

OPTIONS

`resolution x y`	The horizontal and vertical resolution of the output image in dots per inch. Defaults to `78dpi`.
`pnmfile`	The filename for the image file in PNM format. If this argument is omitted, input is read from `stdin`.
`ddiffile`	The filename for the image file to be created in DDIF format. If this argument is omitted, the `ddiffile` is written to standard output. It can only specified if a `pnmfile` is also specified.

DESCRIPTION

`pnmtoddif` takes a portable anymap from standard input and converts it into a DDIF image file on standard output or the specified DDIF file.

PBM format (bitmap) data is written as 1-bit DDIF, PGM format data (grayscale) as 8-bit grayscale DDIF, and PPM format data is written as 8,8,8-bit color DDIF. All DDIF image files are written as uncompressed. The data plane organization is interleaved by pixel.

In addition to the number of pixels in the width and height dimensions, DDIF images also carry information about the size that the image should have, that is, the physical space that a pixel occupies. PBMPLUS images do not carry this information, hence it has to be externally supplied. The default of `78dpi` has the beneficial property of not causing a resize on most Digital Equipment Corporation color monitors.

AUTHOR

Burkhard Neidecker-Lutz
Digital Equipment Corporation, CEC Karlsruhe
`neideck@nestvx.enet.dec.com`

pnmtofits

pnmtofits—Convert a portable anymap into FITS format

SYNOPSIS

```
pnmtofits [-max f][-min f][pnmfile]
```

DESCRIPTION

`pnmtofits` reads a portable anymap as input and produces a FITS (Flexible Image Transport System) file as output. The resolution of the output file is either 8 bits/pixel, or 16 bits/pixel, depending on the value of `maxval` in the input file. If the input file is a portable bitmap or a portable graymap, the output file consists of a single plane image (`NAXIS = 2`). If instead the input file is a portable pixmap, the output file will consist of a three-plane image (`NAXIS = 3`, `NAXIS3 = 3`). A full description of the FITS format can be found in *Astronomy & Astrophysics Supplement Series 44* (1981), page 363.

OPTIONS

Flags `-min` and `-max` can be used to set `DATAMAX`, `DATAMIN`, `BSCALE`, and `BZERO` in the FITS header, but do not cause the data to be rescaled.

SEE ALSO

`fitstopnm`(1), `pgm`(5)

SEE ALSO

ppm(5)

AUTHOR

2 November 1993

ppmbrighten

ppmbrighten—Change an image's saturation and value from an HSV map

SYNOPSIS

```
ppmbrighten [-n] [-s <+- saturation>] [-v <+- value>] <ppmfile>
```

DESCRIPTION

ppmbrighten reads a portable pixmap as input, converts the image from RGB space to HSV space, and changes the value by <+- value> as a percentage; the same with the saturation. Use

```
ppmbrighten -v 100
```

to add 100 percent to the value.

The -n option normalizes the value to exist between 0 and 1 (normalized).

SEE ALSO

pgmnorm(1), ppm(5)

AUTHOR

NOTES

This program does not change the number of colors.

20 November 1990

ppmchange

ppmchange—Change all pixels of one color to another in a portable pixmap

SYNOPSIS

```
ppmchange oldcolor newcolor [...] [ppmfile]
```

DESCRIPTION

ppmchange reads a portable pixmap as input and changes all pixels of oldcolor to newcolor, leaving all others unchanged. Up to 256 colors may be replaced by specifying couples of colors on the command line.

`-ice level`

Sets the extent of the polar ice caps to the given floating-point `level`. The default level of `0.4` produces ice caps similar to those of the Earth. Smaller values reduce the amount of ice, while larger `-ice` settings create more prominent ice caps. Sufficiently large values, such as 100 or more, in conjunction with small settings for `-glaciers` (try `0.1`) create "ice balls" like Europa.

`-inclination|-tilt angle`

The inclination angle of the planet with regard to its primary star is set to `angle`, which can be any floating-point value from -90 to 90. The inclination angle can be thought of as specifying, in degrees, the "season" the planet is presently experiencing or, more precisely, the latitude at which the star transits the zenith at local noon. If 0, the planet is at equinox; the star is directly overhead at the equator. Positive values represent summer in the northern hemisphere, negative values summer in the southern hemisphere. The Earth's inclination angle, for example, is about 23.5 at the June solstice, 0 at the equinoxes in March and September, and -23.5 at the December solstice. If no inclination angle is specified, a random value between -21.6 and 21.6 degrees is chosen.

`-mesh size`

A mesh of `size` by `size` will be used for the fast Fourier transform (FFT). Note that memory requirements and computation speed increase as the square of `size`; if you double the mesh size, the program will use four times the memory and run four times as long. The default mesh is `256x256`, which produces reasonably good looking pictures while using half a megabyte for the `256x256` array of single precision complex numbers required by the FFT. On machines with limited memory capacity, you may have to reduce the mesh size to avoid running out of RAM. Increasing the mesh size produces better looking pictures; the difference becomes particularly noticeable when generating high-resolution images with relatively high fractal dimensions (between 2.2 and 3).

`-night`

A starry sky is generated. The stars are created by the same algorithm used for the stars that surround planet pictures, but the output consists exclusively of stars.

`-power factor`

Sets the power factor used to scale elevations synthesized from the FFT to `factor`, which can be any floating-point number greater than zero. If no factor is specified, a default of `1.2` is used if a planet is being generated, or `0.75` if clouds are selected by the `-clouds` option. The result of the FFT image synthesis is an array of elevation values between 0 and 1. A nonunity power factor exponentiates each of these elevations to the specified power. For example, a power factor of `2` squares each value, while a power factor of `0.5` replaces each with its square root. (Note that exponentiating values between 0 and 1 yields values that remain within that range.) Power factors less than 1 emphasize large-scale elevation changes at the expense of small variations. Power factors greater than 1 increase the roughness of the terrain and, like high fractal dimensions, may require a larger FFT mesh size or higher screen resolution to look good.

`-saturation sat`

Controls the degree of color saturation of the stars that surround planet pictures and fill starry skies created with the `-night` option. The default value of `125` creates stars that resemble the sky as seen by the human eye from Earth's surface. Stars are dim; only the brightest activate the cones in the human retina, causing color to be perceived. Higher values of `sat` approximate the appearance of stars from Earth orbit, where better dark adaptation, absence of sky glow, and the concentration of light from a given star onto a smaller area of the retina thanks to the lack of atmospheric turbulence enhances the perception of color. Values greater than 250 create "science fiction" skies that, while pretty, don't occur in this universe.

Thanks to the inverse square law combined with nature's love of mediocrity, there are many, many dim stars for every bright one. This population relationship is accurately reflected in the skies created by `ppmforge`. Dim, low mass stars live much longer than bright, massive stars; consequently there are many reddish stars for every blue giant. This relationship is preserved by `ppmforge`. You can reverse the proportion, simulating the sky as seen in a starburst galaxy, by specifying a negative `sat` value.

Usenet: kelvin@Autodesk.com
Fax: 038/33 88 15
Voice: 038/33 76 33

10 October 1991

ppmtobmp

ppmtobmp—Convert a portable pixmap into a BMP file

SYNOPSIS

```
ppmtobmp [-windows][-os2][ppmfile]
```

DESCRIPTION

ppmtobmp reads a portable pixmap as input and produces a Microsoft Windows or OS/2 BMP file as output.

OPTIONS

-windows Tells the program to produce a Microsoft Windows BMP file.

-os2 Tells the program to produce an OS/2 BMP file. (This is the default.)

All flags can be abbreviated to their shortest unique prefix.

SEE ALSO

bmptoppm(1), ppm(5)

AUTHOR

26 October 1992

ppmtogif

ppmtogif—Convert a portable pixmap into a GIF file

SYNOPSIS

```
ppmtogif [-interlace][-sort][-map mapfile][-transparent color][ppmfile]
```

DESCRIPTION

ppmtogif reads a portable pixmap as input and produces a GIF file as output.

OPTIONS

-interlace Tells the program to produce an interlaced GIF file.

-sort Produces a GIF file with a sorted colormap.

-map mapfile Uses the colors found in the mapfile to create the colormap in the GIF file, instead of the colors from ppmfile. The mapfile can be any ppm file; all that matters is the colors in it. If the colors in

SEE ALSO

picttoppm(1), ppm(5), mcvert(1)

AUTHOR

15 April 1990

ppmtopj

ppmtopj—Convert a portable pixmap to an HP PaintJet file

SYNOPSIS

```
ppmtopj [-gamma val][-xpos val][-ypos val][-back dark¦lite][-rle][-center]
[-render none¦snap¦bw¦dither¦diffuse¦monodither¦monodiffuse¦clusterdither¦
monoclusterdither][ppmfile]
```

DESCRIPTION

ppmtopj reads a portable pixmap as input and converts it into a format suitable to be printed by an HP PaintJet printer.

For best results, the input file should be in 8-color RGB form, that is it should have only the eight binary combinations of full-on and full-off primaries. You could get this by sending the input file through ppmquant -map with a mapfile such as

```
P3
8 1
255
0 0 0 2550 0 02550 0 0 255
255 255 0 255 0 255 0 255 255 255 255 255
```

Or else you could use ppmdither -red 2 -green 2 -blue 2.

OPTIONS

-rle	Run-length encode the image. (This can result in larger images.)
-back	Enhance the foreground by indicating if the background is light or dark compared to the foreground.
-render alg	Use an internal rendering algorithm (default dither).
-gamma int	Gamma correct the image using the integer parameter as a gamma (default 0).
-center	Center the image to an 8.5 by 11 page.
-xpos pos	Move by pos pixels in the x direction.
-ypos pos	Move by pos pixels in the y direction.

REFERENCES

HP PaintJet XL Color Graphics Printer User's Guide

SEE ALSO

pnmdepth(1), ppmquant(1), ppmdither(1), ppm(5)

BUGS

Most of the options have not been tested because of the price of the paper.

AUTHOR

13 July 1991

DESCRIPTION

`ppmtoyuvsplit` reads a portable pixmap as input and produces three raw files—`basename.Y`, `basename.U`, and `basename.V`—as output. These files are the subsampled raw YUV representation of the input pixmap, as required by the Stanford MPEG code. The subsampling is done by arithmetic mean of 4 pixels colors into one. The YUV values are scaled according to CCIR.601, as assumed by MPEG.

SEE ALSO

`mpeg`(1), `ppm`(5)

AUTHOR

9 September 1993

pr

`pr`—Convert text files for printing

SYNOPSIS

```
pr [+PAGE] [-COLUMN] [-abcdfFrt] [-e[in-tab-char[in-tab-width]]] [-h header]
[-i[out-tab-char[out-tab-width]]] [-l page-length] [-n[number-separator[digits]]]
[-o left-margin] [-s[column-separator]] [-w page-width] [--help] [-- version] [file...]
```

DESCRIPTION

This manual page documents the GNU version of `pr`. `pr` prints on the standard output a paginated and optionally multicolumn copy of the text files given on the command line, or of the standard input if no files are given or when the filename `-` is encountered. Form feeds in the input cause page breaks in the output.

OPTIONS

`PAGE`	Begin printing with page `PAGE`.
`-COLUMN`	Produce `COLUMN`-column output and print columns down. The column width is automatically decreased as `COLUMN` increases; unless you use the `-w` option to increase the page width as well, this option might cause some columns to be truncated.
`-a`	Print columns across rather than down.
`-b`	Balance columns on the last page.
`-c`	Print control characters using hat notation (for example, ^G); print other unprintable characters in octal backslash notation.
`-d`	Double-space the output.
`-e[in-tab-char [in-tab-width]]`	Expand tabs to spaces on input. Optional argument `in-tab-char` is the input tab character, default `tab`. Optional argument `in-tab-width` is the input tab character's width, default `8`.
`-F, -f`	Use a form feed instead of newlines to separate output pages.
`-h header`	Replace the filename in the header with the string `header`.
`--help`	Print a usage message and exit with a nonzero status.
`-i[out-tab-char [out-tab-width]]`	Replace spaces with tabs on output. Optional argument `out-tab-char` is the output tab character, default `tab`. Optional argument `out-tab-width` is the output tab character's width, default `8`.
`-l page-length`	Set the page length to `page-length` lines. The default is `66`. If `page-length` is less than 10, the headers and footers are omitted, as if the `-t` option had been given.
`-m`	Print all files in parallel, one in each column.

`w`	Wide output: don't truncate command lines to fit on one line.
`h`	No header.
`r`	Running procs only.
`n`	Numeric output for `USER` and `WCHAN`.
`txx`	Only procs with controlling `tty` `xx`; use for `xx` the same letters as shown in the `TT` field. The `tty` name must be given immediately after the option, with no intervening space, for example, `ps -tv1`.
`O[+|-]k1[,[+|-]k2[,...]]`	Order the process listing according to the multilevel sort specified by the sequence of `short` keys from `SORT KEYS`, `k1`, `k2`, ... Default order specifications exist for each of the various formats of `ps`. These are overridden by a user-specified ordering. The `+` is quite optional, merely reiterating the default direction on a key. `-` reverses direction only on the key it precedes. As with `t` and `pids`, the `O` option must be the last option in a single command argument, but specifications in successive arguments are catenated.
`pids`	List only the specified processes; they are comma-delimited. The list must be given immediately after the last option in a single command-line argument, with no intervening space, for example, `ps -j1,4,5`. Lists specified in subsequent arguments are catenated, for example, `ps -l1,23,4 5 6` will list all the processes 1-6 in long format.

LONG COMMAND-LINE OPTIONS

These options are preceded by a double hyphen.

`--sortX[+|-]key[X[+|-]key[...]]`	Choose a multiletter key from the `SORTKEYS` section. `X` may be any convenient separator character. To be UNIX-ish, use `=`. The `+` is really optional because default direction is increasing numerical or lexicographic order. Example: `ps -jax --sort=uid,-ppid,+pid`
`--help`	Get a help message that summarizes the usage and gives a list of supported sort keys. This list may be more up-to-date than this man page.

SORT KEYS

Note that the values used in sorting are the internal values `ps` uses and *not* the "cooked" values used in some of the output format fields. If someone wants to volunteer to write special comparison functions for the cooked values,...;-)

Short	*Long*	*Description*
`c`	`cmd`	Simple name of executable
`C`	`cmdline`	Full command line
`f`	`flags`	Flags as in long format F field
`g`	`pgrp`	Process group ID
`G`	`tpgid`	Controlling `tty` process group ID
`j`	`cutime`	Cumulative user time
`J`	`cstime`	Cumulative system time
`k`	`utime`	User time
`K`	`stime`	System time
`m`	`min_flt`	Number of minor page faults
`M`	`maj_flt`	Number of major page faults
`n`	`cmin_flt`	Cumulative minor page faults
`N`	`cmaj_flt`	Cumulative major page faults
`o`	`session`	Session ID
`p`	`pid`	Process ID

continues

SEE ALSO

ppm(5)

AUTHOR

25 August 1989

quota

quota—Display disk usage and limits

SYNOPSIS

```
quota [ -guv ¦ q ]
quota [ -uv ¦ q ] user
quota [ -gv ¦ q ] group
```

DESCRIPTION

quota displays users' disk usage and limits. By default, only the user quotas are printed.

-g Print group quotas for the group of which the user is a member. The optional -u flag is equivalent to the default.

-v Will display quotas on filesystems where no storage is allocated.

-q Print a more terse message, containing only information on filesystems where usage is over quota.

Specifying both -g and -u displays both the user quotas and the group quotas (for the user).

Only the superuser may use the -u flag and the optional user argument to view the limits of other users. Non-superusers can use the -g flag and optional group argument to view only the limits of groups of which they are members.

The -q flag takes precedence over the -v flag.

quota reports the quotas of all the filesystems listed in /etc/fstab. For filesystems that are NFS-mounted, a call to the rpc.rquotad on the server machine is performed to get the information. If quota exits with a nonzero status, one or more filesystems are over quota.

FILES

quota.user	Located at the filesystem root with user quotas
quota.group	Located at the filesystem root with group quotas
/etc/fstab	To find filesystem names and locations

SEE ALSO

quotactl(2), fstab(5), edquota(8), quotacheck(8), quotaon(8), repquota(8)

8 January 1993

ranlib

ranlib—Generate index to archive

SYNOPSIS

```
ranlib [ -v¦-V] archive
```

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.6; Release Date: 1995/06/01.

SEE ALSO

`ci`(1), `co`(1), `ident`(1), `merge`(1), `rcs`(1), `rcsdiff`(1), `rcsintro`(1), `rlog`(1), `rcsfile`(5)

"RCS—A System for Version Control" by Walter F. Tichy, `Software-Practice & Experience` 15, 7 (July 1985), pages 637–654.

rdist

`rdist`—Remote file distribution program

SYNOPSIS

```
rdist [-nqbRhivwy] [-f distfile] [-d var=value] [-m host] [name ...]
rdist [-nqbRhivwy] -c name ... [login@host:dest]
```

DESCRIPTION

`rdist` is a program to maintain identical copies of files over multiple hosts. It preserves the owner, group, mode, and mtime of files if possible and can update programs that are executing. `rdist` reads commands from `distfile` to direct the updating of files and/or directories.

Options specific to the first `SYNOPSIS` form:

`-` If `distfile` is `-`, the standard input is used.

`-f distfile` Use the specified `distfile`.

If either the `-f` or `-` option is not specified, the program looks first for `distfile`, then `Distfile` to use as the input. If no names are specified on the command line, `rdist` will update all of the files and directories listed in `distfile`. Otherwise, the argument is taken to be the name of a file to be updated or the label of a command to execute. If label and filenames conflict, it is assumed to be a label. These may be used together to update specific files using specific commands.

Options specific to the second `SYNOPSIS` form:

`-c` Forces `rdist` to interpret the remaining arguments as a small `distfile`.

The equivalent `distfile` is as follows:

```
name ... -¿ login@ host install dest ;
```

Options common to both forms:

`-b` Binary comparison. Perform a binary comparison and update files if they differ rather than comparing dates and sizes.

`-d var=value` Define `var` to have `value`. The `-d` option is used to define or override variable definitions in the `distfile`. `value` can be the empty string, one name, or a list of names surrounded by parentheses and separated by tabs and/or spaces.

`-h` Follow symbolic links. Copy the file that the link points to rather than the link itself.

`-i` Ignore unresolved links. `rdist` will normally try to maintain the link structure of files being transferred and warn the user if all the links cannot be found.

`-m host` Limit which machines are to be updated. Multiple `-m` arguments can be given to limit updates to a subset of the hosts listed in the `distfile`.

NOTES

You might want to generate a `tags` file for the directory that contains the source code for standard C library on your system. If licensing restrictions prevent you from making the library source readable by everybody, then you can have `ctags` generate a `refs` file, and make `refs` readable by everybody.

If your system doesn't come with the library source code, then perhaps you can produce something workable from the `lint` libraries.

SEE ALSO

`elvis`(1), `ctags`(1)

AUTHOR

Steve Kirkendall (`kirkenda@cs.pdx.edu`)

reset

`reset`—Reset the terminal

SYNOPSIS

```
clear
```

DESCRIPTION

`reset` calls `tput`(1) with the `clear`, `rmacs`, `rmm`, `smkx`, `rs1`, `rs2`, and `rs3` arguments. This causes `tput` to send appropriate reset strings to the terminal based on information in `/etc/termcap` (for the GNU or BSD `tput`) or in the `terminfo` database (for the `ncurses tput`). This sequence seems to be sufficient to reset the Linux VC's when they start printing "funny-looking" characters. For good measure, `stty`(1) is called with the `sane` argument in an attempt to get Cooked mode back.

SEE ALSO

`reset`(1), `stty`(1), `tput`(1)

AUTHOR

Rik Faith (`faith@cs.unc.edu`)

Linux 0.99, 10 October 1993

resize

`resize`—Set `TERMCAP` and terminal settings to current `xterm` window size

SYNOPSIS

```
resize [ -u ¦ -c ][-s [ row col ]]
```

DESCRIPTION

`resize` prints a shell command for setting the `TERM` and `TERMCAP` environment variables to indicate the current size of `xterm` window from which the command is run. For this output to take effect, `resize` must either be evaluated as part of the command line (usually done with a shell alias or function) or else redirected to a file that can then be read in. From the C shell (usually known as `/bin/csh`), the following alias could be defined in the user's `.cshrc`:

```
% alias rs 'set noglob; eval 'resize''
```

After resizing the window, the user would type:

```
%rs
```

SYNOPSIS

```
rgb3toppmredpgmfile greenpgmfile bluepgmfile
```

DESCRIPTION

`rgb3toppm` reads three portable graymaps as input and combines them and produces one portable pixmap as output.

SEE ALSO

`ppmtorgb3`(1), `pgmtoppm`(1), `ppmtopgm`(1), `ppm`(5), `pgm`(5)

AUTHOR

15 February 1991

rlog

`rlog`—Print log messages and other information about RCS files

SYNOPSIS

```
rlog [ options ] file ...
```

DESCRIPTION

`rlog` prints information about RCS files.

Pathnames matching an `RCS` suffix denote RCS files; all others denote working files. Names are paired as explained in `ci`(1).

`rlog` prints the following information for each RCS file: RCS pathname, working pathname, head (the number of the latest revision on the trunk), default branch, access list, locks, symbolic names, suffix, total number of revisions, number of revisions selected for printing, and descriptive text. This is followed by entries for the selected revisions in reverse chronological order for each branch. For each revision, `rlog` prints revision number, author, date/time, state, number of lines added/deleted (with respect to the previous revision), locker of the revision (if any), and log message. All times are displayed in Coordinated Universal Time (UTC) by default; this can be overridden with `-z`. Without options, `rlog` prints complete information. The options below restrict this output.

`-L`	Ignore RCS files that have no locks set. This is convenient in combination with `-h`, `-l`, and `-R`.
`-R`	Print only the name of the `RCS` file. This is convenient for translating a working pathname into an RCS pathname.
`-h`	Print only the RCS pathname, working pathname, head, default branch, access list, locks, symbolic names, and suffix.
`-t`	Print the same as `-h`, plus the descriptive text.
`-N`	Do not print the symbolic names.
`-b`	Print information about the revisions on the default branch, normally the highest branch on the trunk.
`-ddates`	Print information about revisions with a checkin date/time in the ranges given by the semicolon-separated list of `dates`. A range of the form `d1<d2` or `d2>d1` selects the revisions that were deposited between `d1` and `d2` exclusive. A range of the form `<d` or `d>` selects all revisions earlier than `d`. A range of the form `d<` or `>d` selects all revisions dated later than `d`. If `<` or `>` is followed by `=`, then the ranges are inclusive, not exclusive. A range of the form `d` selects the single, latest revision dated `d` or earlier. The date/time strings `d`, `d1`, and `d2` are in the free format explained in `co`(1). Quoting is normally necessary, especially for `<` and `>`. Note that the separator is a semicolon.
`-l[lockers]`	Print information about locked revisions only. In addition, if the comma-separated list `lockers` of login names is given, ignore all locks other than those held by the `lockers`. For example, `rlog -L -R -lwft RCS/*` prints the name of RCS files locked by the user `wft`.

IDENTIFICATION

Author: Walter F. Tichy
Manual Page Revision: 5.9; Release Date: 1995/06/16

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsintro(1), rcsmerge(1), rcsfile(5)

"RCS-A System for Version Control" by Walter F. Tichy, *Software-Practice & Experience* 15, 7 (July 1985), pages 637–654.

BUGS

The separator for revision ranges in the -r option used to be - instead of :, but this leads to confusion when symbolic names contain -. For backwards compatibility, rlog -r still supports the old - separator, but it warns about its obsolete use.

GNU, 16 June 1995

rlogin

rlogin—Remote login

SYNOPSIS

rlogin [-8EKLx] [-e char] [-k realm] [-l username] host

DESCRIPTION

rlogin starts a terminal session on a remote host host.

rlogin first attempts to use the Kerberos authorization mechanism, described in the following subsection. If the remote host does not support Kerberos, the standard Berkeley authorization mechanism is used. The options are as follows:

-8 The -8 option allows an eight-bit input data path at all times; otherwise, parity bits are stripped except when the remote side's stop and start characters are other than ^S/^Q.

-E The -E option stops any character from being recognized as an escape character. When used with the -8 option, this provides a completely transparent connection.

-K The -K option turns off all Kerberos authentication.

-L The -L option allows the rlogin session to be run in litout mode.(See tty(4) for details).

-d The -d option turns on socket debugging (see the setsockopt(2) man page) on the TCP sockets used for communication with the remote host.

-e The -e option allows user specification of the escape character, which is the tilde (~) by default. This specification may be as a literal character, or as an octal value in the form nnnn.

-k The -k option requests rlogin to obtain tickets for the remote host in realm realm instead of the remote host's realm as determined by krb_realmofhost(3).

-x The -x option turns on DES encryption for all data passed via the rlogin session. This may impact response time and CPU utilization, but provides increased security.

A line of the form <escape char> disconnects from the remote host. Similarly, the line <escape char>^Z will suspend the rlogin session, and <escape char><delayed-suspend char> suspends the send portion of the rlogin, but allows output from the remote system. By default, the tilde (~) character is the escape character, and normally control -Y (^Y) is the delayed-suspend character.

All echoing takes place at the remote site, so that (except for delays) the rlogin is transparent. Flow control via ^S/^Q and flushing of input and output on interrupts is handled properly.

`-f`, `--force`	Ignore nonexistent files and never prompt the user.
`-i`, `--interactive`	Prompt whether to remove each file. If the response does not begin with `y` or `Y`, the file is skipped.
`-r`, `-R`, `--recursive`	Remove the contents of directories recursively.
`-v`, `--verbose`	Print the name of each file before removing it.
`--help`	Print a usage message on standard output and exit successfully.
`--version`	Print version information on standard output, then exit successfully.

GNU File Utilities

rmdir

`rmdir`—Remove empty directories

SYNOPSIS

`rmdir` `[-p] [--parents] [--help] [--version] dir...`

DESCRIPTION

This manual page documents the GNU version of `rmdir`. `rmdir` removes each given empty directory. If any nonoption argument does not refer to an existing empty directory, it is an error.

OPTIONS

`-p`, `--parents`	Remove any parent directories that are explicitly mentioned in an argument, if they become empty after the argument file is removed.
`--help`	Print a usage message on standard output and exit successfully.
`--version`	Print version information on standard output, then exit successfully.

GNU File Utilities

rmmod

`rmmod`—Unload loadable modules

SYNOPSIS

```
rmmod [ -r ] module ...
```

DESCRIPTION

`rmmod` unloads loadable modules from the kernel.

`rmmod` tries to unload a set of modules from the kernel, with the restriction that they are not in use and that they are not referred to by other modules.

If more than one module is named on the command line, the modules will be removed in the given order. This supports unloading of stacked modules.

With the option `-r`, a recursive removal of modules will be attempted. This means that if a top module in a stack is named on the command line, all modules that are used by this module will be removed as well, if possible.

SEE ALSO

`insmod`(1), `lsmod`(1), `ksyms`(1), `modules`(2)

HISTORY

Written by Rich $alz (rsalz@uunet.uu.net) for InterNetNews.

SEE ALSO

innd(8).

rpcgen

rpcgen—An RPC protocol compiler

SYNOPSIS

```
rpcgen infile
rpcgen [-D name[= value]] [-T] [-K secs] infile
rpcgen -c¦-h¦-l¦-m¦-t [-o outfile] infile
rpcgen [-I] -s nettype [-o outfile] infile
rpcgen -n netid [-o outfile] infile
```

DESCRIPTION

rpcgen is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language similar to C known as RPC language (Remote Procedure Call Language). rpcgen is normally used as in the first synopsis where it takes an input file and generates up to four output files. If the infile is named proto.x, then rpcgen will generate a header file in proto.h, xdr routines in proto_xdr.c, server-side stubs in proto_svc.c, and client-side stubs in proto_clnt.c.

With the -T option, it will also generate the RPC dispatch table in proto_tbl.i. With the -Sc option, it will also generate sample code that would illustrate how to use the remote procedures on the client side. This code would be created in proto_client.c. With the -Ss option, it will also generate a sample server code that would illustrate how to write the remote procedures. This code would be created in proto_server.c. The server created can be started both by the port monitors (for example, inetd or listen) or by itself. When it is started by a port monitor, it creates servers only for the transport for which the file descriptor 0 was passed. The name of the transport must be specified by setting up the environmental variable PM_TRANSPORT.

When the server generated by rpcgen is executed, it creates server handles for all the transports specified in NETPATH environment variable, or if it is unset, it creates server handles for all the visible transports from /etc/netconfig file. Note: the transports are chosen at runtime and not at compile time. When the server is self-started, it backgrounds itself by default. A special define symbol RPC_SVC_FG can be used to run the server process in the foreground. The second synopsis provides special features that allow for the creation of more sophisticated RPC servers. These features include support for user-provided #defines and RPC dispatch tables. The entries in the RPC dispatch table contain

- Pointers to the service routine corresponding to that procedure
- A pointer to the input and output arguments
- The size of these routines

A server can use the dispatch table to check authorization and then to execute the service routine; a client library may use it to deal with the details of storage management and xdr data conversion. The other three synopses shown in the preceding paragraph are used when one does not want to generate all the output files, but only a particular one. (Some examples of their usage is described in the "Example" subsection.) When rpcgen is executed with the -s option, it creates servers for that particular class of transports. When executed with the -n option, it creates a server for the transport specified by netid. If infile is not specified, rpcgen accepts the standard input. The C preprocessor, cc -E (see cc(1) for details), is run on the input file before it is actually interpreted by rpcgen. For each type of output file, rpcgen defines a special preprocessor symbol for use by the rpcgen programmer, as follows:

Any commands run are provided a few useful pieces of information in environment variables. The exact names are configurable, but the supplied defaults are

`$RSTART_CONTEXT`	The name of the context
`$RSTART_GLOBAL_CONTEXTS`	The global contexts directory
`$RSTART_LOCAL_CONTEXTS`	The local contexts directory
`$RSTART_GLOBAL_COMMANDS`	The global generic commands directory
`$RSTART_LOCAL_COMMANDS`	The local generic commands directory

`$RSTART_{GLOBAL,LOCAL}_CONTEXTS` should contain one special file, `@List`, which contains a list of the contexts in that directory in the format specified for `ListContexts`. The supplied version of `ListContexts` will `cat` both the global and local copies of `@List`.

Generic commands are searched for in several places: (defaults)

Per-user per-context directory	`$HOME/.rstart.commands/<context>`
Global per-context directory	`<XRoot>/lib/X11/rstart/commands/<context>`
Per-user all-contexts directory	`$HOME/.rstart.commands`
Global all-contexts directory	`(<XRoot>/lib/X11/rstart/commands)`

(Yes, this means you can't have an all-contexts generic command with the same name as a context. It didn't seem like a big deal.)

Each of these directories should have a file called `@List` that gives the names and descriptions of the commands in that directory in the format specified for `ListGenericCommands`.

CONFIGURATION KEYWORDS

There are several special `rstart` keywords defined for `rstartd` configuration. Unless otherwise specified, there are no defaults; related features are disabled in this case.

`Internal-Registries name...`—Gives a space-separated list of `MISC` registries that this system understands. (Registries other than these are accepted but generate a warning.)

`Internal-Local-Default relative_filename`—Gives the name (`$HOME relative`) of the per-user config file.

`INTERNAL-GLOBAL-CONTEXTS absolute_directory_name`—Gives the name of the system-wide contexts directory.

`INTERNAL-LOCAL-CONTEXTS relative_directory_name`—Gives the name (`$HOME` relative) of the per-user contexts directory.

`INTERNAL-GLOBAL-COMMANDS absolute_directory_name`—Gives the name of the system-wide generic commands directory.

`INTERNAL-LOCAL-COMMANDS relative_directory_name`—Gives the name (`$HOME` relative) of the per-user generic commands directory.

`INTERNAL-VARIABLE-PREFIX prefix`—Gives the prefix for the configuration environment variables `rstartd` passes to its kids.

`INTERNAL-AUTH-PROGRAM authscheme program argv[0] argv[1]...`—`Specifies` the program to run to set up authentication for the specified authentication scheme. `program argv[0] ...` gives the program to run and its arguments, in the same form as the `EXEC` keyword.

`INTERNAL-AUTH-INPUT authscheme`—Specifies the data to be given to the authorization program as its standard input. Each argument is passed as a single line. `$n`, where `n` is a number, is replaced by the nth argument to the `AUTH authscheme arg1 arg2 ...` line.

`INTERNAL-PRINT arbitrary text`—Prints its arguments as a Debug message. Mostly for `rstartd` debugging, but could be used to debug config files.

GROUPING COMMANDS

Braces ({, }) can be used to nest one address within another or to apply multiple commands to the same address:

```
[address][,address]{
command 1 command 2 ...
}
```

The opening { must end a line and the closing } must be on a line by itself.

COMMANDS

The maximum number of permissible addresses for each command is indicated in parentheses in the following list.

An argument denoted `text` consists of one or more lines of text. If `text` is longer than one line in length, then any newline characters must be hidden by preceding them with a backslash (\).

An argument denoted `read-filename` or `write-filename` must terminate the command line and must be preceded by exactly one space. Each `write-filename` is created before processing begins.

`(0)` An empty command is ignored.

`(0) #comment` The line is a comment and is ignored by sed. If however, the first script line in a script is of the form `#n`, then sed behaves as if the `-n` flag had been specified.

`(0) : label` Affix `label` to a line in the script for a transfer of control by `b` or `t` commands.

`(1) =` Write the current line number on the standard output as a line.

`(1)a\text` Append `text` following each line matched by the address on the standard output before reading the next input line.

`(2) b label` Unconditionally transfer control to the `:` command bearing the `label`. If no `label` is specified, then branch to the end of the script; no more commands are executed on the current pattern space.

`(2) c\text` Change the pattern space by replacing the selected pattern with `text`. When multiple lines are specified, all lines in the pattern space are replaced with a single copy of `text`. The end result is that the pattern space is deleted and no further editing commands can be applied to it.

`(2) d` Delete the pattern space, preventing the line from being passed to the standard output, and start the next cycle.

`(2) D` Delete the initial segment of the pattern space through the first newline and start the next cycle.

`(2) g` Replace the contents of the pattern space by the contents of the hold space.

`(2) G` Append a newline character followed by the contents of the hold space to the pattern space.

`(2) h` Replace the contents of the hold space by the contents of the pattern space.

`(2) H` Append a newline character followed by the contents of the pattern space to the hold space.

`(1) i\text` Insert `text` by writing it to the standard output.

`(2) l` Write the pattern space to standard output in a visually unambiguous form. Nonprinting characters are displayed as either three-digit octal values, preceded by a \, or as one of the following character constant escape sequences:

`\\`	Backslash
`\a`	Alert
`\b`	Backspace
`\f`	Form-feed
`\n`	Newline
`\r`	Carriage-return
`\t`	Tab
`\v`	Vertical tab

	Long lines are folded, with the point of folding indicated by a backslash (\) and a newline character. The end of every line is marked with a $.
(2) n	Copy the pattern space to the standard output. Replace the pattern space with the next line of input.
(2) N	Append the next line of input to the pattern space with an embedded newline. (The current line number changes.)
(2) p	Print the pattern space to the standard output.
(2) P	Copy the initial segment of the pattern space through the first newline to the standard output.
(1) q	Quit by transferring control to the end of the script and do not start a new cycle. The pattern space is still written to the standard output.
(2) r read-filename	Read the contents of read-filename. Place them on the output before reading the next input line.
(2) s/regularexpression/ replacement/flags	Substitute the replacement string for instances of the regular expression in the pattern space. Any character may be used instead of / (For a full description, see the explanation of replacement patterns in the "Regular Expressions" section of this manual page.) flags is zero or more of: n Substitute for just the nth occurrence of the regular expression. g Globally substitute for all nonoverlapping instances of the regular expression rather than just the first one. p Print the pattern space if a replacement was made.
w write-filename	Append the pattern space to write-filename if a replacement was made.
(2) t label	Branch to the : command bearing the label if any substitutions have been made since the most recent reading of an input line or execution of a t. If label is empty, branch to the end of the script.
(2) w write-filename	Append the pattern space to write-filename.
(2) x	Exchange the contents of the pattern and hold spaces.
(2) y/string1/string2/	Replace all occurrences of characters in string1 with the corresponding character in string2. The lengths of string1 and string2 must be equal. Any character other than '' or newline can be used instead of slash to delimit the strings. Within string1 and string2, the delimiter itself can be used as a literal character if it is preceded by a backslash.

DIAGNOSTICS

Command only uses one address—A command that takes one address had two addresses specified.

Command doesn't take any addresses—A command that takes no addresses had an address specified.

Extra characters after command—A command had extra text after the end.

Unexpected End-of-file—The end of a script was reached before it should have been. This usually occurs when a command is started, but not finished.

No previous regular expression—A metacharacter calling for a previous regular expression before any regular expressions were used.

Missing command—An address was not followed by a command.

Unknown command—A command was not one of the ones recognized by sed.

Unexpected ','—A command had a spurious comma after an address.

Multiple '!'s—More than one ! (exclamation point) was used in a command.

Unexpected g—A g character was given in a command without a preceding f.

Unexpected f—An f character was given in a command without a following g.

} doesn't want any addresses—} should be alone on a line.

`: doesn't want any addresses`—The `:` command should not be preceded by an address.

`Unterminated s command`—The replacement field of the `s` command should be completed with a `/` character.

`Multiple p options to s command` —The `p` option was given more than once in an `s` command.

`Multiple g options to s command`—The `g` option was given more than once in an `s` command.

`Multiple number options to s command`—More than one number option was given to an `s` command.

`Unknown option to s`—An unknown option was used for the `s` command. Maybe you shouldn't do that.

`Strings for y command are different lengths`—There should be a one-to-one mapping between strings for the `y` command.

`Missing ' ' before filename`—There was no space between an `r`, `w`, or `s///w` command, and the filename specified for that command.

`Hopelessly evil compiled in limit on number of open file. re-compile sed.`—An attempt was made to open too many files, no matter how you look at it.

SEE ALSO

`awk`(1), `ed`(1), `grep`(1), `perl`(1), `regex`(3)

HISTORY

A `sed` command appeared in version 7 AT&T UNIX.

STANDARDS

GNU `sed` is expected to be a superset of the IEEE Std 1003.2 (POSIX) specification.

CAVEATS

GNU `sed` uses the POSIX basic regular expression syntax. According to the standard, the meaning of some escape sequences is undefined in this syntax; notably `\|` and `\+`.

As in all GNU programs that use POSIX basic regular expressions, `sed` interprets these escape sequences as metacharacters. So, `x\+` matches one or more occurrences of `x`. `abc\|def` matches either `abc` or `def`.

This syntax may cause problems when running scripts written for other versions of `sed`. Some `sed` programs have been written with the assumption that `\|` and `\+` match the literal characters `|` and `+`. Such scripts must be modified by removing the spurious backslashes if they are to be used with GNU `sed`.

BUGS

It has long been noted that GNU `sed` is much slower than other implementations. The current bottleneck is the way `sed` reads and writes data files. It should read large blocks at a time (or even map files, where that is supported). When possible, it should avoid copying its input from one place in memory to another. Patches to make it do those things are welcome!

Version 2.05, December 1994

sessreg

`sessreg`—Manage `utmp`/`wtmp` entries for non-`init` clients

SYNOPSIS

```
sessreg [-w wtmp-file] [-u utmp-file] [-l line-name] [-h host-name]
[-s slot-number] [-x Xservers-file] [-t ttys-file] [-a] [-d] user-name
```

DESCRIPTION

`sessreg` is a simple program for managing `utmp`/`wtmp` entries for `xdm` sessions.

setterm

setterm—Set terminal attributes

SYNOPSIS

```
setterm [ -term terminal name ]

setterm [-reset ]

setterm [ -initialize ]

setterm [ -cursor [on¦off] ]

setterm [ -keyboard pc¦olivetti¦dutch¦extended ]

setterm [ -repeat [on¦off] ]

setterm [ -appcursorkeys [on¦off] ]

setterm [ -linewrap [on¦off] ]

setterm [ -snow [on¦off] ]

setterm [ -softscroll [on¦off] ]

setterm [ -defaults ]

setterm [ -foreground black¦red¦green¦yellow¦blue¦magenta¦cyan¦white¦default ]

setterm [ -background black¦red¦green¦yellow¦blue¦magenta¦cyan¦white¦default ]

setterm [ -ulcolor black¦grey¦red¦green¦yellow¦blue¦magenta¦cyan¦white ]

setterm [ -ulcolor bright red¦green¦yellow¦blue¦magenta¦cyan¦white ]

setterm [ -hbcolor black¦grey¦red¦green¦yellow¦blue¦magenta¦cyan¦white ]

setterm [ -hbcolor bright red¦green¦yellow¦blue¦magenta¦cyan¦white ]

setterm [ -inversescreen [on¦off] ]

setterm [ -bold [on¦off] ]

setterm [ -half-bright [on¦off] ]

setterm [ -blink [on¦off] ]

setterm [ -reverse [on¦off] ]

setterm [ -underline [on¦off] ]

setterm [ -store ]

setterm [ -clear [ all¦rest ] ]

setterm [ -tabs [tab1 tab2 tab3 ... ] ] where (tabn = 1-160)

setterm [ -clrtabs [ tab1 tab2 tab3 ... ] where (tabn = 1-160)
```

OPTIONS

`-verbose` Give some information about the SGI image file

BUGS

Probably

REFERENCES

SGI Image File Format documentation (draft v0.95) by Paul Haeberli (`paul@sgi.com`). Available via `ftp` at `sgi.com:graphics/SGIIMAGESPEC`.

SEE ALSO

`pnm`(5), `pnmtosgi`(1)

AUTHOR

Copyright© 1994 by Ingo Wilken (`Ingo.Wilken@informatik.uni-oldenburg.de`)

9 January 1994

shar

`shar`—Creat shell archives

SYNOPSIS

```
shar [ options ] file ...
shar -S [ options ]
```

DESCRIPTION

`shar` creates `shell` archives (or `shar` files) that are in text format and can be mailed. These files may be unpacked later by executing them with `/bin/sh`. The resulting archive is sent to standard out unless the `-o` option is given. A wide range of features provide extensive flexibility in manufacturing `shar`s and in specifying `shar` "smartness." Archives may be "vanilla" or comprehensive. This manual page reflects `shar` version 4.0.

OPTIONS

Options have a one-letter version starting with `-` or a long version starting with `--`. The exceptions are `--help` and `--version`, which do not have short versions. Options can be given in any order. Some options depend on each other: The `-o` option is required if the `-l` or `-L` option is used.

The `-n` option is required if the `-a` option is used.

See `-V` in the following list.

These are the available options:

`--version` Print the version number of the program on standard output, then immediately exit.

`--help` Print a help summary on standard output, then immediately exit.

`-V`, `--vanilla-operation` Produce vanilla `shar`s that rely only upon the existence of `sed` and `echo` in the unsharing environment. In addition, `if test` must also be supported if the `-X` option is used. The `-V` silently disables options offensive to the `network cop` (or `brown shirt`), but does warn you if it is specified with `-B`, `-z`, `-Z`, `-p`, or `-M` (any of which does or might require `uudecode`, `gzip` or `compress` in the unsharing environment).

showrgb

showrgb—Uncompile an RGB colorname database

SYNOPSIS

```
showrgb [ database ]
```

DESCRIPTION

The showrgb program reads an RGB colorname database compiled for use with the dbm database routines and converts it back to source form, printing the result to standard output. The default database is the one that X was built with, and may be overridden on the command line. Specify the database name without the .pag or .dir suffix.

FILES

<XRoot>/lib/X11/rgb Default database

shrinkfile

shrinkfile—Shrink a file on a line boundary

SYNOPSIS

```
shrinkfile [ -s size ][-v ] file...
```

DESCRIPTION

The shrinkfile program shrinks files to a given size, preserving the data at the end of the file. Truncation is performed on line boundaries, where a line is a series of bytes ending with a newline, \n. There is no line length restriction and files may contain any binary data.

Temporary files are created in the /tmp directory. The TMPDIR environment variable may be used to specify a different directory.

A newline will be added to any nonempty file that does not end with a newline. The maximum file size will not be exceeded by this addition.

By default, files are truncated to zero bytes. The -s flag may be used to change the maximum size. Because the program truncates only on line boundaries, the final size may be may be smaller then the specified maximum. The size parameter may end with a k, m, or g, indicating kilobyte (1024), megabyte (1048576) or gigabyte (1073741824) lengths. Uppercase letters are also allowed. The maximum file size is 2147483647 bytes.

If the -v flag is used, then shrinkfile will print a status line if a file was shrunk.

HISTORY

Written by Landon Curt Noll (chongo@toad.com) and Rich $alz (rsalz@uunet.uu.net) for InterNetNews.

sirtopnm

sirtopnm—Convert a Solitaire file into a portable anymap

SYNOPSIS

```
sirtopnm [sirfile]
```

`-b` Ignore leading blanks when finding sort keys in each line.

`-d` Sort in phone directory order; ignore all characters except letters, digits, and blanks when sorting.

`-f` Fold lowercase characters into the equivalent uppercase characters when sorting so that, for example, `b` is sorted the same way `B` is.

`-i` Ignore characters outside the ASCII range 040–0176 octal (inclusive) when sorting.

`-M` An initial string, consisting of any amount of whitespace, followed by three letters abbreviating a month name, is folded to uppercase and compared in the order `'JAN' < 'FEB' < ... < 'DEC'`. Invalid names compare low to valid names.

`-n` Compare according to arithmetic value an initial numeric string consisting of optional whitespace, an optional `-` sign, and zero or more digits, optionally followed by a decimal point and zero or more digits.

`-r` Reverse the result of comparison, so that lines with greater key values appear earlier in the output instead of later.

Other options are

`-o output-file` Write output to `output-file` instead of to the standard output. If `output-file` is one of the input files, `sort` copies it to a temporary file before sorting and writing the output to `output-file`.

`-t separator` Use character `separator` as the field separator when finding the sort keys in each line. By default, fields are separated by the empty string between a nonwhitespace character and a whitespace character. That is to say, given the input line `foo bar`, `sort` breaks it into fields `foo` and `bar`. The field separator is not considered to be part of either the field preceding or the field following it.

`-u` For the default case or the `-m` option, only output the first of a sequence of lines that compare equal. For the `-c` option, check that no pair of consecutive lines compares equal.

`+POS1 [-POS2]` Specify a field within each line to use as a sorting key. The field consists of the portion of the line starting at `POS1` and up to (but not including) `POS2` (or to the end of the line if `POS2` is not given). The fields and character positions are numbered starting with `0`.

`-k POS1[,POS2]` An alternate syntax for specifying sorting keys. The fields and character positions are numbered starting with `1`.

A position has the form `f.c`, where `f` is the number of the field to use and `c` is the number of the first character from the beginning of the field (for `+pos`) or from the end of the previous field (for `-pos`). The `.c` part of a position may be omitted, in which case it is taken to be the first character in the field. If the `-b` option has been given, the `.c` part of a field specification is counted from the first nonblank character of the field (for `+pos`) or from the first nonblank character following the previous field (for `-pos`).

A `+pos` or `-pos` argument may also have any of the option letters `Mbdfinr` appended to it, in which case the global ordering options are not used for that particular field. The `-b` option may be independently attached to either or both of the `+pos` and `-pos` parts of a field specification, and if it is inherited from the global options, it will be attached to both. If a `-n` or `-M` option is used, thus implying a `-b` option, the `-b` option is taken to apply to both the `+pos` and the `-pos` parts of a key specification. Keys may span multiple fields.

In addition, when GNU `join` is invoked with exactly one argument, the following options are recognized:

`--help` Print a usage message on standard output and exit successfully

`--version` Print version information on standard output, then exit successfully

COMPATIBILITY

Historical (BSD and System V) implementations of `sort` have differed in their interpretation of some options, particularly `-b`, `-f`, and `-n`. GNU `sort` follows the POSIX behavior, which is usually (but not always) like the System V behavior. According to POSIX, `-n` no longer implies `-b`. For consistency, `-M` has been changed in the same way. This may affect the meaning of character positions in field specifications in obscure cases. If this bites you, the fix is to add an explicit `-b`.

BUGS

The different meaning of field numbers depending on whether `-k` is used is confusing. It's all POSIX's fault!

`--help` Print a usage message and exit with a non-zero status.

`--version` Print version information on standard output then exit.

GNU Text Utilities

spottopgm

`spottopgm`—Convert SPOT satellite images to portable graymap format

SYNTAX

```
spottopgm [-1¦2¦3] [Firstcol Firstline Lastcol Lastline] inputfile
```

OPTIONS

`-1¦2¦3` Extract the given color from the SPOT image. The colors are infrared, visible light, and ultra-violet, although I don't know which corresponds to which number. If the image is in color, this will be announced on standard error. The default color is 1.

`Firstcol Firstline Lastcol Lastline` Extract the specified rectangle from the SPOT image. Most SPOT images are 3,000 lines long and 3,000 or more columns wide. Unfortunately, the SPOT format only gives the width and not the length. The width is printed on standard error. The default rectangle is the width of the input image by 3,000 lines.

DESCRIPTION

`spottopgm` converts the named `inputfile` into portable graymap format, defaulting to the first color and the whole SPOT image unless specified by the options.

INSTALLATION

You must edit the source program and either define `BIGENDIAN` or `LITTLEENDIAN`, and fix the `typedefs` for `uint32t`, `uint16t`, and `uint8t` appropriately.

BUGS

Currently, `spottopgm` doesn't determine the length of the input file; this would involve two passes over the input file. It defaults to 3,000 lines instead.

`spottopgm` could extract a three-color image (`ppm`), but I didn't feel like making the program more complicated than it is now. Besides, there is no one-to-one correspondence between red, green, blue, and infra-red, visible, and ultra-violet.

I've had only a limited number of SPOT images to play with, and therefore wouldn't guarantee that this will work on any other images.

AUTHOR

Warren Toomey (`wkt@csadfa.cs.adfa.oz.au`)

SEE ALSO

The rest of the `pbmplus` suite.

sputoppm

`sputoppm`—Convert an Atari uncompressed Spectrum file into a portable pixmap

subst

subst—Substitute definitions into file(s)

SYNOPSIS

```
subst [ -e editor ] -f substitutions victim ...
```

DESCRIPTION

subst makes substitutions into files, in a way that is suitable for customizing software to local conditions. Each victim file is altered according to the contents of the substitutions file.

The substitutions file contains one line per substitution. A line consists of two fields separated by one or more tabs. The first field is the name of the substitution, the second is the value. Neither should contain the character #, and use of text-editor metacharacters like & and \ is also unwise; the name in particular is best restricted to alphanumeric. A line starting with # is a comment and is ignored.

In the victim files, each line on which a substitution is to be made (a target line) must be preceded by a prototype line. The prototype line should be delimited in such a way that it will be taken as a comment by whatever program processes the file later. The prototype line must contain a prototype of the target line bracketed by =()< and >()=; everything else on the prototype line is ignored. subst extracts the prototype, changes all instances of substitution names bracketed by @< and >@ to their values, and then replaces the target line with the result.

Substitutions are done using the ed(1) editor, which must be found in either the /bin or /usr/bin directories. To specify a different executable use the -e flag.

EXAMPLE

If the substitutions file is

```
FIRST 111
SECOND 222
```

and the victim file is

```
x =2;
/* =()<y =@<FIRST>@+@<SECOND>@;>()= */
y =88 +99;
z =5;
```

then subst -f substitutions victim changes victim to

```
x =2;
/* =()<y =@<FIRST>@+@<SECOND>@;>()= */
y = 111 + 222;
z =5;
```

FILES

victimdir/substtmp.new	New version being built
victimdir/substtmp.old	Old version during renaming

SEE ALSO

sed(1)

DIAGNOSTICS

Complains and halts if it is unable to create its temporary files or if they already exist.

NOTE

You may want to redirect `stdout` to a file when you run `SuperProbe` (especially if your OS does not support Virtual Terminals on the console).

However, if you have any 8-bit cards installed, you should initially run `SuperProbe` as

```
SuperProbe -verbose -no16
```

(the `-verbose` option is included so you can see what `SuperProbe` is skipping).

Finer granularity can be obtained with an exclusion list, for example,

```
SuperProbe -verbose -excl 0x200,0x220-0x230,0x250
```

will not test for any device that uses port `0x200`, ports `0x220` through `0x230`, inclusive, or port `0x250`. If you have any 8-bit cards installed, you should add `-mask10` to the list of options.

To restrict the search to Western Digital, Tseng, and Cirrus chipset, run SuperProbe as follows:

```
SuperProbe -order WD,Tseng,Cirrus
```


BUGS

Probably a lot at this point. Please report any bugs or incorrect identifications to the author.

It is possible that SuperProbe can lock up your machine. Be sure to narrow the search by using the `-no16`, `-excl`, and `-mask10` options provided to keep SuperProbe from conflicting with other installed hardware.

SEE ALSO

The `vgadoc3.zip` documentation package by Finn Thoegersen, available in the MS-DOS archives of many FTP repositories.

Programmer's Guide to the EGA and VGA Cards, Second Edition, by Richard Ferraro.

AUTHOR

David E. Wexelblat (`dwex@xfree86.org`) with help from David Dawes (`dawes@xfree86.org`) and the XFree86 development team.

Version 2.2

tac

`tac`—Concatenate and print files in reverse

SYNOPSIS

```
tac [-br] [-s separator] [--before] [--regex] [--separator=separator]
[--help] [--version] [file...]
```

DESCRIPTION

This manual page documents the GNU version of `tac`. `tac` copies each given file, or the standard input if none are given or when a filename of `-` is encountered, to the standard output with the order of the records reversed. The records are separated by instances of a string, or a newline if none is given. By default, the separator string is attached to the end of the record that it follows in the file.

tcal

tcal—Runs the gcal program with the date of tomorrow's day

SYNOPSIS

```
tcal [ --help ¦ --version ] ¦ [ --shift=[+¦-]number ][Argument... ]
```

DESCRIPTION

tcal is a program that runs gcal with a date set one day ahead (equivalent to the --shift=1 option). All given arguments are passed unmodified to the gcal program. If the gcal program shall be called with a date other than tomorrow's date, this desired date can be selected by using the --shift=[+¦-]number option, in which [+¦-]number is the number of days the desired date is distant from the actual date. The --shift option must be given before all other arguments, which are passed to the gcal program. An exit status of 0 means all processing is successfully done; any other value means an error has occurred.

OPTIONS

--help	Print a usage message listing all available options, then exit successfully.
--version	Print the version number, then exit successfully.
--shift=[+¦-]number	Define the displacement in [+¦-]number days the desired date is distant from the actual date.

ENVIRONMENT

GCALPROG	The GCALPROG environment variable contains the filename of the executable gcal program, which is used by tcal. So tcal GCALPROG takes precedence over the filename gcal, which is burned-in during the compilation step of tcal.

COPYRIGHT

Any suggestions, improvements, extensions, bug reports, donations, proposals for contract work, and so forth are welcome! If you like this tool, I'd appreciate a postcard from you!

Enjoy it =8^)

AUTHOR

Thomas Esken (esken@uni-muenster.de)
m Hagenfeld 84
D-48147 Muenster; Germany
Phone : +49 251 232585

SEE ALSO

gcal(1)

tftp

tftp—Trivial file transfer program

SYNOPSIS

```
tftp [host]
```

DESCRIPTION

tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote host may be specified on the command line, in which case tftp uses host as the default host for future transfers. (See the connect command in the following section.)

COMMANDS

Once tftp is running, it issues the prompt:

```
tftp>
```

and recognizes the following commands:

`? command-name ...`	Print help information
`ascii`	Shorthand for "mode ascii"
`binary`	Shorthand for "mode binary"
`connect host-name [port]`	Set the host (and optionally port) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the connect command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the connect command; the remote host can be specified as part of the get or put commands.

`get filename`
`get remotename localname`
`get file1 file2 ... fileN`
	Get a file or set of files from the specified sources. Source can be in one of two forms: a filename on the remote host, if the host has already been specified; or a string of the form hosts:filename to specify both a host and filename at the same time. If the latter form is used, the last hostname specified becomes the default for future transfers.

`mode transfer-mode`	Set the mode for transfers; transfer-mode may be ascii or binary. The default is ascii.

`put file`
`put localfile remotefile`
`put file1 file2 ...`
`fileN remote-directory`
	Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a filename on the remote host, if the host has already been specified; or a string of the form hosts:filename to specify both a host and filename at the same time. If the latterform is used, the hostname specified becomes the default for future transfers. If the remote-directory form is used, the remote host is assumed to be a UNIX machine.

`quit`	Exit tftp . An end-of-file also exits.
`rexmt retransmission-timeout`	Set the per-packet retransmission time-out, in seconds.
`status`	Show current status.
`timeout total-transmission-timeout`	Set the total transmission time-out, in seconds.
`trace`	Toggle packet tracing.
`verbose`	Toggle verbose mode.

BUGS

Because there is no user-login or validation within the TFTP protocol, the remote site will probably have some sort of file-access restrictions in place. The exact methods are specific to each site and therefore difficult to document here.

SEE ALSO

pnmtotiff(1), pnm(5)

BUGS

This program is not self-contained. To use it you must fetch the TIFF Software package listed in the OTHER.SYSTEMS file and configure pbmplus to use libtiff. See the pbmplus Makefile for details on this configuration.

AUTHOR

Derived by Jef Poskanzer from tif2ras.c, which is copyright© 1990 by Sun Microsystems, Inc. Author Patrick J. Naughton (naughton@wind.sun.com).

13 January 1991

tin, rtin, cdtin, tind

tin, rtin, cdtin, tind—A Netnews reader

SYNOPSIS

tin/rtin/cdtin/tind [options][newsgroup]

DESCRIPTION

tin is a full-screen easy-to-use Netnews reader. It can read news locally (/usr/spool/news) or remotely (rtin or tin -r option) via an NNTP (Network News Transport Protocol) server. cdtin can read news locally and news archived on CD-ROM. It will automatically utilize nov (news overview)-style index files if available locally or via the nntp xover command.

tin has five separate levels of operation: group selection level, spooldir selection level, group level, thread level and article level. Use the h (help) command to view a list of the commands available at a particular level.

On startup, tin will show a list of the newsgroups found in $HOME/.newsrc. An arrow -> or highlighted bar will point to the first newsgroup. Move to a group by using the terminal arrow keys (terminal-dependent) or j and k. Use PgUp/PgDn (terminal-dependent) or Ctrl-U and Ctrl-D to page up/down. Enter a newsgroup by pressing Return.

The Tab key advances to the next newsgroup with unread articles and enters it.

OPTIONS

-c	Create/update index files for every group in $HOME/.newsrc or file specified by -f option and mark all articles as read.
-f file	Use the specified file of subscribed to newsgroups in place of $HOME/.newsrc.
-h	Help listing all command-line options.
-H	Brief introduction to tin that is also shown the first time it is started.
-I dir	Directory in which to store newsgroup index files. Default is $HOME/.tin/.index.
-m dir	Mailbox directory to use. Default is $HOME/Mail.
-M user	Mail unread articles to specified user for later reading. For more information read the “Automatic Mailing and Saving New News” section later in this manual page.
-n	Only load groups from the active file that are also subscribed to in the users .newsrc. This allows a noticeable speedup when connecting via a slow line.
-p program	Print program with options.
-q	Quick start without checking for new newsgroups.
-P	Purge group index files of articles that no longer exist. Care should be taken when using this command as it starts each and every article in each group that is accessed. On a low-speed connection, this can have an undesirable effect and it also knocks the hell out of your file system.

`-r`	Read news remotely from the default NNTP server specified in the environment variable `NNTPSERVER` or contained in the file `/etc/nntpserver`.
`-R`	Read news saved by `-S` option (not yet implemented).
`-s dir`	Save articles to directory. Default is `$HOME/News`.
`-S`	Save unread articles for later reading by `-R` option. For more information, see "Automatic Mailing and Saving New News."
`-u`	Create/update index files for every group in `$HOME/.newsrc` or file specified by `-f` option. This option is disabled if `tin` retrieves its index files via an NNTP server.
`-U`	Start `tin` in the background to update index files while reading news in the foreground. This option is disabled if `tin` retrieves its index files via an NNTP server.
`-v`	Verbose mode for `-c`, `-M`, `-S`, `-u`, and `-Z` options.
`-w`	Quick mode to post an article and then exit.
`-z`	Only start `tin` if there is any new/unread news. If there is news, `tin` will position cursor at first group with unread news. Useful for putting in login file.
`-Z`	Check if there is any new/unread news and exit with appropriate status. If `-v` option is specified, the number of unread articles in each group is printed. An exit code `0` indicates no news, `1` that an error occurred, and `2` that new/unread news exists. Useful for writing scripts.

`tin` can also dynamically change its options by the `M` menu command. Any changes are written to `$HOME/.tin/tinrc`.

The index daemon version of `tin`, only supports the `-f`, `-h`, `-l` and `-u` options.

INDEX FILES

In order to keep track of threads, `tin` maintains an index for each newsgroup. There are a number of methods in which index files can be created and updated.

The simplest method is that each user creates/updates his or her own index files that are stored in `$HOME/.tin/.index`. This has the advantage that any user can compile and install `tin`, but the disadvantage is that each user is going to be creating duplicate files and using precious disk space. A good way to keep index files updated is by doing a `tin -U` that will update index files in the background while you are reading news in the foreground. You can also update index files via the system batcher `cron` with the `-u` option: `30 6 ***/usr/local/bin/tin -u`.

A slightly better method is to set `tin` `setuid` news and have all index files created and updated in the news spool directory (that is, `/usr/spool/news/.index`). This has the advantage that there will only be one copy of the index files on each machine on your network, but the disadvantage is that you will have `tin` running `setuid` news.

A better method is to install the `tind` index file updating daemon and have it create and update index files for all groups in your active file at regular intervals in the news spool directory (`/usr/spool/news/.index`). This has the advantage that there will only be one copy of the index files on each machine on your network, and `tin` must not be `setuid` news, but the disadvantage is that you will have to have news permissions to install `tind` and root permissions to install an entry in the `cron` batcher system to have `tind` regularly update index files.

The best method is to install the `tind` index file updating daemon on your NNTP server and have it create and update index files for all groups in your active file at regular intervals in the news spool directory (`/usr/spool/news/.index`). This has the advantage that there will only be one copy of the index files on the NNTP server for the whole of your network, but the disadvantage is that you will have to install my NNTP server patches to allow `tin` to retrieve index file from your NNTP server and you must install an entry in the `cron` batcher system to have `tind` regularly update index files. (This is the method we use on our network of 40 to 50 machines and we have not had any problems.)

Entering a group the first time tends to be slow because the index file must be built from scratch unless the `tind` update daemon is being used. To alleviate the slowness, start `tin` to create all index files for the groups you subscribe to with `tin -u -v` and go for a coffee. Subsequent readings of a group will cause incremental updating of the index file.

```
<Author><Organization>
<Article body>
i.e.,
 24 Jul 15:20:03 GMT      alt.sources          Thread 1 of 2
 Article 452              Bnews sources?       3 responses
 iain@anl433.uucp         Organization name
```

COMMON MOVING KEYS

The following table shows the common keys/commands for moving at all five levels within `tin`:

Beginning of list/article	`Home 1` (`^R` or `g` at article level)
End of list/article	`End $` (also `G` at article level)
Page up	`PgUp ^U` or `^B` or `b`
Page down	`PgDn ^D` or `^F` or `<SPACE>`
Line up	`Up arrow k` (not at article level)
Line down	`Down arrow j` (not at article level)

COMMON EDITING COMMANDS

An `emacs`-style editing package allows the easy editing of input strings. A history list allows the easy reuse of previously entered strings. The following commands are available when editing a string:

`^A`, `^E`	Move to beginning or end of line, respectively.
`^F`, `^B`	Nondestructive move forward or back one location, respectively.
`^D`	Delete the character currently under the cursor, or send `EOF` if no characters are in the buffer.
`^H`, `<DEL>`	Delete character left of the cursor.
`^K`	Delete from cursor to end of line.
`^P`, `^N`	Move through history, previous and next, respectively.
`^L`, `^R`	Redraw the current line.
`<CR>`	Places line on history list if nonblank, appends newline, and returns to the caller.
`<ESC>`	Aborts the present editing operation.

NEWSGROUP SELECTION COMMANDS

`4`	Select group 4.
`^K`	Delete current group from `$HOME/.newsrc` file.
`^L`	Redraw page.
`^R`	Reset `$HOME/.newsrc` file.
`<CR>`	Read current group.
`<TAB>`	View next group with unread news. Will wrap around to the beginning of the group selection list looking for unread groups.
`B`	Mail a bug report or comment to the author. This is the best way to get bugs fixed and features added/changed.
`c`	Mark current group as all read with confirmation and go to next group in group selection list.
`C`	Mark current group as all read and go to next unread group in group selection list.
`d`	Toggle display to show just the group name or the group name and the group's description.
`g`	Choose a new group by name. The position of the group within the group list will also be asked for. When `1` is entered, the new group will be the first group in the displayed list; when `8` is entered, the group will be the eighth group in the list; and so on. When `$` is entered, the group will be the last group displayed.

GROUP INDEX COMMANDS

`4`	Select article 4.
`^K`	Kill current article (for more information, see the "Automatic Kill and Selection" section later in this manual page).
`^L`	Redraw page.
`<CR>`	Read current article.
`<TAB>`	View next unread article or group.
`a`	Author forward search.
`A`	Author backward search.
`c`	Mark all articles as read with confirmation.
`C`	Mark all articles as read and go to next group with unread news.
`d`	Toggle display to show just the subject or the subject and author.
`g`	Choose a new group by name.
`h`	Help screen of group index commands.
`H`	Toggle the display of help mini-menu at the bottom of the screen.
`I`	Toggle inverse video.
`K`	Mark article/thread as read in advance to next unread article/thread.
`l`	List the author's last response in current thread and enter thread selection level.
`m`	Mail current article/thread/auto-selected (hot) articles/articles matching pattern/tagged articles to someone.
`M`	User-configurable Options menu (for more information see "Global Options Menu" section).
`n`	Go to next group.
`N`	Go to next unread article.
`o`	Output current article/thread/autoselected (hot) articles/articles matching pattern/tagged articles to printer.
`p`	Go to previous group.
`P`	Go to previous unread article.
`q`	Return to previous level.
`Q`	Quit `tin`.
`s`	Save current article/thread/autoselected (hot) articles/articles matching pattern/tagged articles to file/files/ mailbox. To save to a mailbox, enter `=` or `=mailbox` when asked for filename to save to. To save in `<newsgroup name>/<filename>` format, enter `+filename`. Environment variables are allowed within a filename (for example, `$SOURCES/dir/filename`).
`t`	Tag current article/thread for mailing (`m`)/piping (`|`)/printing (`o`)/saving (`s`)/crossposting (`x`).
`u`	Toggle display to show all articles as unthreaded or threaded.
`U`	Untag all articles that were tagged.
`v`	Print `tin` version information.
`w`	Post an article to current group.
`W`	List articles posted by user. The date posted, the newsgroup, and the subject are listed.
`x`	Crosspost already posted current article/thread/autoselected (hot) articles/articles matching pattern/tagged articles to another newsgroup(s). Useful for reposting from global to local newsgroups.
`X`	Mark all unread articles that have not been selected as read, redo screen to reflect changes, and put index at the first thread to begin reading. Pressing `X` again will toggle back to the way it was before. See `¯` command for clearing the toggle effect.
`z`	Mark current article as unread.
`Z`	Mark current thread as unread.

newsgroups. Once entered, the user can abort the command and not save the `kill` description, edit the `kill` file, or save the `kill` description.

On starting `tin`, the user's `kill` file `$HOME/.tin/kill` is read and on entering a newsgroup any `kill` or `select` descriptions are applied.

Articles that match a `kill` description are marked killed and are not displayed. Articles that match an autoselect description are marked with an `*` when displayed.

POSTING ARTICLES

`tin` allows posting of articles, follow-up to already posted articles, and replying direct through mail to the author of an article.

Use the `w` command to post an article to a newsgroup. After entering the post subject, the default editor (such as `vi`) or the editor specified by the `$VISUAL` environment variable will be started and the article can be entered. To cross-post articles, simply add a comma and the name of the newsgroup(s) to the end of the "Newsgroups:" line at the beginning of the article. After saving and exiting the editor, you are asked if you wish to abort posting the article, edit the article again or post the article to the specified newsgroup(s).

Use the `W` command to display a history of the articles you have posted. The date the article was posted, which newsgroups the article was posted to, and the article's subject line are displayed.

Use the `f/F` command to post a follow-up article to an already posted article. The `f` command will copy the text of the original article into the editor. The editing procedure is the same as when posting an article with the `w` command.

Use the `r/R` command to reply direct through mail to the author of an already posted article. The `r` command will copy the text of the original article into the editor. The editing procedure is the same as when posting an article with the `w` command. After saving and exiting the editor, you are asked if you wish to abort sending the article, edit the article again, or send the article to the author.

CUSTOMIZING THE ARTICLE QUOTE STRING

When posting a follow-up to an article or replying direct to the author of an article via e-mail, the text of the article can be quoted. The beginning of the quoted text can contain information about the quoted article (for example, the Name and the Message ID of the article). To allow for different situations, certain information from the article can be used in the quoted string. The following variables are expanded if found in the `tinrc` variables `mail_quote_format=` or `news_quote_format=`:

`%A`	Address (e-mail)
`%D`	Date
`%F`	Full address (`%N (%A)`)
`%G`	Groupname
`%M`	Message ID
`%N`	Name of user

For example,

```
mail_quote_format=On %D in %G you wrote:
news_quote_format=In %M, %F wrote:
```

would expand when used to:

```
On 21 Jul 1992 09:45:51 -0400 in alt.sources you wrote:
In <abcINN123@anl433.uucp>, Iain Lea (iain@erlm.siemens.de) wrote:
```

MAILING, PIPING, PRINTING, REPOSTING, AND SAVING ARTICLES

The command interface to mail (`m`), pipe (`|`), print (`o`), crosspost (`x`) and save (`s`) articles is the same for ease of use.

The initial command will ask you to select which article, thread, hot (autoselected) `regex` pattern, tagged articles you wish to mail, pipe, and so on.

AUTHOR

Iain Lea (`iain.lea@erlm.siemens.de`)

Version 1.2 PL2

tload

`tload`—Graphic representation of system load average

SYNOPSIS

```
tload [-s scale] [-d delay] [tty]
```

DESCRIPTION

`tload` prints a graph of the current system load average to the specified `tty` (or the `tty` of the `tload` process if none is specified).

OPTIONS

The `-s scale` option allows a vertical scale to be specified for the display (in characters between graph ticks); thus, a smaller value represents a larger scale, and vice versa.

The `-d delay` sets the delay between graph updates in seconds.

FILES

`/proc/loadavg` Load average information

SEE ALSO

`ps`(1), `top`(1), `uptime`(1), `w`(1)

BUGS

The `-d delay` option sets the time argument for an `alarm`(2); if `-d 0` is specified, the alarm is set to `0`, which will never send the `SIGALRM` and update the display.

AUTHORS

Branko Lankester, David Engel (`david@ods.com`), and Michael K. Johnson (`johnsonm@sunsite.unc.edu`)

Cohesive Systems, 20 March 1993

top

`top`—Display top CPU processes

SYNOPSIS

```
top [-][dde|ay][q][S][s][i]
```

DESCRIPTION

`top` provides an ongoing look at processor activity in real time. It displays a listing of the most CPU-intensive tasks on the system, and can provide an interactive interface for manipulating processes.

`IconManagerFont string`
This variable specifies the font to be used when displaying icon manager entries. The default is `variable`.

`IconManagerForeground string [{ win-list }]`
This variable specifies the foreground color to be used when displaying icon manager entries, and may be specified only inside of a `Color`, `Grayscale`, or `Monochrome` list. The optional `win-list` is a list of window names and colors so that per-window colors may be specified. See the `BorderColor` variable for a complete description of the `win-list`. The default is `black`.

`IconManagerGeometry string [ columns ]`
This variable specifies the geometry of the icon manager window. The `string` argument is standard geometry specification that indicates the initial full size of the icon manager. The icon manager window is then broken into `columns` pieces and scaled according to the number of entries in the icon manager. Extra entries are wrapped to form additional rows. The default number of columns is `1`.

`IconManagerHighlight string [{ win-list }]`
This variable specifies the border color to be used when highlighting the icon manager entry that currently has the focus, and can only be specified inside of a `Color`, `Grayscale`, or `Monochrome` list. The optional `win-list` is a list of window names and colors so that per-window colors may be specified. See the `BorderColor` variable for a complete description of the `win-list`. The default is `black`.

`IconManagers { iconmgr-list }`
This variable specifies a list of icon managers to create. Each item in the `iconmgr-list` has the following format:

```
"winname"["iconname"]
geometry  columns
```

where `winname` is the name of the windows that should be put into this icon manager, `iconname` is the name of that icon manager window's icon, `geometry` is a standard geometry specification, and `columns` is the number of columns in this icon manager as described in `IconManagerGeometry`. For example:

```
IconManagers
{
"XTerm"  "=300x5+800+5" 5
"myhost"  "=400x5+100+5" 2
}
```

Clients whose name or class is `XTerm` will have an entry created in the `XTerm` icon manager. Clients whose name was `myhost` would be put into the `myhost` icon manager.

`IconManagerShow{ win-list }` This variable specifies a list of windows that should appear in the icon manager. When used in conjunction with the `IconManagerDontShow` variable, only the windows in this list will be shown in the icon manager.

`IconRegion geomstring vgrav hgrav gridwidth gridheight`
This variable specifies an area on the root window in which icons are placed if no specific icon location is provided by the client. The `geomstring` is a quoted string containing a standard geometry specification. If more than one `IconRegion` line is given, icons will be put into the succeeding icon regions when the first is full. The `vgrav` argument should be either `North` or `South` and is used to control whether icons are first filled in from the top or bottom of the icon region. Similarly, the `hgrav` argument should be either `East` or `West` and is used to control whether icons should be filled in from the left or from the right. Icons are laid out within the region in a grid with cells `gridwidth` pixels wide and `gridheight` pixels high.

`Icons { win-list }`
This variable specifies a list of window names and the `bitmap` filenames that should be used as their icons. For example,

```
Icons
{
"XTerm"  "xterm.icon"
"xfd"  "xfd_icon"
}
```

Windows that match `"XTerm"` and would not be iconified by unmapping would try to use the icon bitmap in the file `xterm.icon`. If `ForceIcons` is specified, this bitmap will be used even if the client has requested its own icon pixmap.

`InterpolateMenuColors` This variable indicates that menu entry colors should be interpolated between entry specified colors. In this example,

```
Menu "mymenu"
{
"Title" ("black":"red") f.title
"entry1" f.nop
"entry2" f.nop
"entry3" ("white":"green") f.nop
"entry4" f.nop
"entry5" ("red":"white") f.nop
}
```

the foreground colors for "`entry1`" and "`entry2`" will be interpolated between black and white, and the background colors between red and green. Similarly, the foreground for "`entry4`" will be halfway between white and red, and the background will be halfway between green and white.

`MakeTitle { win-list }` This variable specifies a list of windows on which `title-bar` should be placed and is used to request titles on specific windows when `NoTitle` has been set.

`MaxWindowSize string` This variable specifies a geometry in which the width and height give the maximum size for a given window. This is typically used to restrict windows to the size of the screen. The default width is `32767`—screen width. The default height is `32767`—screen height.

`MenuBackground string` This variable specifies the background color used for menus, and can only be specified inside of a `Color` or `Monochrome` list. The default is `white`.

`MenuFont string` This variable specifies the font to use when displaying menus. The default is `variable`.

`MenuForeground string` This variable specifies the foreground color used for menus and can only be specified inside of a `Color`, `Grayscale`, or `Monochrome` list. The default is `black`.

`MenuShadowColor string` This variable specifies the color of the shadow behind pull-down menus and can only be specified inside of a `Color`, `Grayscale`, or `Monochrome` list. The default is `black`.

`MenuTitleBackground string` This variable specifies the background color for `f.title` entries in menus, and can only be specified inside of a `Color`, `Grayscale`, or `Monochrome` list. The default is `white`.

`MenuTitleForeground string` This variable specifies the foreground color for `f.title` entries in menus and can only be specified inside of a `Color` or `Monochrome` list. The default is `black`.

`Monochrome { colors }` This variable specifies a list of color assignments that should be made if the screen has a depth of `1`. See the description of `Colors`.

`MoveDelta pixels` This variable specifies the number of pixels the pointer must move before the `f.move` function starts working. Also see the `f.deltastop` function. The default is zero pixels.

`NoBackingStore` This variable indicates that `twm`'s menus should not request backing store to minimize repainting of menus. This is typically used with servers that can repaint faster than they can handle backing store.

`NoCaseSensitive` This variable indicates that case should be ignored when sorting icon names in an icon manager. This option is typically used with applications that capitalize the first letter of their icon name.

`NoDefaults` This variable indicates that `twm` should not supply the default titlebuttons and bindings. This option should only be used if the startup file contains a completely new set of bindings and definitions.

`NoGrabServer` This variable indicates that `twm` should not grab the server when popping up menus and moving opaque windows.

`NoHighlight [{ win-list }]` This variable indicates that borders should not be highlighted to track the location of the pointer. If the optional `win-list` is given, highlighting will only be disabled for those windows. When the border is highlighted, it will be drawn in the current `BorderColor`. When the border is not highlighted, it will be stippled with a gray pattern using the current `BorderTileForeground` and `BorderTileBack-ground` colors.

`NoIconManagers` This variable indicates that no icon manager should be created.

`NoMenuShadows` This variable indicates that menus should not have drop shadows drawn behind them. This is typically used with slower servers because it speeds up menu drawing at the expense of making the menu slightly harder to read.

`NoRaiseOnDeiconify` This variable indicates that windows that are deiconified should not be raised.

`NoRaiseOnMove` This variable indicates that windows should not be raised when moved. This is typically used to allow windows to slide underneath each other.

`NoRaiseOnResize` This variable indicates that windows should not be raised when resized. This is typically used to allow windows to be resized underneath each other.

`NoRaiseOnWarp` This variable indicates that windows should not be raised when the pointer is warped into them with the `f.warpto` function. If this option is set, warping to an occluded window may result in the pointer ending up in the occluding window instead of the desired window, which causes unexpected behavior with `f.warpring`.

`NoSaveUnders` This variable indicates that menus should not request save-unders to minimize window repainting following menu selection. It is typically used with displays that can repaint faster than they can handle save unders.

`NoStackMode [{ win-list }]` This variable indicates that client window requests to change stacking order should be ignored. If the optional `win-list` is given, only requests on those windows will be ignored. This is typically used to prevent applications from relentlessly popping themselves to the front of the window stack.

`NoTitle [{ win-list }]` This variable indicates that windows should not have title-bars. If the optional `win-list` is given, only those windows will not have titlebars. `MakeTitle` may be used with this option to force titlebars to be put on specific windows.

`NoTitleFocus` This variable indicates that `twm` should not set keyboard input focus to each window as it is entered. Normally, `twm` sets the focus so that focus and key events from the titlebar and icon managers are delivered to the application. If the pointer is moved quickly and `twm` is slow to respond, input can be directed to the old window instead of the new. This option is typically used to prevent this input lag and to work around bugs in older applications that have problems with focus events.

`NoTitleHighlight [{ win-list }]` This variable indicates that the highlight area of the titlebar, which is used to indicate the window that currently has the input focus, should not be displayed. If the optional `win-list` is given, only those windows will not have highlight areas. This and the `SqueezeTitle` options can be set to substantially reduce the amount of screen space required by titlebars.

`OpaqueMove` This variable indicates that the `f.move` function should actually move the window instead of just an outline so that the user can immediately see what the window will look like in the new position. This option is typically used on fast displays (particularly if `NoGrabServer` is set).

`Pixmaps { pixmaps }` This variable specifies a list of pixmaps that define the appearance of various images. Each entry is a keyword indicating the pixmap to set, followed by a string giving the name of the `bitmap` file. The following pixmaps may be specified: `0 Pixmaps { TitleHighlight "gray1" }`

The default for `TitleHighlight` is to use an even stipple pattern.

`Priority priority` This variable sets `twm`'s priority. `priority` should be an unquoted, signed number (for example, `999`). This variable has an effect only if the server supports the SYNC extension.

`RandomPlacement` This variable indicates that windows with no specified geometry should be placed in a pseudorandom location instead of having the user drag out an outline.

User-defined functions contain the name by which they are referenced in calls to `f.function` and a list of other functions to execute. For example,

```
Function "move-or-lower" { f.move f.deltastop f.lower }
Function "move-or-raise" { f.move f.deltastop f.raise }
Function "move-or-iconify" { f.move f.deltastop f.iconify }
Function "restore-colormap" { f.colormap "default" f.lower }
```

The function name must be used in `f.function` exactly as it appears in the function specification.

In the following descriptions, if the function is said to operate on the selected window, but is invoked from a root menu, the cursor will be changed to the Select cursor and the next window to receive a button press will be chosen:

`! string` — This is an abbreviation for `f.exec string`.

`f.autoraise` — This function toggles whether or not the selected window is raised whenever entered by the pointer. See the description of the variable `AutoRaise`.

`f.backiconmgr` — This function warps the pointer to the previous column in the current icon manager, wrapping back to the previous row if necessary.

`f.beep` — This function sounds the keyboard bell.

`f.bottomzoom` — This function is similar to the `f.fullzoom` function, but resizes the window to fill only the bottom half of the screen.

`f.circledown` — This function lowers the topmost window that occludes another window.

`f.circleup` — This function raises the bottommost window that is occluded by another window.

`f.colormap string` — This function changes the colormaps (obtained from the `WM_COLORMAP_WINDOWS` property on the window) that twm will display when the pointer is in this window. The argument `string` may have one of the following values: `next`, `prev`, and `default`. It should be noted here that in general, the installed colormap is determined by keyboard focus. A pointer-driven keyboard focus will install a private colormap upon entry of the window owning the colormap. Using the click-to-type model, private colormaps will not be installed until the user clicks on the target window.

`f.deiconify` — This function deiconifies the selected window. If the window is not an icon, this function does nothing.

`f.delete` — This function sends the `WM_DELETE_WINDOW` message to the selected window if the client application has requested it through the `WM_PROTOCOLS` window property. The application is supposed to respond to the message by removing the indicated window. If the window has not requested `WM_DELETE_WINDOW` messages, the keyboard bell will be rung, indicating that the user should choose an alternative method. Note this is very different from `f.destroy`. The intent here is to delete a single window, not necessarily the entire application.

`f.deltastop` — This function allows a user-defined function to be aborted if the pointer has been moved more than `MoveDelta` pixels. See the example definition given for `Function "move-or-raise"` at the beginning of the section.

`f.destroy` — This function instructs the `X` server to close the display connection of the client that created the selected window. This should only be used as a last resort for shutting down runaway clients. See also `f.delete`.

`f.downiconmgr` — This function warps the pointer to the next row in the current icon manger, wrapping to the beginning of the next column if necessary.

`f.exec string` — This function passes the argument `string` to `/bin/sh` for execution. In multiscreen mode, if `string` starts a new `X` client without giving a display argument, the client will appear on the screen from which this function was invoked.

`f.focus` — This function toggles the keyboard focus of the server to the selected window, changing the focus rule from pointer-driven if necessary. If the selected window already was focused, this function executes an `f.unfocus`.

`-q, --list` Display the status of commands, executions, and conversations for all remote systems for which commands or executions are queued. The `-s, --system`, `-S, --not-system`, `-o, --older-than`, `-y`, and `--younger-than` options may be used to restrict the systems that are listed. Systems for which no commands or executions are queued will never be listed.

`-m, --status` Display the status of conversations for all remote systems.

`-p, --ps` Display the status of all processes holding `uucp` locks on systems or ports.

`-i, --prompt` For each listed job, prompt whether to kill the job or not. If the first character of the input line is `y` or `Y`, the job will be killed.

`-K, --kill-all` Automatically kill each listed job. This can be useful for automatic cleanup scripts, in conjunction with the `--mail` and `--notify` options.

`-R, --rejuvenate-all` Automatically rejuvenate each listed job. This may not be used with `--kill-all`.

`-M, --mail` For each listed job, send mail to the UUCP administrator. If the job is killed (due to `--kill-all` or `--prompt` with an affirmative response), the mail will indicate that. A comment specified by the `--comment` option may be included. If the job is an execution, the initial portion of its standard input will be included in the mail (`--mail-lines` controls the number of lines to include may be set with the `--mail-line` option; the default is `100`). If the standard input contains `null` characters, it is assumed to be a binary file and is not included.

`-N, --notify` For each listed job, send mail to the user who requested the job. The mail is identical to that sent by the `-M` or `--mail` options.

`-W, --comment` Specify a comment to be included in mail sent with the `-M, --mail`, `-N`, or `--notify` options.

`-Q, --no-list` Do not actually list the job, but only take any actions indicated by the `-i, --prompt`, `-K, --kill-all`, `-M, --mail`, `-N`, or `--notify` options.

`-x type, --debug type` Turn on particular debugging types. The following types are recognized: `abnormal`, `chat`, `handshake`, `uucp-proto`, `proto`, `port`, `config`, `spooldir`, `execute`, `incoming`, `outgoing`. Only `abnormal`, `config`, `spooldir`, and `execute` are meaningful for `uustat`.

Multiple types may be given, separated by commas, and the `--debug` option may appear multiple times. A number may also be given, which will turn on that many types from the foregoing list; for example, `--debug 2` is equivalent to `--debug abnormal,chat`.

`-I file, --config file` Set configuration file to use. This option may not be available, depending upon how `uustat` was compiled.

`-v, --version` Report version information and exit.

`--help` Print a help message and exit.

EXAMPLES

`uustat -all` Display status of all jobs. A sample output line is as follows:

```
bugsA027h bugs ian 04-01 13:50 Executing rmail ian@airs.com (sending 1283 bytes)
```

The format is

```
jobid system user queue-date command (size)
```

The `jobid` may be passed to the `--kill` or `--rejuvenate` options. The size indicates how much data is to be transferred to the remote system, and is absent for a file receive request. The `--system`, `--not-system`, `--user`, `--not-user`, `--command`, `--not-command`, `--older-than`, and `--younger-than` options may be used to control which jobs are listed.

`uustat -executions` Display status of queued up execution requests. A sample output line is as follows:

```
bugs bugs!ian 05-20 12:51 rmail ian
```

The format is

```
system requestor queue-date command
```

The `--system`, `--not-system`, `--user`, `--not-user`, `--command`, `--not-command`, `--older-than`, and `--younger-than` options may be used to control which requests are listed.

OPTIONS

`-b`	Search only for binaries.
`-m`	Search only for manual sections.
`-s`	Search only for sources.
`-u`	Search for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus `whereisnn-mnn-unn*` asks for those files in the current directory which have no documentation.
`-B`	Change or otherwise limit the places where `whereis` searches for binaries.
`-M`	Change or otherwise limit the places where `whereis` searches for manual sections.
`-S`	Change or otherwise limit the places where `whereis` searches for sources.
`-f`	Terminate the last directory list and signals the start of filenames; *must* be used when any of the `-B`, `-M`, or `-S` options are used.

EXAMPLE

Find all files in `/usr/bin` that are not documented in `/usr/man/man1` with source in `/usr/src`:

```
example% cd /usr/bin
example% whereis -u -M /usr/man/man1 -S /usr/src -f *
```

FILES

```
/{bin,sbin,etc}
/usr/{lib,bin,old,new,local,games,include,etc,src,man,sbin,
X386, TeX, g++-include}
/usr/local/{X386,TeX,X11,include,lib,man,etc,bin,games,emacs}
```


SEE ALSO

`chdir`(2V)

BUGS

Since `whereis` uses `chdir`(2V) to run faster, `pathnames` given with the `-M`, `-S`, or `-B` must be full; that is, they must begin with a `/`.

8 May 1994

write

`write`—Send a message to another user

SYNOPSIS

```
write user [ttyname]
```

DESCRIPTION

`write` allows you to communicate with other users by copying lines from your terminal to theirs.

When you run the `write` command, the user you are writing to gets a message of the form:

```
Message from yourname@yourhost on yourtty at hh:mm ...
```

Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, he or she must run `write` as well.

When you are done, type an end-of-file or interrupt character. The other user will see the message `EOF`, indicating that the conversation is over.

X DEFAULTS

There are no X defaults used by this program.

SEE ALSO

X(1), x11perf(1)

AUTHORS

Mark Moraes wrote the original scripts to compare servers. Joel McCormack just munged them together a bit.

X Version 11 Release 6

xargs

xargs—Build and execute command lines from standard input

SYNOPSIS

```
xargs [-0prtx] [-e[eof-str]] [-i[replace-str]] [-l[max-lines]] [-n max-args]
[-s max-chars] [-P max-procs] [--null] [--eof[=eof-str]] [--replace[=replace-str]]
[--max-lines[=max-lines]] [--interactive] [--max-chars=max-chars] [--verbose]
[--exit] [--max-procs=max-procs] [--max-args=max-args] [--no-run-if-empty]
[--version] [--help] [command [initial-arguments]]
```


DESCRIPTION

This manual page documents the GNU version of xargs. xargs reads arguments from the standard input, delimited by blanks (which can be protected with double or single quotes or a backslash) or newlines, and executes the command (default is /bin/echo) one or more times with any initial-arguments followed by arguments read from standard input. Blank lines on the standard input are ignored.

xargs exits with the following status:

0 if it succeeds

123 if any invocation of the command exited with status 1-125

124 if the command exited with status 255

125 if the command is killed by a signal

126 if the command cannot be run

127 if the command is not found

1 if some other error occurred.

OPTIONS

--null, -0	Input filenames are terminated by a null character instead of by whitespace, and the quotes and backslash are not special (every character is taken literally). Disables the end-of-file string, which is treated like any other argument. Useful when arguments might contain whitespace, quote marks, or backslashes. The GNU find -print0 option produces input suitable for this mode.
--eof[=eof-str], -e[eof-str]	Set the end-of-file string to eof-str. If the end-of-file string occurs as a line of input, the rest of the input is ignored. If eof-str is omitted, there is no end of file string. If this option is not given, the end-of-file string defaults to an underscore.
--help	Print a summary of the options to xargs and exit.
--replace[=replace-str], -i[replace-str]	Replace occurrences of replace-str in the initial arguments with names read from standard input. Also, unquoted blanks do not terminate arguments. If replace-str is omitted, it defaults to {} (like for find -exec). Implies -x and -l 1.

`DisplayManager.accessFile`
To prevent unauthorized XDMCP service and to allow forwarding of XDMCP `IndirectQuery` requests, this file contains a database of hostnames that are either allowed direct access to this machine, or have a list of hosts to which queries should be forwarded to. The format of this file is described in the subsection "XDMCP Access Control."

`DisplayManager.exportList`
A list of additional environment variables, separated by whitespace, to pass on to the `Xsetup`, `Xstartup`, `Xsession`,and `Xreset` programs.

`DisplayManager.randomFile`
A file to checksum to generate the seed of authorization keys. This should be a file that changes frequently. The default is `/dev/mem`.

`DisplayManager.greeterLib`
On systems that support a dynamically loadable greeter library, the name of the library. Default is `<XRoot>/lib/X11/xdm/libXdmGreet.so`.

`DisplayManager.choiceTimeout`
Number of seconds to wait for display to respond after user has selected a host from the `chooser`. If the display sends an XDMCP `IndirectQuery` within this time, the request is forwarded to the chosen host. Otherwise, it is assumed to be from a new session and the `chooser` is offered again. Default is 15.

`DisplayManager.DISPLAY.resources`
This resource specifies the name of the file to be loaded by `xrdb` as the resource database onto the root window of screen 0 of the display. The `Xsetup` program, the Login widget, and `chooser` will use the resources set in this file. This resource database is loaded just before the authentication procedure is started, so it can control the appearance of the login window. See the subsection "Authentication Widget," which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but `<XRoot>/lib/X11/xdm/Xresources` is the conventional name.

`DisplayManager.DISPLAY.chooser`
Specifies the program run to offer a host menu for `Indirect` queries redirected to the special hostname `CHOOSER`. `<XRoot>/lib/X11/xdm/chooser` is the default. See the subsections "XDMCP Access Control" and "`chooser`."

`DisplayManager.DISPLAY.xrdb`
Specifies the program used to load the resources. By default, `xdm` uses `<XRoot>/bin/xrdb`.

`DisplayManager.DISPLAY.cpp`
This specifies the name of the C preprocessor that is used by `xrdb`.

`DisplayManager.DISPLAY.setup`
This specifies a program that is run (as root) before offering the Login window. This may be used to change the appearance of the screen around the Login window or to put up other windows (for example, you may want to run `xconsole` here). By default, no program is run. The conventional name for a file used here is `Xsetup`. See the subsection "Setup Program."

`DisplayManager.DISPLAY.startup`
This specifies a program that is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is `Xstartup`. See the subsection "Startup Program."

`DisplayManager.DISPLAY.session`
This specifies the session to be executed (not running as root). By default, `<XRoot>/bin/xterm` is run. The conventional name is `Xsession`. See the subsection "Session Program."

```
#!/bin/sh
#
# Xsession
#
# This is the program that is run as the client
# for the display manager.
case $# in
  1)
  case $1 in
    failsafe)
    exec xterm -geometry 80x24-0-0
    ;;
  esac
esac
startup=$HOME/.xsession
resources=$HOME/.Xresources
if [ -f "$startup" ]; then
  exec "$startup"
else
if [ -f "$resources" ]; then
  xrdb -load "$resources"
fi
twm &
xman -geometry +10-10 &
exec xterm -geometry 80x24+10+10 -ls
fi
```


The user's `.xsession` file might look something like the following example. Don't forget that the file must have execute permission.

```
#!/bin/csh
# no -f in the previous line so .cshrc gets run to set $PATH
twm &
xrdb -merge "$HOME/.Xresources"
emacs -geometry +0+50 &
xbiff -geometry -430+5 &
xterm -geometry -0+50 -ls
```

RESET PROGRAM

Symmetrical with `Xstartup`, the `Xreset` script is run after the user session has terminated. Run as root, it should contain commands that undo the effects of commands in `Xstartup`, removing entries from `/etc/utmp` or unmounting directories from file servers. The environment variables that were passed to `Xstartup` are also passed to `Xreset`.

A sample `Xreset` script:

```
#!/bin/sh
#
# Xreset
#
# This program is run as root after the session ends
#
sessreg -d -l $DISPLAY -x /usr/X11R6/lib/xdm/Xservers $USER
/usr/X11R6/lib/xdm/TakeConsole
exit 0
```

CONTROLLING THE SERVER

`xdm` controls local servers using POSIX signals. `SIGHUP` is expected to reset the server, closing all client connections and performing other cleanup duties. `SIGTERM` is expected to terminate the server. If these signals do not perform the expected actions, the resources `DisplayManager.DISPLAY.resetSignal` and `DisplayManager.DISPLAY.termSignal` can specify alternate signals.

`slow_vram`—(`S3`) Adjusts the VRAM timings for cards using slow VRAM. This is required for some Diamond Stealth `64` VRAM and Hercules Terminator `64` cards.

`Fast_vram`—(`S3`) Adjusts the VRAM timings for faster VRAM access. There will be display errors and pixel garbage if your card can't support it.

`Slow_dram_refresh`—(S3) Adjusts the DRAM refresh for cards with slow DRAM to avoid lines of corrupted pixels when switching modes.

`No_block_write`—(`Mach64`) Disables the block write mode on certain types of VRAM Mach64 cards. If noise or shadows appear on the screen, this option should remove them.

`Block_write`—(Mach64) Enables the block write mode on certain types of VRAM Mach64 cards. Normally the Mach64 server will automatically determine if the card can handle block write mode, but this option will override the probe result.

`No_bios_clocks`—(Mach64) The Mach64 server normally reads the clocks from the bios. This option overrides the bios clocks and forces the server to use the clocks given in the `XF86Config` file.

There are also numerous tuning options for the AGX server. Refer to `README.agx` for details.

Note that `XFree86` has some internal capabilities to determine what hardware it is running on. Thus, normally the keywords `chipset`, `clocks`, and `videoram` don't have to be specified. But there may be occasions when this autodetection mechanism fails, (for example, too high a load on the machine when you start the server). For cases like this, one should first run the server on an unloaded machine, look at the results of the autodetection (that are printed out during server startup), and then explicitly specify these parameters in the configuration file. It is recommended that all parameters, especially `Clock` values, be specified in the `XF86Config` file.

FILES

`<XRoot>/bin/XF86 S3`	The 8-, 16-, and 24-bit color X server for S3
`<XRoot>/bin/XF86 Mach8`	The 8-bit color X server for Mach8
`<XRoot>/bin/XF86 Mach32`	The 8- and 16-bit color X server for Mach32
`<XRoot>/bin/XF86 Mach64`	The 8-, 16-, and 24-bit color X server for Mach64
`<XRoot>/bin/XF86 P9000`	The 8-, 16-, and 24-bit color X server for the P9000
`<XRoot>/bin/XF86 AGX`	The 8- and 16-bit color X server for AGX and XGA
`<XRoot>/bin/XF86 W32`	The 8-bit color X server for ET4000/W32
`<XRoot>/bin/XF86 8514`	The 8-bit color X server for IBM 8514 and true compatibles
`/etc/XF86Config`	Server configuration file
`<XRoot>/lib/X11/XF86Config`	Server configuration file (secondary location)
`<XRoot>/lib/X11/doc/README.agx`	Extra documentation for the AGX server
`<XRoot>/lib/X11/doc/README.P9000`	Extra documentation for the P9000 server
`<XRoot>/lib/X11/doc/README.S3`	Extra documentation for the S3 server
`<XRoot>/lib/X11/doc/README.W32`	Extra documentation for the W32 server

Note: `<XRoot>` refers to the root of the X11 install tree.

Avance Logic	AL2101, ALI2301, ALI2302, ALI2308, ALI2401
Chips & Technology	65520, 65530, 65540, 65545
MX	MX68000, MX68010
Video7	HT216-32

Accelerated support is included for most of the Cirrus chipsets, and for the Western Digital WD90C31 and WD90C33 chipsets. Accelerated support for the ET4000/W32 is implemented in a separate server (see XF86_W32(1)). Users of boards based on ATI's Mach8, Mach32, and Mach64 chipsets should refer to the XF86_Mach8(1), XF86_Mach32(1) and XF86_Mach64(1) manual pages, respectively.

OPTIONS

In addition to the normal server options described in the Xserver(1) manual page, XF86_SVGA accepts some more command line switches, as described in the XFree86(1) man page.

SETUP

XFree86 uses a configuration file called XF86Config for its initial setup.

See the XF86Config(4/5) man page for general details. Here only the XF86_SVGA specific parts are explained.

This server requires a Screen section in the XF86Config file with the Driver entry set to vga.

Entries for the Device section in the XF86Config file include

chipset "name"

Specifies a chipset so the correct driver can be used. Possible chipsets are

ATI	vgawonder
Tseng	et3000, et4000, et4000w32, et4000w32i, et4000w32p
Western Digital	pvga1, wd90c00, wd90c10, wd90c24, wd90c30, wd90c31, wd90c33
Genoa	gvga
Trident	tvga8800cs, tvga8900b, tvga8900c, tvga8900cl, tvga9000
NCR	ncr77c22, ncr77c22e
Cirrus Logic	clgd5420, clgd5422, clgd5424, clgd5426, clgd5428, clgd5429, clgd5430, clgd5434, clgd5436, clgd6205, clgd6215, clgd6225, clgd6235, cl6410, cl6412, cl6420, cl6440
RealTek	realtek
ARK	ark1000pv, ark1000vl, ark2000pv
Compaq	cpq_avga
Oak	oti067, oti077, oti087
Avance Logic	al2101, ali2301, ali2302, ali2308, ali2401
Chips & Technology	ct65520, ct65530, ct65540, ct65545
MX	mx
Video7	video7
Generic	generic

`<XRoot>/lib/X11/doc/README.cirrus`	Extra documentation for the Cirrus driver
`<XRoot>/lib/X11/doc/README.trident`	Extra documentation for the Trident driver
`<XRoot>/lib/X11/doc/README.tseng`	Extra documentation for the `ET4000` and `ET3000` drivers
`<XRoot>/lib/X11/doc/README.Oak`	Extra documentation for the Oak driver
`<XRoot>/lib/X11/doc/README.Video7`	Extra documentation for the Video7 driver
`<XRoot>/lib/X11/doc/README.WstDig`	Extra documentation for the WD/PVGA driver

Note: `<XRoot>` refers to the root of the X11 install tree.

SEE ALSO

`X`(1), `Xserver`(1), `XFree86`(1), `XF86Config`(4/5), `xf86config`(1), `xvidtune`(1), `xdm`(1), `xinit`(1)

BUGS

Bug reports are welcome, and should be e-mailed to the address listed below.

CONTACT INFO

`XFree86` source is available from the FTP server `ftp.XFree86.org`.

Send e-mail to `XFree86@XFree86.org` for detai

AUTHORS

Refer to the XFree86(1) manual page.

XF86_VGA16

`XF86 VGA16`—4-bit nonaccelerated X Window System server for UNIX on x86 platforms

SYNOPSIS

```
XF86 VGA16 [:displaynumber] [ option ] ...
```

DESCRIPTION

`XF86_VGA16` is a 4-bit color server for `VGA` cards. The default root visual for this server is `StaticColor`. It also includes support for the non-VGA monochrome cards described in the `XF86_Mono`(1) manual page. It may be run in a dual-headed configuration.

CONFIGURATIONS

The `XF86_VGA16` server supports the following popular `SVGA` chipsets in 16-color mode.

ATI	`18800`, `18800-1`, `28800-2`, `28800-4`, `28800-5`, `28800-6`, `68800-3`, `68800-6`, `68800AX`, `68800LX`, `88800CX`, `88800GX`
Tseng	`ET4000`
Trident	`TVGA8800CS`, `TVGA8900B`, `TVGA8900C`, `TVGA8900CL`, `TVGA9000`
Cirrus	`CL6410`, `CL6412`, `CL6420`, `CL6440`
Oak	`OTI067`, `OTI077`, `OTI087`

Additionally, it supports generic VGA cards.

`XF86_VGA16` does not support the accelerated functions of the supported chipsets.

```
Xfd xfd
      Paned pane
           Label fontname
                Box box
                     Command quit
                     Command prev
                     Command next
           Label select
           Label metrics
           Label range
           Label start
Form form
FontGrid grid
```

FONTGRID RESOURCES

The FontGrid widget is an application-specific widget, and a subclass of the Simple widget in the Athena widget set. The effects and instance names of this widget's resources are given in the "Options" subsection. Capitalize the first letter of the resource instance name to get the corresponding class name.

APPLICATION SPECIFIC RESOURCES

The instance names of the application-specific resources are given in the following list. Capitalize the first letter of the resource instance name to get the corresponding class name. These resources are unlikely to be interesting unless you are localizing `xfd` for a different language.

`selectFormat`
: Specifies a `printf`-style format string used to display information about the selected character. The default is `character 0x%02x%02x (%u,%u) (%#o,%#o)`. The arguments that will come after the format string are

  `char.byte1`, `char.byte2`, `char.byte1`, `char.byte2`, `char.byte1`, `char.byte2`. `char.byte1` is byte 1 of the selected character. `char.byte2` is byte 2 of the selected character.

`metricsFormat`
: Specifies a `printf`-style format string used to display character metrics. The default is `width %d; left %d, right %d; ascent %d, descent %d (font %d, %d)`. The arguments that will come after the format string are the character metrics width, lbearing, rbearing, character ascent, character descent, font ascent, and font descent.

`rangeFormat`
: Specifies a `printf`-style format string used to display the range of characters currently being displayed. The default is `range: 0x%02x%02x (%u,%u) thru 0x%02x%02x (%u,%u)`. The arguments that will come after the format string are the following fields from the `XFontStruct` that is returned from opening the font:

  `min_byte1`, `min_char_or_byte2`, `min_byte1`, `min_char_or_byte2`, `max_byte1`, `max_char_or_byte2`, `max_byte1`, `max_char_or_byte2`.

`startFormat`
: Specifies a `printf`-style format string used to display information about the character at the upper left corner of the font grid. The default is `upper left: 0x%04x (%d,%d)`. The arguments that will come after the format string are the new character, the high byte of the new character, and the low byte of the new character.

`nocharFormat`
: Specifies a `printf`-style format string to display when the selected character does not exist. The default is `no such character 0x%02x%02x (%u,%u) (%#o,%#o`. The arguments that will come after the format string are the same as for the `select-Format` resource.

`-showtechs` Display a comprehensive list of techniques, by category, indicating which of the techniques are supported by the `XIE` server.

`-showlabels` Print test label to screen prior to calling any of the test code. This allows the user to know which test is executing in case the test hangs for some reason.

`-showevents` Be verbose when running event and error tests. Also, can be used to catch and display information on any signals received during execution of `xieperf`. Note that this flag is best used in a debugging situation, or to validate that the error events received by `xieperf` are valid the first time the tests are executed on a new platform.

`-events` Run tests that test for event generation.

`-errors` Run tests that test for error event generation.

`-loCal` Skip test calibration. This may be used when running `xieperf` in situations where execution timing is not important. Execution times will not be reported by `xieperf` when this option is enabled. The inner loop repeat count, additionally, is set to a value of 5 (but can be overridden by the `-reps` option).

`-all` Run all tests. This may take a while, depending on the speed of your machine, and its floating-point capabilities. This option is ignored if a script file is used.

`-tests` Generate a list of the available tests for the `xieperf` program. In `x11perf`, this list is normally displayed in the usage statement. It was yanked from the usage of `xieperf` because it was too lengthy.

`-mkscript` Generate a script file suitable for use with the script option. If `-repeat` or `-reps` are also specified, they will be automatically placed at the end of each command in the script. The script is generated to `stderr`. See the `-script` command, above.

`-cache <n>` Most test flos utilize a photomap resource for a source. A photomap cache of up to `n` entries is controlled by `xieperf` to avoid having to constantly reload these images during test initialization. The default cache size is `4`. If a value less than the default is specified, the cache size will be set to the default.

`-labels` Generates just the descriptive labels for each test specified. Use `-all` or `-range` to specify which tests are included. See `x11perfcomp`(1) for more details.

`-DIS` Pretend we are running `xieperf` while connected to a `DIS`-only capable implementation of `XIE`. This will cause `xieperf` to execute those tests that only use protocol requests found in the `DIS` subset of `XIE`, and bypass those which are not `DIS`-compatible. If `xieperf` detects a `DIS` server, it will do this automatically, and this option is ignored. Use `-all` or `-range` to specify the initial range of tests.

`-range <test1>[,<test2>]` Runs all the tests starting from the specified name *test1* until the name *test2*, including both the specified tests. Some tests, like the event and error tests, also require the `-errors` or `-events` options to specified. This option is ignored if a script is used.

configuration. The tests can be configured so that the test `init` function will constrain the input `photomap` to a specified number of levels, on a per band basis, so that word-sized and quad-sized pixels are passed through the flos. Some tests below take advantage of this. See `tests.c` for test configuration, and hints on how to add similar tests.

`-point1` through `-point3` — Simple `Point` element tests. Drawable destination.

`-pointroi1` — Simple `Point` element test that uses ROIs. Drawable destination.

`-pointcplane1` — Simple `Point` element test that uses a control plane. Drawable destination.

`-pointphoto1` — Simple `Point` element test. Photomap destination.

`-pointroiphoto1` — Simple `Point` element test that uses ROIs. Photomap destination.

`-pointcplanephoto1` — Simple `Point` element test that uses a control plane. Photomap destination.

`-redefine` — Two flographs are created that are the same in structure, except for the x and y offsets specified for the `ExportDrawable` flo elements. The test `init` function creates a photoflo based upon one of the two flographs. The inner loop of the test function uses `XieRedefinePhotoflo()` to alternate between each of the flographs. Make sure that your inner loop reps are 2 or greater in order to exercise this test fully (see `-reps`).

`-modify1` — Test `XieModifyPhotoflo()` by adjusting ROI offsets and size.

`-modify2` — Test `XieModifyPhotoflo()` by changing the LUT input to a `Point` element.

`-modify3` — Test `XieModifyPhotoflo()` by changing `ExportDrawable` x and y offsets.

`-modify4` — This test creates a rather long flo of arithmetic elements, each of which does nothing more than add `1` to a small image. The test `init` function scales the input photomap. The `ExportDrawable` x and y offset is modified randomly during each iteration of the test function inner loop.

`-modify5` — This test creates a rather long flo of `arithmetic` elements, each of which does nothing more than add `1` to a large image. Each `rep`, the `Geometry` and `ExportDrawable` elements at the end of the flo are modified to crop a small piece of the input into its appropriate place in the larger image.

`-rgb1` through `-rgb16` — These tests all basically take an `UncompressedTriple` image as input, send it to `ConvertFromRGB`, which converts the image to some configured colorspace, and then send the converted image on to `ConvertToRGB` prior to display. The original image is displayed in the left-hand window, and the image that has passed through the flo is shown in the right-hand window. The goal of these test is to show that `ConvertFromRGB -> ConvertToRGB` is lossless.

`-converttoindexpixel` — `ConvertToIndex` test, `TripleBand BandByPixel`.

`-converttoindexplane` — `ConvertToIndex` test, `TripleBand BandByPlane`.

xinetd

xinetd—The extended Internet services daemon

SYNOPSIS

```
xinetd [options]
```

DESCRIPTION

xinetd performs the same function as inetd: it starts programs that provide Internet services. Instead of having such servers started at system initialization time, and be dormant until a connection request arrives, xinetd is the only daemon process started and it listens on all service ports for the services listed in its configuration file. When a request comes in, xinetd starts the appropriate server. Because of the way it operates, xinetd (as well as inetd) is also referred to as a super-server.

The services listed in xinetd's configuration file can be separated into two groups. Services in the first group are called multithreaded and they require the forking of a new server process for each new connection request. The new server then handles that connection. For such services, xinetd keeps listening for new requests so that it can spawn new servers. On the other hand, the second group includes services for which the server daemon is responsible for handling all new connection requests. Such services are called single-threaded and xinetd will stop handling new requests for them until the server dies. Services in this group are usually datagram based.

So far, the only reason for the existence of a super-server was to conserve system resources by avoiding to fork a lot of processes who might be dormant for most of their lifetime. While fulfilling this function, xinetd takes advantage of the idea of a super-server to provide features such as access control and logging. Furthermore, xinetd is not limited to services listed in /etc/services. Therefore, anybody can use xinetd to start special-purpose servers.

OPTIONS

-d

Enables debug mode. This produces a lot of debugging output, and it makes it possible to use a debugger on xinetd.

-syslog syslog_facility

This option enables syslog logging of xinetd-produced messages using the specified syslog facility. The following facility names are supported: daemon, auth, user, local[0-7] (check syslog.conf(5) for their meanings). This option is ineffective in debug mode because all relevant messages are sent to the terminal.

-filelog logfile

xinetd-produced messages will be placed in the specified file. Messages are always appended to the file. If the file does not exist, it will be created. This option is ineffective in debug mode because all relevant messages are sent to the terminal.

-f config_file

Determines the file that xinetd uses for configuration. The default is /etc/xinetd.conf.

-pid

The process pid is written to standard error. This option is ineffective in debug mode.

-loop rate

This option sets the loop rate beyond which a service is considered in error and is deactivated. The loop rate is specified in terms of the number of servers per second that can be forked for a process. The speed of your machine determines the correct value for this option. The default rate is 10.

-reuse

If this option is used, xinetd will set the socket option SO_REUSEADDR before binding the service socket to an Internet address. This allows binding of the address even if there are programs that use it, which happens when a previous instance of xinetd has started some servers that are still running. This option has no effect on RPC services.

OPTIONS

`-display dpy`
This option specifies the X server to which to connect.

`-format string`
This option specifies a `printf`-style string used to list each atom `<value,name>` pair, printed in that order (`value` is an unsigned long and `name` is a `char *`). `xlsatoms` will supply a newline at the end of each line. The default is `%ld\t%s`.

`-range [low]-[high]`
This option specifies the range of atom values to check. If `low` is not given, a value of `1` is assumed. If `high` is not given, `xlsatoms` will stop at the first undefined atom at or above `low`.

`-name string`
This option specifies the name of an atom to list. If the atom does not exist, a message will be printed on the standard error

SEE ALSO

`X`(1), `Xserver`(1), `xprop`(1)

ENVIRONMENT

`DISPLAY`
To get the default host and display to use

AUTHOR

Jim Fulton (MIT X Consortium)

X Version 11 Release 6

xlsclients

`xlsclients`—List client applications running on a display

SYNOPSIS

```
xlsclients [-display displayname] [-a] [-l] [-m maxcmdlen]
```

DESCRIPTION

`xlsclients` is a utility for listing information about the client applications running on a display. It may be used to generate scripts representing a snapshot of the user's current session.

OPTIONS

`-display displayname`
This option specifies the X server to contact.

`-a`
This option indicates that clients on all screens should be listed. By default, only those clients on the default screen are listed.

`-l`
List in long format, giving the window name, icon name, and class hints in addition to the machine name and command string shown in the default format.

`-m maxcmdlen`
This option specifies the maximum number of characters in a command to print out. The default is `10000`.

ENVIRONMENT

`DISPLAY`
To get the default host, display number, and screen.

OPTIONS

The following options may be used with `xmodmap`:

`-display display`
: This option specifies the host and display to use.

`-help`
: This option indicates that a brief description of the command-line arguments should be printed on the standard error channel. This will be done whenever an unhandled argument is given to `xmodmap`.

`-grammar`
: This option indicates that a help message describing the expression grammar used in files and with `-e` expressions should be printed on the standard error.

`-verbose`
: This option indicates that `xmodmap` should print logging information as it parses its input.

`-quiet`
: This option turns off the verbose logging. This is the default.

`-n`
: This option indicates that `xmodmap` should not change the mappings, but should display what it would do, like `make`(1) does when given this option.

`-e expression`
: This option specifies an expression to be executed. Any number of expressions may be specified from the command line.

`-pm`
: This option indicates that the current modifier map should be printed on the standard output.

`-pk`
: This option indicates that the current keymap table should be printed on the standard output.

`-pke`
: This option indicates that the current keymap table should be printed on the standard output in the form of expressions that can be fed back to `xmodmap`.

`-pp`
: This option indicates that the current pointer map should be printed on the standard output.

`-`
: A lone dash means that the standard input should be used as the input file.

The filename specifies a file containing `xmodmap` expressions to be executed. This file is usually kept in the user's home directory with a name like `.xmodmaprc`.

EXPRESSION GRAMMAR

The `xmodmap` program reads a list of expressions and parses them all before attempting to execute any of them. This makes it possible to refer to keysyms that are being redefined in a natural way without having to worry as much about name conflicts.

`keycode NUMBER = KEYSYMNAME ...`
: The list of keysyms is assigned to the indicated keycode (which may be specified in decimal, hex, or octal and can be determined by running the `xev` program.

`keycode any = KEYSYMNAME ...`
: If no existing key has the specified list of keysyms assigned to it, a spare key on the keyboard is selected and the keysyms are assigned to it. The list of keysyms may be specified in decimal, hex, or octal.

`keysym KEYSYMNAME = KEYSYMNAME ...`
: The `KEYSYMNAME` on the left side is translated into matching keycodes used to perform the corresponding set of `keycode` expressions. The list of keysym names may be found in the

`-font font`	This argument allows the user to specify that the properties of font `font` should be displayed.
`-root`	This argument specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.
`-display display`	This argument allows you to specify the server to connect to; see `X`(1).
`-len n`	Specifies that at most, *n* bytes of any property should be read or displayed.
`-notype`	Specifies that the type of each property should not be displayed.
`-fs file`	Specifies that file `file` should be used as a source of more formats for properties.
`-frame`	Specifies that when selecting a window by hand (that is, if neither `-name`, `-root` nor `-id` is given), look at the window manager frame (if any) instead of looking for the client window.
`-remove property-name`	Specifies the name of a property to be removed from the indicated window.
`-spy`	Examine window properties forever, looking for property change events.
`-f name format [dformat]`	Specifies that the format for `name` should be `format` and that the dformat for `name` should be `dformat`. If `dformat` is missing, `"= $0+\n"` is assumed.

DESCRIPTION

For each of these properties, its value on the selected window or font is printed using the supplied formatting information, if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, `not defined` is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the `-root` argument may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by id number, such as might be obtained from `xwininfo`, or by name if the window possesses a name. The `-id` argument selects a window by id number in either decimal or hex (must start with `0x`) while the `-name` argument selects a window by name.

The last way to select a window does not involve the command line at all. If none of `-font`, `-id`, `-name`, and `-root` are specified, a crosshairs cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the `-font` argument must be used.

Other than the preceding four arguments and the `-help` argument for obtaining help, and the `-grammar` argument for listing the full grammar for the command line, all the other command-line arguments are used in specifying both the format of the properties to be displayed and how to display them. The `-len n` argument specifies that at most, *n* bytes of any given property will be read and displayed. This is useful, for example, when displaying the cut buffer on the root window, which could run to several pages if displayed in full.

Normally, each property name is displayed by printing first the property name then its type (if it has one) in parentheses, followed by its value. The `-notype` argument specifies that property types should not be displayed. The `-fs` argument is used to specify a file containing a list of formats for properties, while the `-f` argument is used to specify the format for one property.

NUM_SCREENS=num	The total number of screens.
SCREEN_NUM=num	The number of the current screen (from zero).
BITS_PER_RGB=num	The number of significant bits in an RGB color specification. This is the log base 2 of the number of distinct shades of each primary that the hardware can generate. Note that it usually is not related to PLANES.
CLASS=visualclass	One of StaticGray, GrayScale, StaticColor, PseudoColor, TrueColor, DirectColor. This is the visual class of the root window.
CLASS_visualclass=visualid	The visual class of the root window in a form you can #ifdef on. The value is the numeric id of the visual.
COLOR	Defined only if CLASS is one of StaticColor, PseudoColor, TrueColor, or DirectColor.
CLASS_visualclass_depth=num	A symbol is defined for each visual supported for the screen. The symbol includes the class of the visual and its depth; the value is the numeric id of the visual. (If more than one visual has the same class and depth, the numeric id of the first one reported by the server is used.)
HEIGHT=num	The height of the root window in pixels.
WIDTH=num	The width of the root window in pixels.
PLANES=num	The number of bit planes (the depth) of the root window.
X_RESOLUTION=num	The x resolution of the screen in pixels per meter.
Y_RESOLUTION=num	The y resolution of the screen in pixels per meter.

SRVR_name, CLNT_name, VNDR_name, and EXT_name identifiers are formed by changing all characters other than letters and digits into underscores.

Lines that begin with an exclamation mark (!) are ignored and may be used as comments.

Note that since xrdb can read from standard input, it can be used to change the contents of properties directly from a terminal or from a shell script.

OPTIONS

xrdb program accepts the following options:

-help	This option (or any unsupported option) will cause a brief description of the allowable options and parameters to be printed.
-display display	This option specifies the X server to be used; see X(1). It also specifies the screen to use for the -screen option, and it specifies the screen from which preprocessor symbols are derived for the -global option.
-all	This option indicates that operation should be performed on the screen-independent resource property (RESOURCE_MANAGER), as well as the screen-specific property (SCREEN_RESOURCES) on every screen of the display. For example, when used in conjunction with -query, the contents of all properties are output. For -load, -override, and -merge, the input file is processed once for each screen. The resources that occur in common in the output for every screen are collected, and these are applied as the screen-independent resources. The remaining resources are applied for each individual per-screen property. This the default mode of operation.

`-backup string`	This option specifies a suffix to be appended to the filename used with `-edit` to generate a backup file.
`-Dname[=value]`	This option is passed through to the preprocessor and is used to define symbols for use with conditionals such as `#ifdef`.
`-Uname`	This option is passed through to the preprocessor and is used to remove any definitions of this symbol.
`-Idirectory`	This option is passed through to the preprocessor and is used to specify a directory to search for files that are referenced with `#include`.

FILES

Generalizes `~/.Xdefaults` files

SEE ALSO

`X`(1), Xlib Resource Manager documentation, `Xt` resource documentation

ENVIRONMENT

`DISPLAY` To figure out which display to use.

BUGS

The default for argument should be to query, not to overwrite, so that it is consistent with other programs.

AUTHORS

Bob Scheifler and Phil Karlton, rewritten from the original by Jim Gettys

X Version 11 Release 6

xrefresh

`xrefresh`—Refresh all or part of an X screen

SYNOPSIS

```
xrefresh [-option ...]
```

DESCRIPTION

`xrefresh` is a simple X program that causes all or part of your screen to be repainted. This is useful when system messages have messed up your screen. `xrefresh` maps a window on top of the desired area of the screen and then immediately unmaps it, causing refresh events to be sent to all applications. By default, a window with no background is used, causing all applications to repaint smoothly. However, the various options can be used to indicate that a solid background (of any color) or the root window background should be used instead.

ARGUMENTS

`-white`	Use a white background. The screen just appears to flash quickly, and then repaint.
`-black`	Use a black background (in effect, turning off all of the electron guns to the tube). This can be somewhat disorienting as everything goes black for a moment.
`-solid color`	Use a solid background of the specified color. Try green.
`-root`	Use the root window background.

ENVIRONMENT

DISPLAY	To get default host and display number

SEE ALSO

X(1)

AUTHOR

Donna Converse (MIT X Consortium)

X Version 11 Release 6

xterm

xterm—Terminal emulator for X

SYNOPSIS

```
xterm [-toolkitoption ...] [-option ...]
```

DESCRIPTION

The xterm program is a terminal emulator for the X Window System. It provides DEC VT102- and Tektronix 4014-compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), xterm will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals all have their own windows so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper-left area of the window.

Although both windows may be displayed at the same time, one of them is considered the active window for receiving keyboard input and terminal output. This is the window that contains the text cursor. The active window can be chosen through escape sequences, the VT Options menu in the VT102 window, and the Tek Options menu in the 4014 window.

EMULATIONS

The VT102 emulation is fairly complete, but does not support smooth scrolling, VT52 mode, the blinking character attribute nor the double-wide and double-size character sets. termcap(5) entries that work with xterm include xterm, vt102, vt100, and ansi, and xterm automatically searches the termcap file in this order for these entries and then sets the TERM and the TERMCAP environment variables.

Many of the special xterm features may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the "Xterm Control Sequences" document.)

The Tektronix 4014 emulation is also fairly good. It supports 12-bit graphics addressing, scaled to the window size. Four different font sizes and five different lines types are supported. There is no write-through or defocused mode support. The Tektronix text and graphics commands are recorded internally by xterm and may be written to a file by sending the COPY escape sequence (or through the Tektronix menu, discussed later in this section). The name of the file will be COPYyy–MM–dd.hh:mm:ss, where *yy*, *MM*, *dd*, *hh*, *mm*, and *ss* are the year, month, day, hour, minute, and second when the COPY was performed (the file is created in the directory xterm is started in, or the home directory for a login xterm).

OTHER FEATURES

xterm automatically highlights the text cursor when the pointer enters the window (selected) and unhighlights it when the pointer leaves the window (unselected). If the window is the focus window, then the text cursor is highlighted no matter where the pointer is. In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which

`-name name`
This option specifies the application name under which resources are to be obtained, rather than the default executable filename. Name should not contain `.` or `*` characters.

`-title string`
This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the `-e option`, if any; otherwise, the application name.

`-rv`
This option indicates that reverse video should be simulated by swapping the foreground and background colors.

`-geometry geometry`
This option specifies the preferred size and position of the VT102 window; see `X`(1).

`-display display`
This option specifies the X server to contact; see `X`(1).

`-xrm resourcestring`
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command-line options.

`-iconic`
This option indicates that `xterm` should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes as well as the following:

`iconGeometry (class IconGeometry)`
Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

`iconName (class IconName)`
Specifies the icon name. The default is the application name.

`termName (class TermName)`
Specifies the terminal type name to be set in the `TERM` environment variable.

`title (class Title)`
Specifies a string that may be used by the window manager when displaying this application.

`ttyModes (class TtyModes)`
Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include `intr`, `quit`, `erase`, `kill`, `eof`, `eol`, `swtch`, `start`, `stop`, `brk`, `susp`, `dsusp`, `rprnt`, `flush`, `weras`, and `lnext`. Control characters may be specified as `^char` (for example, `^c` or `^u`) and `^?` may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an `stty` every time an `xterm` is started.

`useInsertMode (class UseInsertMode)`
Force use of insert mode by adding appropriate entries to the `TERMCAP` environment variable. This is useful if the system `termcap` is broken. The default is `False`.

`utmpInhibit (class UtmpInhibit)`
Specifies whether or not `xterm` should try to record the user's terminal in `/etc/utmp`.

`sunFunctionKeys (class SunFunctionKeys)`
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

`waitForMap (class WaitForMap)`
Specifies whether or not `xterm` should wait for the initial window map before starting the subprocess. The default is `false`.

```
48, 48, 48, 48, 48, 48, 48, 48,
/*8 9 :;< => ?*/
48, 48, 58, 59, 60, 61, 62, 63,
/*@ABC D E F G*/
64, 48, 48, 48, 48, 48, 48, 48,
/* H I J K L M N O */
48, 48, 48, 48, 48, 48, 48, 48,
/*P QR S T UVW*/
48, 48, 48, 48, 48, 48, 48, 48,
/*XY Z [\]^ */
48, 48, 48, 91, 92, 93, 94, 48,
/*'ab c d e fg */
96, 48, 48, 48, 48, 48, 48, 48,
/*h ijklm n o */
48, 48, 48, 48, 48, 48, 48, 48,
/*p q rs tu v w */
48, 48, 48, 48, 48, 48, 48, 48,
/*x y zf jg~ DEL */
48, 48, 48, 123, 124, 125, 126, 1};
```

For example, the string `33:48,37:48,45-47:48,64:48` indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters and numbers. This is useful for cutting and pasting electronic mailing addresses and filenames.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input by changing the translations for the `vt100` or `tek4014` widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the `vt100` or `tek4014` translations resources:

`bell([percent])`
: This action rings the keyboard bell at the specified percentage above or below the base volume.

`ignore()`
: This action ignores the event but checks for special pointer position escape sequences.

`insert()`
: This action inserts the character or string associated with the key that was pressed.

`insert-seven-bit()`
: This action is a synonym for `insert()`.

`insert-eight-bit()`
: This action inserts an eight-bit (Meta) version of the character or string associated with the key that was pressed. The exact action depends on the value of the `eightBitInput` resource.

`insert-selection (sourcename [, ...])`
: This action inserts the string found in the selection or cut buffer indicated by `sourcename`. Sources are checked in the order given (case is significant) until one is found. Commonly used selections include `PRIMARY`, `SECONDARY`, and `CLIPBOARD`. Cut buffers are typically named `CUT_BUFFER0` through `CUT_BUFFER7`.

`keymap(name)`
: This action dynamically defines a new translation table whose resource name is `name` with the suffix `Keymap` (case is significant). The name `None` restores the original translation table.

`popup-menu(menuname)`
: This action displays the specified pop-up menu. Valid names (case is significant) include `mainMenu`, `vtMenu`, `fontMenu`, and `tekMenu`.

`secure()`
: This action toggles the Secure Keyboard mode described in the "Security" subsection, and is invoked from the `securekbd` entry in `mainMenu`.

`set-scrollbar(on/off/toggle)`	This action toggles the `scrollbar` resource and is also invoked by the `scrollbar` entry in `vtMenu`.
`set-jumpscroll(on/off/toggle)`	This action toggles the `jumpscroll` resource and is also invoked by the `jumpscroll` entry in `vtMenu`.
`set-reverse-video(on/off/toggle)`	This action toggles the `reverseVideo` resource and is also invoked by the `reversevideo` entry in `vtMenu`.
`set-autowrap(on/off/toggle)`	This action toggles automatic wrapping of long lines and is also invoked by the `autowrap` entry in `vtMenu`.
`set-reversewrap(on/off/toggle)`	This action toggles the `reverseWrap` resource and is also invoked by the `reversewrap` entry in `vtMenu`.
`set-autolinefeed(on/off/toggle)`	This action toggles automatic insertion of line-feeds and is also invoked by the `autolinefeed` entry in `vtMenu`.
`set-appcursor(on/off/toggle)`	This action toggles the handling Application Cursor Key mode and is also invoked by the `appcursor` entry in `vtMenu`.
`set-appkeypad(on/off/toggle)`	This action toggles the handling of Application Key-pad mode and is also invoked by the `appkeypad` entry in `vtMenu`.
`set-scroll-on-key(on/off/toggle)`	This action toggles the `scrollKey` resource and is also invoked from the `scrollkey` entry in `vtMenu`.
`set-scroll-on-tty-output(on/off/toggle)`	This action toggles the `scrollTtyOutput` resource and is also invoked from the `scrollttyoutput` entry in `vtMenu`.
`set-allow132(on/off/toggle)`	This action toggles the `c132` resource and is also invoked from the `allow132` entry in `vtMenu`.
`set-cursesemul(on/off/toggle)`	This action toggles the `curses` resource and is also invoked from the `cursesemul` entry in `vtMenu`.
`set-visual-bell(on/off/toggle)`	This action toggles the `visualBell` resource and is also invoked by the `visualbell` entry in `vtMenu`.
`set-marginbell(on/off/toggle)`	This action toggles the `marginBell` resource and is also invoked from the `marginbell` entry in `vtMenu`.
`set-altscreen(on/off/toggle)`	This action toggles between the alternate and current screens.
`soft-reset()`	This action resets the scrolling region and is also invoked from the `softreset` entry in `vtMenu`.
`hard-reset()`	This action resets the scrolling region, tabs, window size, and cursor keys, and clears the screen. It is also invoked from the `hardreset` entry in `vtMenu`.
`clear-saved-lines()`	This action does `hard-reset()` and also clears the history of lines saved off the top of the screen. It is also invoked from the `clearsavedlines` entry in `vtMenu`.
`set-terminal-type(type)`	This action directs output to either the `vt` or `tek` windows, according to the `type` string. It is also invoked by the `tekmode` entry in `vtMenu` and the `vtmode` entry in `tekMenu`.
`set-visibility(vt/tek,on/off/toggle)`	This action controls whether or not the `vt` or `tek` windows are visible. It is also invoked from the `tekshow` and `vthide` entries in `vtMenu` and the `vtshow` and `tekhide` entries in `tekMenu`.
`set-tek-text(large/2/3/small)`	This action sets font used in the Tektronix window to the value of the resources `tektextlarge`, `tektext2`, `tektext3`, and `tektextsmall` according to the argument. It is also by the entries of the same names as the resources in `tekMenu`.
`tek-page()`	This action clears the Tektronix window and is also invoked by the `tekpage` entry in `tekMenu`.

ENVIRONMENT

`xterm` sets the environment variables `TERM` and `TERMCAP` properly for the size window you have created. It also uses and sets the environment variable `DISPLAY` to specify which bit map display terminal to use. The environment variable `WINDOWID` is set to the X window `id` number of the `xterm` window.

SEE ALSO

`resize`(1), `X`(1), `pty`(4), `tty`(4)

Xterm Control Sequences

BUGS

Large pastes do not work on some systems. This is not a bug in `xterm`; it is a bug in the pseudo-terminal driver of those systems. `xterm` feeds large pastes to the `pty` only as fast as the `pty` will accept data, but some `pty` drivers do not return enough information to know if the write has succeeded.

Many of the options are not resettable after `xterm` starts.

Only fixed-width, character-cell fonts are supported.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

There needs to be a dialog box to allow entry of the Tek `COPY` filename.

AUTHORS

Far too many people, including Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP), Jonathan Kamens (MIT-Athena)

X Version 11 Release 6

Xvfb

`Xvfb`—Virtual framebuffer X server for X Version 11

SYNOPSIS

```
Xvfb [ option ] ...
```

DESCRIPTION

`Xvfb` is an X server that can run on machines with no display hardware and no physical input devices. It emulates a dumb framebuffer using virtual memory.

The primary use of this server is intended to be server testing. The `mfb` or `cfb` code for any depth can be exercised with this server without the need for real hardware that supports the desired depths.

A secondary use is testing clients against unusual depths and screen configurations.

xvidtune

`xvidtune`—Video mode tuner for XFree86

SYNOPSIS

```
xvidtune [ -prev j -next j -unlock j -query j -saver suspendtime [ offtime ]][-toolkitoption ... ]
```

DESCRIPTION

`Xvidtune` is a client interface to the XFree86 X server video mode extension (XFree86-VidModeExtension).

When given one of the nontoolkit options, `xvidtune` provides a command-line interface to either switch the video mode or get/set monitor powersaver time-outs.

Without any options (or with only toolkit options) it presents the user with various buttons and sliders that can be used to interactively adjust existing video modes. It will also print the settings in a format suitable for inclusion in an `XF86Config` file.

NOTE

The original mode settings can be restored by pressing the `R` key, and this can be used to restore a stable screen in situations where the screen becomes unreadable.

The available buttons are:

`Left`
`Right`
`Up`
`Down`

Adjust the video mode so that the display will be moved in the appropriate direction:

`Wider`
`Narrower`
`Shorter`
`Taller`

Adjust the video mode so that the display size is altered appropriately:

`Quit`	Exit the program.
`Apply`	Adjust the current video mode to match the selected settings.
`Auto`	Cause the Up/Down/Right/Left, Wider/Narrower/Shorter/Taller, Restore, and the special S3 buttons to be applied immediately. This button can be toggled.
`Test`	Temporarily switch to the selected settings.
`Restore`	Return the settings to their original values.
`Fetch`	Query the server for its current settings.
`Show`	Print the currently selected settings to `stdout` in `XF86Config Modeline` format. The primary selection is similarly set.
`Next`	Switch the `Xserver` to the next video mode.
`Prev`	Switch the `Xserver` to the previous video mode.

For some S3-based cards (964 and 968) the following are also available:

`InvertVCLK`	Change the VCLK invert/noninvert state.
`EarlySC`	Change the Early SC state. This affects screen wrapping.

`BlankDelay1, BlankDelay2` — Set the blank delay values. This affects screen wrapping. Acceptable values are in the range 0–7. The values may be incremented or decremented with the + and - buttons, or by pressing the + or - keys in the text field.

For S3-864/868 based cards, `InvertVCLK` and `BlankDelay1` may be useful. For S3 Trio32/Trio64 cards, only `InvertVCLK` is available. At the moment there are no default settings available for these chips in the video mode extension and thus this feature is disabled in `xvidtune`. It can be enabled by setting any of the optional S3 commands in the screen section of `XF86Config`, for example, using

```
blank delay "" 0
```

OPTIONS

`xvidtune` accepts the standard X Toolkit command-line options as well as the following:

`-prev` — Switch the `Xserver` to the previous video mode.

`-next` — Switch the `Xserver` to the next video mode.

`-unlock` — Normally, `xvidtune` will disable the switching of video modes via hot keys while it is running. If for some reason the program did not exit cleanly and they are still disabled, the program can be rerun with this option to re-enable the mode switching key combinations.

`-saver suspendtime [offtime]` — Set the `suspend` and `off` screen saver inactivity time-outs. The values are in seconds.

`-query` — Display the monitor parameters and extended screen saver time-outs.

SEE ALSO

`XF86Config(4/5)`

AUTHORS

Kaleb S. Keithley (X Consortium), additions and modifications by Jon Tombs, David Dawes, and Joe Moss

X Version 11 Release 6

xvminitoppm

`xvminitoppm`—Convert an XV thumbnail picture to PPM

SYNOPSIS

```
xvminitoppm [xvminipic]
```

DESCRIPTION

Reads an XV *thumbnail* picture (a miniature picture generated by the VisualSchnauzer browser) as input. Produces a portable pixmap as output.

SEE ALSO

`ppm`(5), `xv`(1)

AUTHOR

14 December 1993

xwd

`xwd`—Dump an image of an X window

SYNOPSIS

```
xwd [-debug] [-help] [-nobdrs] [-out file] [-xy] [-frame] [-add value]
[-root j -id id j -name name ] [-icmap] [-screen] [-display display]
```

DESCRIPTION

`xwd` is an X Window System window dumping utility. `xwd` allows X users to store window images in a specially formatted dump file. This file can then be read by various other X utilities for redisplay, printing, editing, formatting, archiving, image processing, and so on. The target window is selected by clicking the pointer in the desired window. The keyboard bell is rung once at the beginning of the dump and twice when the dump is completed.

OPTIONS

`-display display`
: This argument allows you to specify the server to connect to; see `X(1)`.

`-help`
: Print out the `Usage:` command syntax summary.

`-nobdrs`
: This argument specifies that the window dump should not include the pixels that compose the X window border. This is useful in situations in which you may wish to include the window contents in a document as an illustration.

`-out file`
: This argument allows the user to explicitly specify the output file on the command line. The default is to output to standard out.

`-xy`
: This option applies to color displays only. It selects `XY` format dumping instead of the default `Z` format.

`-add value`
: This option specifies a signed value to be added to every pixel.

`-frame`
: This option indicates that the window manager frame should be included when manually selecting a window.

`-root`
: This option indicates that the root window should be selected for the window dump, without requiring the user to select a window with the pointer.

`-id id`
: This option indicates that the window with the specified resource `id` should be selected for the window dump, without requiring the user to select a window with the pointer.

`-name name`
: This option indicates that the window with the specified `WM_NAME` property should be selected for the window dump, without requiring the user to select a window with the pointer.

`-icmap`
: Normally, the colormap of the chosen window is used to obtain RGB values. This option forces the first installed colormap of the screen to be used instead.

`-screen`
: This option indicates that the `GetImage` request used to obtain the image should be done on the root window, rather than directly on the specified window. In this way, you can obtain pieces of other windows that overlap the specified window, and more importantly, you can capture menus or other popups that are independent windows but that appear over the specified window.

DESCRIPTION

`Zcmp` and `zdiff` are used to invoke the `cmp` or the `diff` program on compressed files. All options specified are passed directly to `cmp` or `diff`. If only one file is specified, then the files compared are `file1` and an uncompressed `file1.gz`. If two files are specified, then they are uncompressed if necessary and fed to `cmp` or `diff`. The exit status from `cmp` or `diff` is preserved.

SEE ALSO

`cmp`(1), `diff`(1), `zmore`(1), `zgrep`(1), `znew`(1), `zforce`(1), `gzip`(1), `gzexe`(1)

BUGS

Messages from the `cmp` or `diff` programs refer to temporary filenames instead of those specified.

zeisstopnm

`zeisstopnm`—Convert a Zeiss confocal file into a portable anymap

SYNOPSIS

```
zeisstopnm [-pgm j -ppm][zeissfile]
```

DESCRIPTION

Reads a Zeiss confocal file as input. Produces a portable anymap as output. The type of the output file depends on the input file—if it's greyscale, a `pgm` file will be produced; otherwise, it will be a `ppm` file. The program tells you which type it is writing.

OPTIONS

`-pgm`	Force the output to be a `pgm` file
`-ppm`	Force the output to be a `ppm` file

SEE ALSO

`pnm`(5)

AUTHOR

15 June 1993

zforce

`zforce`—Force a `.gz` extension on all `gzip` files

SYNOPSIS

```
zforce [ name ... ]
```

DESCRIPTION

`zforce` forces a `.gz` extension on all `gzip` files so that `gzip` will not compress them twice. This can be useful for files with names truncated after a file transfer. On systems with a 14-character limitation on filenames, the original name is truncated to make room for the `.gz` suffix. For example, `12345678901234` is renamed to `12345678901.gz`. A filename such as `foo.tgz` is left intact.

Part II:

System Calls

CONFORMS TO

SVID, AT&T, POSIX, X/OPEN, BSD 4.3

SEE ALSO

`stat`(2), `open`(2), `chmod`(2), `chown`(2), `setuid`(2), `setgid`(2)

Linux 1.2.13, 17 March 1996

acct

`acct`—Switches process accounting on or off

SYNOPSIS

```
#include <unistd.h>
int acct(const char *filename);
```

DESCRIPTION

Warning: Since this function is not implemented as of Linux 0.99.11, it will always return -1 and set `errno` to `ENOSYS`. If `acctkit` is installed, the function performs as advertised.

When called with the name of an existing file as argument, accounting is turned on and records for each terminating process are appended to `filename` as it terminates. An argument of `NULL` causes accounting to be turned off.

RETURN VALUE

On success, `0` is returned. On error, `-1` is returned and `errno` is set appropriately.

NOTES

No accounting is produced for programs running when a crash occurs. In particular, nonterminating processes are never accounted for.

SEE ALSO

`acct`(5)

Linux 0.99.11, 10 August 1993

adjtimex

`adjtimex`—Tunes kernel clock

SYNOPSIS

```
#include <sys/timex.h>
int adjtimex(struct timex *buf);
```

DESCRIPTION

Linux uses David Mill's clock adjustment algorithm. `adjtimex` reads and optionally sets adjustment parameters for this algorithm.

`adjtimex` takes a pointer to a `timex` structure, updates kernel parameters from field values, and returns the same structure with current kernel values. This structure is declared as follows:

specified by `serv_addr`, which is an address in the communications space of the socket. Each communications space interprets the `serv_addr`, parameter in its own way. Generally, stream sockets may successfully connect only once; datagram sockets may use `connect` multiple times to change their association. Datagram sockets may dissolve the association by connecting to an invalid address, such as a null address.

RETURN VALUE

If the connection or binding succeeds, `0` is returned. On error, `-1` is returned and `errno` is set appropriately.

ERRORS

See the Linux kernel source code for details.

HISTORY

The `connect` function call first appeared in BSD 4.2.

SEE ALSO

`accept`(2), `bind`(2), `listen`(2), `socket`(2), `getsockname`(2)

Linux 0.99.11, 23 July 1993

dup, dup2

`dup`, `dup2` – Duplicate a file descriptor

SYNOPSIS

```
#include <unistd.h>
int dup(int oldfd);
int dup2(int oldfd,intnewfd);
```

DESCRIPTION

`dup` and `dup2` create a copy of the file descriptor `oldfd`.

The old and new descriptors can be used interchangeably. They share locks, file position pointers and flags; for example, if the file position is modified by using `lseek` on one of the descriptors, the position is also changed for the other.

The two descriptors do not share the close-on-exec flag, however.

`dup` uses the lowest-numbered unused descriptor for the new descriptor.

`dup2` makes `newfd` be the copy of `oldfd`, closing `newfd` first if necessary.

RETURN VALUE

`dup` and `dup2` return the new descriptor, or `-1` if an error occurred (in which case `errno` is set appropriately).

ERRORS

`EBADF`	`oldfd` isn't an open file descriptor, or `newfd` is out of the allowed range for file descriptors.
`EMFILE`	The process already has the maximum number of file descriptors open and tried to open a new one.

WARNING

The error returned by `dup2` is different from that returned by `fcntl(...,F_DUPFD,...)` when `newfd` is out of range. On some systems `dup2` also sometimes returns `EINVAL` like `F_DUPFD`.

ERRORS

EBADF	fd is not a valid file descriptor open for writing.
EROFS, EINVAL	fd is bound to a special file that does not support synchronization.
EIO	An error occurred during synchronization.

CONFORMS TO

POSIX.1b

SEE ALSO

bdflush(2), fdatasync(2), sync(2), update(8), sync(8)

Linux 1.3.85, 13 April 1996

getdents

getdents—Gets directory entries

SYNOPSIS

```
#include <unistd.h>
#include <linux/dirent.h>
#include <linux/unistd.h>
syscall3(int, getdents, uint, fd, struct dirent *, dirp, uint, count);
int getdents(unsigned int fd, struct dirent *dirp, unsigned int count);
```

DESCRIPTION

getdents reads several dirent structures from the directory pointed at by fd into the memory area pointed to by dirp. The parameter count is the size of the memory area.

The dirent structure is declared as follows:

```
struct dirent
             {
                 long d_ino;                 /* inode number */
                 off_t d_off;                /* offset to next dirent */
                 unsigned short d_reclen;    /* length of this dirent */
                 char d_name [NAME_MAX+1];   /* file name (null-terminated) */
             }
```

d_ino is an inode number. d_off is the distance from the start of the directory to the start of the next dirent. d_reclen is the size of this entire *dirent*. d_name is a null-terminated filename.

This call supersedes readdir(2).

RETURN VALUE

On success, the number of bytes read is returned. On end of directory, 0 is returned. On error, -1 is returned and errno is set appropriately.

ERRORS

EBADF	Invalid file descriptor fd.
ENOTDIR	File descriptor does not refer to a directory.

SEE ALSO

readdir(2), readdir(3)

Linux 1.3.6, 22 July 1995

In addition to these errors, `setpriority` will fail with the following:

`EPERM`	A process was located, but neither its effective nor real user ID matched the effective user ID of the caller.
`EACCES`	A nonsuperuser attempted to lower a process priority.

HISTORY

These function calls appeared in BSD 4.2.

SEE ALSO

`nice`(1), `fork`(2), `renice`(8)

getrlimit, getrusage, setrlimit

`getrlimit`, `getrusage`, `setrlimit`—Get/set resource limits and usage

SYNOPSIS

```
#include <sys/time.h>
#include <sys/resource.h>
#include <unistd.h>

int getrlimit (int resource, struct rlimit *rlim);
int getrusage (int who, struct rusage *usage);
int setrlimit (int resource, const struct rlimit *rlim);
```

DESCRIPTION

`getrlimit` and `setrlimit` get and set resource limits. `resource` should be one of the following:

```
RLIMIT_CPU /* CPU time in seconds */
RLIMIT_FSIZE /* Maximum filesize */
RLIMIT_DATA /* max data size */
RLIMIT_STACK /* max stack size */
RLIMIT_CORE /* max core file size */
RLIMIT_RSS /* max resident set size */
RLIMIT_NPROC /* max number of processes */
RLIMIT_NOFILE /* max number of open files */
RLIMIT_MEMLOCK /* max locked-in-memory address space*/
```

A resource may be unlimited if you set the limit to `RLIM_INFINITY`. `RLIMIT_OFILE` is the BSD name for `RLIMIT_NOFILE`.

The `rlimit` structure is defined as follows :

```
struct rlimit
            {
                 int  rlim_cur;
                 int  rlim_max;
            };
```

`getrusage` returns the current resource usages for a `who` of either `RUSAGE_SELF` or `RUSAGE_CHILDREN`:

```
struct rusage
            {
                 struct timeval ru_utime; /* user time used */
                 struct timeval ru_stime; /* system time used */
                 long ru_maxrss;          /* maximum resident set size */
                 long ru_ixrss;           /* integral shared memory size */
```

And the following macros are defined to operate on this :

```
#define   timerisset(tvp)\
                ((tvp)->tv_sec ¦¦ (tvp)->tv_usec)
       #define   timercmp(tvp, uvp, cmp)\
                 ((tvp)->tv_sec cmp (uvp)->tv_sec ¦¦\
                 (tvp)->tv_sec == (uvp)->tv_sec &&\
                 (tvp)->tv_usec cmp (uvp)->tv_usec)
       #define   timerclear(tvp)
                 ((tvp)->tv_sec = (tvp)->tv_usec = 0)
```

If either `tv` or `tz` is null, the corresponding structure is not set or returned.

Only the superuser can use `settimeofday`.

ERRORS

`EPERM`	`settimeofday` is called by someone other than the superuser.
`EINVAL`	Time zone (or something else) is invalid.

CONFORMS TO

BSD 4.3

SEE ALSO

`date`(1), `adjtimex`(2), `time`(2), `ctime`(3), `ftime`(3)

Linux 1.2.4, 15 April 1995

getuid, geteuid

`getuid`, `geteuid`—Get user identity

SYNOPSIS

```
#include <unistd.h>
uid_t getuid(void);
uid_t geteuid(void);
```

DESCRIPTION

`getuid` returns the real user ID of the current process.

`geteuid` returns the effective user ID of the current process.

The real ID corresponds to the ID of the calling process. The effective ID corresponds to the set ID bit on the file being executed.

ERRORS

These functions are always successful.

CONFORMS TO

POSIX, BSD 4.3

SEE ALSO

`setreuid`(2), `setuid`(2)

Linux 0.99.11, 23 July 1993

idle

idle—Makes process 0 idle

SYNOPSIS

```
#include <unistd.h>
void idle(void);
```

DESCRIPTION

idle is an internal system call used during bootstrap. It marks the process's pages as swappable, lowers its priority, and enters the main scheduling loop. idle never returns.

Only process 0 may call idle. Any user process, even a process with superuser permission, will receive EPERM.

RETURN VALUE

idle never returns for process 0, and always returns -1 for a user process.

ERRORS

EPERM	Always, for a user process.

Linux 1.1.46, 21 August 1994

ioctl

ioctl—Controls devices

SYNOPSIS

```
#include <sys/ioctl.h>
int ioctl(int d,intrequest, ...);
```

(The "third" argument is traditionally char *argp and will be so named for this discussion.)

DESCRIPTION

The ioctl function manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (for example, terminals) may be controlled with ioctl requests. The argument d must be an open file descriptor.

An ioctl request has encoded in it whether the argument is an in parameter or out parameter, and the size of the argument argp in bytes. Macros and defines used in specifying an ioctl request are located in the file <sys/ioctl.h>.

RETURN VALUE

On success, 0 is returned. On error, -1 is returned and errno is set appropriately.

ERRORS

EBADF	d is not a valid descriptor.
ENOTTY	d is not associated with a character special device.
ENOTTY	The specified request does not apply to the kind of object that the descriptor *d* references.
EINVAL	request or argp is not valid.

HISTORY

An ioctl function call appeared in version 7 AT&T UNIX.

<include/linux/kd.h>

0x00004B3D	KDUNMAPDISP	void // MORE
0x00004B40	GIO_SCRNMAP	struct f char [E_TABSZ]; g *
0x00004B41	PIO_SCRNMAP	const struct f char [E_TABSZ]; g *
0x00004B69	GIO_UNISCRNMAP	struct f short [E_TABSZ]; g *
0x00004B6A	PIO_UNISCRNMAP	const struct f short [E_TABSZ]; g *
0x00004B66	GIO_UNIMAP	struct unimapdesc * // MORE // I-O
0x00004B67	PIO_UNIMAP	const struct unimapdesc * // MORE
0x00004B68	PIO_UNIMAPCLR	const struct unimapinit *
0x00004B44	KDGKBMODE	int *
0x00004B45	KDSKBMODE	int
0x00004B62	KDGKBMETA	int *
0x00004B63	KDSKBMETA	int
0x00004B64	KDGKBLED	int *
0x00004B65	KDSKBLED	int
0x00004B46	KDGKBENT	struct kbentry * // I-O
0x00004B47	KDSKBENT	const struct kbentry *
0x00004B48	KDGKBSENT	struct kbsentry * // I-O
0x00004B49	KDSKBSENT	const struct kbsentry *
0x00004B4A	KDGKBDIACR	struct kbdiacrs *
0x00004B4B	KDSKBDIACR	const struct kbdiacrs *
0x00004B4C	KDGETKEYCODE	struct kbkeycode * // I-O
0x00004B4D	KDSETKEYCODE	const struct kbkeycode *
0x00004B4E	KDSIGACCEPT	int

<include/linux/lp.h>

0x00000601	LPCHAR	int
0x00000602	LPTIME	int
0x00000604	LPABORT	int
0x00000605	LPSETIRQ	int
0x00000606	LPGETIRQ	int *
0x00000608	LPWAIT	int
0x00000609	LPCAREFUL	int
0x0000060A	LPABORTOPEN	int
0x0000060B	LPGETSTATUS	int *
0x0000060C	LPRESET	void
0x0000060D	LPGETSTATS	struct lp stats *

<include/linux/mroute.h>

0x000089E0	SIOCGETVIFCNT	struct sioc_vif_req * // I-O
0x000089E1	SIOCGETSGCNT	struct sioc_sg_req * // I-O

MORE ARGUMENTS

Some `ioctl`s take a pointer to a structure that contains additional pointers. These are documented here in alphabetical order.

`CDROMREADAUDIO` takes an input pointer `const struct cdrom read audio *`. The `buf` field points to an output buffer of length `nframes * CD FRAMESIZE RAW`.

`CDROMREADCOOKED`, `CDROMREADMODE1`, `CDROMREADMODE2`, and `CDROM-READRAW` take an input pointer `const struct cdrom msf *`. They use the same pointer as an output pointer to `char []`. The length varies by request. For `CDROMREADMODE1`, most drivers use `CD_FRAMESIZE`, but the optics storage driver uses `OPT BLOCKSIZE` instead (both have the numerical value 2048).

`CDROMREADCOOKED`	`char [CD_FRAMESIZE]`
`CDROMREADMODE1`	`char [CD_FRAMESIZE or OPT_BLOCKSIZE]`
`CDROMREADMODE2`	`char [CD_FRAMESIZE_RAW0]`
`CDROMREADRAW`	`char [CD_FRAMESIZE_RAW]`

`EQL_ENSLAVE`, `EQL_EMANCIPATE`, `EQL_GETSLAVECFG`, `EQL_SETSLAVECFG`, `EQL_GETMASTERCFG`, and `EQL_SETMASTERCFG` take a `struct ifreq *`. The `ifr data` field is a pointer to another structure as follows:

`EQL_ENSLAVE`	`const struct slaving_request *`
`EQL_EMANCIPATE`	`const struct slaving_request *`
`EQL_GETSLAVECFG`	`struct slave_config *` // I-O
`EQL_SETSLAVECFG`	`const struct slave_config *`
`EQL_GETMASTERCFG`	`struct master_config *`
`EQL_SETMASTERCFG`	`const struct master_config *`

`FDRAWCMD` takes a `struct floppy raw cmd *`. If `flags & FD RAW WRITE` is nonzero, then `data` points to an input buffer of length `length`. If `flags & FD RAW READ` is nonzero, then `data` points to an output buffer of length `length`.

`GIO_FONTX` and `PIO_FONTX` take a `struct console font desc *` or a `const struct console_font_desc *`, respectively. `chardata` points to a buffer of `char [charcount]`. This is an output buffer for `GIO_FONTX` and an input buffer for `PIO_FONTX`.

`GIO_UNIMAP` and `PIO_UNIMAP` take a `struct unimapdesc *` or `a const struct unimapdesc *`, respectively. `entries` points to a buffer of `struct unipair [entry ct]`. This is an output buffer for `GIO_UNIMAP` and an input buffer for `PIO_UNIMAP`.

`KDADDIO`, `KDDELIO`, `KDDISABIO`, and `KDENABIO` enable or disable access to I/O ports. They are essentially alternate interfaces to `ioperm`.

`KDMAPDISP` and `KDUNMAPDISP` enable or disable memory mappings or I/O port access. They are not implemented in the kernel.

`SCSI_IOCTL_PROBE_HOST` takes an input pointer `const int *`, which is a length. It uses the same pointer as an output pointer to a `char []` buffer of this length.

`SIOCADDRT` and `SIOCDELRT` take an input pointer whose type depends on the protocol:

Most protocols	`const struct rtentry *`
AX.25	`const struct ax25_route *`
NET/ROM	`const struct nr_route_struct *`

`SIOCGIFCONF` takes a `struct ifconf *`. The `ifc buf` field points to a buffer of length `ifc len` bytes, into which the kernel writes a list of type `struct ifreq []`.

`SIOCSIFHWADDR` takes an input pointer whose type depends on the protocol:

Most protocols	`const struct ifreq *`
AX.25	`const char [AX25_ADDR_LEN]`

`TIOCLINUX` takes a `const char *`. It uses this to distinguish several independent subcases. In the following table, `N + foo` means `foo` after an *N*-byte pad. `struct selection` is implicitly defined in `drivers/char/selection.c`:

The file type should be one of S_IFREG, S_IFCHR, S_IFBLK, or S_IFIFO to specify a normal file (which will be created empty), character special file, block special file, or FIFO (named pipe), respectively, or 0, which will create a normal file.

If the file type is S_IFCHR or S_IFBLK, then dev specifies the major and minor numbers of the newly created device special file; otherwise, it is ignored.

The newly created node will be owned by the effective UID of the process. If the directory containing the node has the set group ID bit set, or if the filesystem is mounted with BSD group semantics, the new node will inherit the group ownership from its parent directory; otherwise it will be owned by the effective GID of the process.

RETURN VALUE

mknod returns 0 on success, or -1 if an error occurred (in which case, errno is set appropriately).

ERRORS

EPERM	mode requested creation of something other than a FIFO (named pipe), and the caller is not the superuser; also returned if the filesystem containing pathname does not support the type of node requested.
EINVAL	mode requested creation of something other than a normal file, device special file or FIFO.
EEXIST	pathname already exists.
EFAULT	pathname points outside your accessible address space.
EACCES	The parent directory does not allow write permission to the process, or one of the directories in pathname did not allow search (execute) permission.
ENAMETOOLONG	pathname was too long.
ENOENT	A directory component in pathname does not exist or is a dangling symbolic link.
ENOTDIR	A component used as a directory in pathname is not, in fact, a directory.
ENOMEM	Insufficient kernel memory was available.
EROFS	pathname refers to a file on a read-only filesystem and write access was requested.
ELOOP	pathname contains a reference to a circular symbolic link, that is, a symbolic link whose expansion contains a reference to itself.
ENOSPC	The device containing pathname has no room for the new node.

BUGS

In some older versions of Linux (for example, 0.99pl7) all the normal filesystems sometime allow the creation of two files in the same directory with the same name. This occurs only rarely and only on a heavily loaded system. It is believed that this bug was fixed in the Minix filesystem in Linux 0.99pl8 prerelease; and it is hoped that it was fixed in the other filesystems shortly afterward.

mknod cannot be used to create directories or socket files, and cannot be used to create normal files by users other than the superuser.

There are many infelicities in the protocol underlying NFS.

SEE ALSO

read(2), write(2), fcntl(2), close(2), unlink(2), open(2), mkdir(2), stat(2), umask(2), mount(2), socket(2), fopen(3)

Linux 1.0, 29 March 1994

mlock

mlock—Disables paging for some parts of memory

```
int msgsnd ( int msqid, struct msgbuf *msgp",int msgsz,int msgflg );

int msgrcv ( int msqid, struct msgbuf *msgp,int msgsz,long msgtyp,int msgflg );
```

DESCRIPTION

To send or receive a message, the calling process allocates a structure that looks like the following:

```
struct msgbuf {
  long mtype; /* message type, must be > 0 */
  char mtext[1]; /* message data */
};
```

but with an array `mtext` of size `msgsz`, a nonnegative integer value. The structure member `mtype` must have a strictly positive integer value that can be used by the receiving process for message selection (see the section about `msgrcv`).

The calling process must have write access permissions to send and read access permissions to receive a message on the queue.

The `msgsnd` system call queues a copy of the message pointed to by the `msgp` argument on the message queue whose identifier is specified by the value of the `msqid` argument.

The argument `msgflg` specifies the system call behavior if, queuing the new message, will require more than `msg_qbytes` in the queue. Asserting `IPC_NOWAIT`, the message will not be sent and the system call fails, returning with `errno` set to `EAGAIN`. Otherwise the process is suspended until the condition for the suspension no longer exists (in which case the message is sent and the system call succeeds), or the queue is removed (in which case the system call fails with `errno` set to `EIDRM`), or the process receives a signal that has to be caught (in which case the system call fails with `errno` set to `EINTR`).

Upon successful completion, the message queue data structure is updated as follows:

`msg_lspid`	Set to the process D of the calling process.
`msg_qnum`	Incremented by 1.
`msg_stime`	Set to the current time.

The system call `msgrcv` reads a message from the message queue specified by `msqid` into the `msgbuf` pointed to by the `msgp` argument, removing from the queue, on success, the read message.

The argument `msgsz` specifies the maximum size in bytes for the member `mtext` of the structure pointed to by the `msgp` argument. If the message text has length greater than `msgsz`, then if the `msgflg` argument asserts `MSG_NOERROR`, the message text will be truncated (and the truncated part will be lost); otherwise, the message isn't removed from the queue and the system call fails, returning with `errno` set to `E2BIG`.

The argument `msgtyp` specifies the type of message requested as follows:

If `msgtyp` is `0`, the message on the queue's front is read.

If `msgtyp` is greater than `0`, the first message on the queue of type `msgtyp` is read if `MSG_EXCEPT` isn't asserted by the `msgflg` argument; otherwise, the first message on the queue of type not equal to `msgtyp` will be read.

If `msgtyp` is less than `0`, the first message on the queue with the lowest type less than or equal to the absolute value of `msgtyp` will be read.

The `msgflg` argument asserts none, one, or more (OR–ing them) among the following flags:

`IPC_NOWAIT`	For immediate return if no message of the requested type is on the queue. The system call fails with `errno` set to `ENOMSG`.
`MSG_EXCEPT`	Used with `msgtyp` greater than `0` to read the first message on the queue with message type that differs from `msgtyp`.
`MSG_NOERROR`	To truncate the message text if longer than `msgsz` bytes.

SEE ALSO

`ipc`(5), `msgctl`(2), `msgget`(2), `msgrcv`(2), `msgsnd`(2)

Linux 0.99.13, 1 November 1993

msync

`msync`—Synchronizes a file with a memory map

SYNOPSIS

```
#include <unistd.h>
#include <sys/mman.h>

#ifdef_POSIX_MAPPED_FILES
#ifdef_POSIX_SYNCHRONIZED_IO

int msync(const void *start, size_t length,int flags);
#endif
#endif
```


DESCRIPTION

`msync` flushes changes made to the in-core copy of a file that was mapped into memory using `mmap`(2) back to disk. Without use of this call, there is no guarantee that changes are written back before `munmap`(2) is called. To be more precise, the part of the file that corresponds to the memory area starting at `start` and having length `length` is updated. The `flags` argument may have the bits `MS_ASYNC`, `MS_SYNC`, and `MS_INVALIDATE` set, but not both `MS_ASYNC` and `MS_SYNC`. `MS_ASYNC` specifies that an update be scheduled, but the call returns immediately. `MS_SYNC` asks for an update and waits for it to complete. `MS_INVALIDATE` asks to invalidate other mappings of the same file (so that they can be updated with the fresh values just written).

RETURN VALUE

On success, `0` is returned. On error, `-1` is returned and `errno` is set appropriately.

ERRORS

`EINVAL`	`start` is not a multiple of `PAGESIZE`, or any bit other than `MS_ASYNC` ¦ `MS_INVALIDATE` ¦ `MS_SYNC` is set in `flags`.
`EFAULT`	The indicated memory (or part of it) was not mapped.

CONFORMS TO

POSIX.4.

SEE ALSO

`mmap`(2), B.O. Gallmeister, POSIX.4, O'Reilly, pp. 128–129, 389–391.

Linux 1.3.86, 12 April 1996

munlock

`munlock`—Reenables paging for some parts of memory

SYNOPSIS

```
#include <sys/mman.h>
int munlock(const void *addr, size_t len);
```

open, creat

`open`, `creat`—Open and possibly create a file or device

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname,int flags);
int open(const char *pathname,int flags,mode_t mode);
int creat(const char *pathname,mode_t mode);
```

DESCRIPTION

`open` attempts to open a file and return a file descriptor (a small, nonnegative integer for use in read, write, and so on).

`flags` is one of `O_RDONLY`, `O_WRONLY`, or `O_RDWR`, which request opening the file read-only, write-only, or read/write, respectively.

`flags` may also be bitwise-ORed with one or more of the following:

`O_CREAT`	If the file does not exist it will be created.
`O_EXCL`	When used with `O_CREAT`, if the file already exists, it is an error and the open will fail. *See* the "Bugs" section, though.
`O_NOCTTY`	If `pathname` refers to a terminal device—see `tty(4)`—it will not become the process's controlling terminal even if the process does not have one.
`O_TRUNC`	If the file already exists, it will be truncated.
`O_APPEND`	The file is opened in append mode. Initially, and before each `write`, the file pointer is positioned at the end of the file, as if with `lseek`.
`O_NONBLOCK` or `O_NDELAY`	The file is opened in nonblocking mode. Neither the `open` nor any subsequent operations on the file descriptor that is returned will cause the calling process to wait.
`O_SYNC`	The file is opened for synchronous I/O. Any `writes` on the resulting file descriptor will block the calling process until the data has been physically written to the underlying hardware. *See* the "Bugs" section, though.

Some of these optional flags can be altered using `fcntl` after the file has been opened.

`mode` specifies the permissions to use if a new file is created. It is modified by the process's `umask` in the usual way: The permission of the created file is `(mode & ~umask)`.

The following symbolic constants are provided for `mode`:

`S_IRWXU`	00700 user (file owner) has read, write, and execute permission.
`S_IRUSR` (`S_IREAD`)	00400 user has read permission.
`S_IWUSR` (`S_IWRITE`)	00200 user has write permission.
`S_IXUSR` (`S_IEXEC`)	00100 user has execute permission.
`S_IRWXG`	00070 group has read, write, and execute permission.
`S_IRGRP`	00040 group has read permission.
`S_IWGRP`	00020 group has write permission.
`S_IXGRP`	00010 group has execute permission.
`S_IRWXO`	00007 others have read, write, and execute permission.
`S_IROTH`	00004 others have read permission.
`S_IWOTH`	00002 others have write permission.
`S_IXOTH`	00001 others have execute permission.

`mode` should always be specified when `O_CREAT` is in the `flags`, and is ignored otherwise.

`creat` is equivalent to open with `flags` equal to `O_CREAT|O_WRONLY|O_TRUNC`.

SYNOPSIS

```
#include <unistd.h>
int pipe(int filedes[2]);
```

DESCRIPTION

`pipe` creates a pair of file descriptors, pointing to a pipe inode, and places them in the array pointed to by `filedes`. `filedes[0]` is for reading, `filedes[1]` is for writing.

RETURN VALUE

On success, `0` is returned. On error, `-1` is returned and `errno` is set appropriately.

ERRORS

`EMFILE`	Too many file descriptors are in use by the process.
`ENFILE`	The system file table is full.
`EFAULT`	`filedes` is not valid.

SEE ALSO

`read`(2), `write`(2), `fork`(2), `socketpair`(2)

Linux 0.99.11, 23 July 1993

profil

`profil`—Execution time profile

SYNOPSIS

```
#include <unistd.h>
int profil(char *buf, int bufsiz,int offset,int scale);
```

DESCRIPTION

Under Linux 0.99.11, `profil` is not implemented in the kernel. Instead, the DLL 4.4.1 libraries provide a user-space implementation.

`buf` points to `bufsiz` bytes of core. Every virtual 10 milliseconds, the user's program counter (PC) is examined: `offset` is subtracted and the result is multiplied by `scale`. If this address is in `buf`, the word pointed to is incremented.

If `scale` is less than `2` or `bufsiz` is `0`, profiling is disabled.

RETURN VALUE

`0` is always returned.

BUGS

`profil` cannot be used on a program that also uses `ITIMER_PROF` itimers.

Calling `profil` with an invalid `buf` will result in a core dump.

True kernel profiling provides more accurate results.

SEE ALSO

`gprof`(1), `setitimer`(2), `signal`(2), `sigaction`(2)

Linux 0.99.11, 23 July 1993

BUGS

As of Linux 1.3.81, `SCHED_RR` has not yet been tested carefully and might not behave exactly as described or required by POSIX.1b.

SEE ALSO

`sched_setparam`(2), `sched_getparam`(2), `sched_yield`(2), `sched_get_priority_max`(2), `sched_get_priority_min`(2), `nice`(2), `setpriority`(2), `getpriority`(2), `mlockall`(2), `munlockall`(2), `mlock`(2), `munlock`(2).

Programming for the Real World—POSIX.4 by Bill O. Gallmeister, O'Reilly & Associates, Inc., ISBN 1-56592-074-0

IEEE Std 1003.1b-1003 (POSIX.1b standard)

ISO/IEC 9945-1:1996—This is the new 1996 revision of POSIX.1, which contains in one single standard POSIX.1(1990), POSIX.1b(1993), POSIX.1c(1995), and POSIX.1i(1995).

Linux 1.3.81, 10 April 1996

sched_yield

`sched_yield`—Yields the processor

SYNOPSIS

```
#include <sched.h>
int sched_yield(void);
```

DESCRIPTION

A process can relinquish the processor voluntarily without blocking by calling `sched_yield`. The process will then be moved to the end of the queue for its static priority and a new process gets to run.

Note: If the current process is the only process in the highest priority list at that time, this process will continue to run after a call to `sched_yield`.

POSIX systems on which `sched_yield` is available define `_POSIX_PRIORITY_SCHEDULING` in `<unistd.h>`.

RETURN VALUE

On success, `sched_yield` returns `0`. On error, `-1` is returned, and `errno` is set appropriately.

STANDARDS

POSIX.1b (formerly POSIX.4)

SEE ALSO

`sched_setscheduler`(2) for a description of Linux scheduling

Programming for the Real World—POSIX.4 by Bill O. Gallmeister, O'Reilly & Associates, Inc., ISBN 1-56592-074-0

IEEE Std 1003.1b-1993 (POSIX.1b standard)

ISO/IEC 9945-1:1996

Linux 1.3.81, 10 April 1996

select, FD_CLR, FD_ISSET, FD_SET, FD_ZERO

`select`, `FD_CLR`, `FD_ISSET`, `FD_SET`, `FD_ZERO`—Synchronous I/O multiplexing

Process groups are used for distribution of signals, and by terminals to arbitrate requests for their input; processes that have the same process group as the terminal are foreground and may read, whereas others will block with a signal if they attempt to read.

These calls are thus used by programs such as csh(1) to create process groups in implementing job control. The TIOCGPGRP and TIOCSPGRP calls described in termios(4) are used to get/set the process group of the control terminal.

RETURN VALUE

On success, setpgid and setpgrp return 0. On error, -1 is returned, and errno is set appropriately.

getpgid returns a process group on success. On error, -1 is returned, and errno is set appropriately.

getpgrp always returns the current process group.

ERRORS

EINVAL	pgid is less than 0.
EPERM	Various permission violations.
ESRCH	pid does not match any process.

CONFORMS TO

The functions setpgid and getpgrp conform to POSIX.1. The function setpgrp is from BSD 4.2. I have no information on the source of getpgid.

SEE ALSO

getuid(2), setsid(2), tcsetpgrp(3), termios(4)

Linux 1.2.4, 15 April 1995

setregid, setegid

setregid, setegid—Sets real and/or effective group ID

SYNOPSIS

```
#include <unistd.h>
int setregid(gid_t rgid, gid_t egid);
int setegid(gid_t egid);
```

DESCRIPTION

setregid sets real and effective group IDs of the current process. Unprivileged users may change the real group ID to the effective group ID, and vice versa.

Prior to Linux 1.1.38, the saved ID paradigm, when used with setregid or setegid, was broken. Starting at 1.1.38, it is also possible to set the effective group ID from the saved user ID.

Only the superuser may make other changes.

Supplying a value of -1 for either the real or effective group ID forces the system to leave that ID unchanged.

Currently (libc-4.x.x), setegid(egid) is functionally equivalent to setregid(-1, egid).

If the real group ID is changed or the effective group ID is set to a value not equal to the previous real group ID, the saved group ID will be set to the new effective group ID.

RETURN VALUE

On success, 0 is returned. On error, -1 is returned, and errno is set appropriately.

```
pid_t wait(int *status)
pid_t waitpid(pid_t pid,int*status,int options);
```

DESCRIPTION

The `wait` function suspends execution of the current process until a child has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal-handling function. If a child has already exited by the time of the call (a so–called *zombie* process), the function returns immediately. Any system resources used by the child are freed.

The `waitpid` function suspends execution of the current process until a child as specified by the `pid` argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal-handling function. Just as with `wait`, if a child requested by `pid` has already exited by the time of the call, the function returns immediately. Any system resources used by the child are freed.

The value of `pid` can be one of the following:

`< -1`	Wait for any child process whose process group ID is equal to the absolute value of `pid`.
`-1`	Wait for any child process; this is the same behavior that `wait` exhibits.
`0`	Wait for any child process whose process group ID is equal to that of the calling process.
`> 0`	Wait for the child whose process ID is equal to the value of `pid`.

The value of `options` is an OR of zero or more of the following constants:

`WNOHANG`	Return immediately if no child has exited.
`WUNTRACED`	Also return for children that are stopped and whose status has not been reported.

If `status` is not `NULL`, `wait` or `waitpid` stores status information in the location pointed to by `statloc`.

This status can be evaluated with the following macros (these macros take the `stat` buffer as an argument—not a pointer to the buffer!):

`WIFEXITED(status)`	Is nonzero if the child exited normally.
`WEXITSTATUS(status)`	Evaluates to the least significant eight bits of the return code of the child that terminated, which may have been set as the argument to a call to `exit()` or as the argument for a return statement in the main program. This macro can only be evaluated if `WIFEXITED` returned nonzero.
`WIFSIGNALED(status)`	Returns true if the child process exited because of a signal that was not caught.
`WTERMSIG(status)`	Returns the number of the signal that caused the child process to terminate. This macro can only be evaluated if `WIFSIGNALED` returned nonzero.
`WIFSTOPPED(status)`	Returns true if the child process that caused the return is currently stopped; this is only possible if the call was done using `WUNTRACED`.
`WSTOPSIG(status)`	Returns the number of the signal that caused the child to stop. This macro can only be evaluated if `WIFSTOPPED` returned nonzero.

RETURN VALUE

The process ID of the child that exited returns `-1` on error or `0` if `WNOHANG` was used and no child was available (in which case `errno` is set to an appropriate value).

ERRORS

`ECHILD`	If the child process specified in `pid` does not exist.
`EPERM`	If the effective user ID of the calling process does not match that of the process being waited for, and the effective user ID of the calling process is not that of the superuser.
`ERESTARTSYS`	If `WNOHANG` was not set and an unblocked signal or a `SIGCHLD` was caught; this is an extension to the POSIX.1 standard.

`WIFSTOPPED(*status)`	Returns true if the child process that caused the return is currently stopped; this is only possible if the call was done using `WUNTRACED`.
`WSTOPSIG(*status)`	Returns the number of the signal that caused the child to stop. This macro can only be evaluated if `WIFSTOPPED` returned nonzero. If `rusage` is not `NULL`, the `struct rusage` as defined in `<sys/resource.h>` it points to will be filled with accounting information. See `getrusage`(2) for details.

RETURN VALUE

These calls return the process ID of the child that exited, `-1` on error, or `0` if `WNOHANG` was used and no child was available (in which case `errno` will be set appropriately).

ERRORS

`ECHILD`	If the child process specified in `pid` does not exist.
`EPERM`	If the effective user ID of the calling process does not match that of the process being waited for, and the effective user ID of the calling process is not that of the superuser.
`ERESTARTSYS`	If `WNOHANG` was not set and an unblocked signal or `SIGCHLD` was caught; this is an extension to the POSIX.1 standard.

CONFORMS TO

POSIX.1

SEE ALSO

`signal`(2), `getrusage`(2), `wait`(2), `signal`(7)

Linux, 24 July 1993

write

`write`—Writes to a file descriptor

SYNOPSIS

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t count);
```

DESCRIPTION

`write` writes up to `count` bytes to the file referenced by the file descriptor `fd` from the buffer starting at `buf`. POSIX requires that a `read()` that can be proved to occur after a `write()` returned returns the new data. Note that not all filesystems are POSIX conforming.

RETURN VALUE

On success, the number of bytes written is returned (`0` indicates nothing was written). On error, `-1` is returned, and `errno` is set appropriately. If `count` is `0` and the file descriptor refers to a regular file, `0` will be returned without causing any other effect. For a special file, the results are not portable.

ERRORS

`EBADF`	`fd` is not a valid file descriptor or is not open for writing.
`EINVAL`	`fd` is attached to an object that is unsuitable for writing.
`EFAULT`	`buf` is outside your accessible address space.

Part III:

Library Functions

SYNOPSIS

```
#include <stdlib.h>
int abs(int j);
```

DESCRIPTION

The `abs()` function computes the absolute value of the integer argument `j`.

RETURN VALUE

Returns the absolute value of the integer argument.

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

NOTES

Trying to take the absolute value of the most negative integer is not defined.

SEE ALSO

`ceil`(3), `floor`(3), `fabs`(3), `labs`(3), `rint`(3)

GNU, 6 June 1993

acos

`acos`—Arc cosine function

SYNOPSIS

```
#include <math.h>
double acos(double x);
```

DESCRIPTION

The `acos()` function calculates the arc cosine of `x`; that is the value whose cosine is `x`. If `x` falls outside the range –1 to 1, `acos()` fails and `errno` is set.

RETURN VALUE

The `acos()` function returns the arc cosine in radians; the value is mathematically defined to be between 0 and pi (inclusive).

ERRORS

`EDOM` `x` is out of range.

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO

`asin`(3), `atan`(3), `atan2`(3), `cos`(3), `sin`(3), `tan`(3)

8 June 1993

acosh

`acosh`—Inverse hyperbolic cosine function

DESCRIPTION

`assert()` prints an error message to standard output and terminates the program by calling `abort()` if `expression` is false (that is, evaluates to `0`). This only happens when the macro `NDEBUG` is undefined.

RETURN VALUE

No value is returned.

CONFORMS TO

ISO9899 (ANSI C)

BUGS

`assert()` is implemented as a macro; if the expression tested has side effects, program behavior will be different depending on whether `NDEBUG` is defined. This may create Heisenbugs, which go away when debugging is turned on.

SEE ALSO

`exit`(3), `abort`(3)

GNU, 4 April 1993

atan

`atan`—Arc tangent function

SYNOPSIS

```
#include <math.h>
double atan(double x);
```

DESCRIPTION

The `atan()` function calculates the arc tangent of `x`—that is, the value whose tangent is `x`.

RETURN VALUE

The `atan()` function returns the arc tangent in radians, and the value is mathematically defined to be between `-PI/2` and `PI/2` (inclusive).

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO

`acos`(3), `asin`(3), `atan2`(3), `cos`(3), `sin`(3), `tan`(3)

8 June 1993

atan2

`atan2`—Arc tangent function of two variables

SYNOPSIS

```
#include <math.h>
double atan2(double y, double x);
```

DESCRIPTION

The exec family of functions replaces the current process image with a new process image. The functions described in this manual page are front ends for the function execve(2). (See the manual page for execve for detailed information about the replacement of the current process.)

The initial argument for these functions is the pathname of a file that is to be executed.

The const char *arg and subsequent ellipses in the execl, execlp, and execle functions can be thought of as arg0, arg1, ..., argn. Together they describe a list of one or more pointers to null-terminated strings that represent the argument list available to the executed program. The first argument, by convention, should point to the file name associated with the file being executed. The list of arguments must be terminated by a NULL pointer.

The exect, execv, and execvp functions provide an array of pointers to null-terminated strings that represent the argument list available to the new program. The first argument, by convention, should point to the filename associated with the file being executed. The array of pointers must be terminated by a NULL pointer.

The execle and exect functions also specify the environment of the executed process by following the NULL pointer that terminates the list of arguments in the parameter list or the pointer to the argv array with an additional parameter. This additional parameter is an array of pointers to null-terminated strings and must be terminated by a NULL pointer. The other functions take the environment for the new process image from the external variable environ in the current process.

Some of these functions have special semantics.

The functions execlp and execvp will duplicate the actions of the shell in searching for an executable file if the specified filename does not contain a slash (/) character. The search path is the path specified in the environment by the PATH variable. If this variable isn't specified, the default path :/bin:/usr/bin: is used (is this true for Linux?). In addition, certain errors are treated specially.

If permission is denied for a file (the attempted execve returned EACCES), these functions will continue searching the rest of the search path. If no other file is found, however, they will return with the global variable errno set to EACCES.

If the header of a file isn't recognized (the attempted execve returned ENOEXEC), these functions will execute the shell with the path of the file as its first argument. (If this attempt fails, no further searching is done.)

If the file is currently busy (the attempted execve returned ETXTBUSY), these functions will sleep for several seconds, periodically re-attempting to execute the file. (Is this true for Linux?)

The function exect executes a file with the program-tracing facilities enabled (see ptrace(2)).

RETURN VALUES

If any of the exec functions returns, an error will have occurred. The return value is -1, and the global variable errno will be set to indicate the error.

FILES

/bin/sh

ERRORS

execl, execle, execlp, and execvp may fail and set errno for any of the errors specified for the library functions execve(2) and malloc(3).

exect and execv may fail and set errno for any of the errors specified for the library function execve(2).

SEE ALSO

sh(1), execve(2), fork(2), trace(2), environ(5), ptrace(2)

COMPATIBILITY

Historically, the default path for the execlp and execvp functions was /bin:/usr/bin. This was changed to place the current directory last to enhance system security.

SYNOPSIS

```
#include <stdio.h>
size_t fread(void *ptr, size_t size, size_t nmemb,FILE*stream);
size_t fwrite(void *ptr, size_t size, size_t nmemb,FILE*stream);
```

DESCRIPTION

The function `fread` reads `nmemb` elements of data, each `size` bytes long, from the stream pointed to by `stream`, storing them at the location given by `ptr`.

The function `fwrite` writes `nmemb` elements of data, each `size` bytes long, to the stream pointed to by `stream`, obtaining them from the location given by `ptr`.

RETURN VALUES

`fread` and `fwrite` return the number of items successfully read or written (that is, not the number of characters). If an error occurs, or the end-of-file is reached, the return value is a short item count (or `0`).

`fread` does not distinguish between end-of-file and error, and callers must use `feof`(3) and `ferror`(3) to determine which occurred.

SEE ALSO

`feof`(3), `ferror`(3), `read`(2), `write`(2)

STANDARDS

The functions `fread` and `fwrite` conform to ANSI X3.159-1989 (“ANSI C”).

BSD Man Page, 17 May 1996

frexp

`frexp`—Converts floating-point number to fractional and integral components

SYNOPSIS

```
#include <math.h>
double frexp(double x, int *exp);
```

DESCRIPTION

The `frexp()` function is used to split the number `x` into a normalized fraction and an exponent that is stored in `exp`.

RETURN VALUE

The `frexp()` function returns the normalized fraction. If the argument `x` is not `0`, the normalized fraction is `x` times a power of 2, and is always in the range ½ (inclusive) to 1 (exclusive). If `x` is `0`, the normalized fraction is 0, and `0` is stored in `exp`.

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO

`ldexp`(3), `modf`(3)

GNU, 6 June 1993

fgetpos, fseek, fsetpos, ftell, rewind

`fgetpos`, `fseek`, `fsetpos`, `ftell`, `rewind`—Reposition a stream

RETURN VALUE

The `gcvt()` function returns the address of the string pointed to by `buf`.

SEE ALSO

`ecvt`(3), `fcvt`(3), `sprintf`(3)

29 March 1993

getcwd, get_current_dir_name, getwd

`getcwd`, `get_current_dir_name`, `getwd`—Get current working directory

SYNOPSIS

```
#include <unistd.h>
char *getcwd(char *buf, size_t size);
char *get_current_working_dir_name(void);
char *getwd(char *buf);
```


DESCRIPTION

The `getcwd()` function copies the absolute pathname of the current working directory to the array pointed to by `buf`, which is of length `size`.

If the current absolute pathname would require a buffer longer than `size` elements, `NULL` is returned, and `errno` is set to `ERANGE`; an application should check for this error, and allocate a larger buffer if necessary.

As an extension to the POSIX.1 standard, `getcwd()` allocates the buffer dynamically using `malloc()` if `buf` is `NULL` on call. In this case, the allocated buffer has the length `size` unless `size` is less than 0, when `buf` is allocated as large as necessary. It is possible (and, indeed, advisable) to free the buffers if they have been obtained this way.

`get_current_dir_name`, which is only prototyped if `__USE_GNU` is defined, will `malloc`(3) an array big enough to hold the current directory name. If the environment variable `PWD` is set, and its value is correct, that value will be returned.

`getwd`, which is only prototyped if `__USE_BSD` is defined, will not `malloc`(3) any memory. The `buf` argument should be a pointer to an array at least `PATH_MAX` bytes long. `getwd` returns only the first `PATH_MAX` bytes of the actual pathname.

RETURN VALUE

`NULL` on failure (for example, if the current directory is not readable), with `errno` set accordingly, and `buf` on success.

CONFORMS TO

POSIX.1

SEE ALSO

`chdir`(2), `free`(3), `malloc`(3).

GNU, 21 July 1993

getdirentries

`getdirentries`—Gets directory entries in a filesystem-independent format

SYNOPSIS

```
#define __USE_BSD  or  #define __USE_MISC
#include <dirent.h>
ssize_t getdirentries(int fd, char *buf, size_t nbytes ,offt *basep);
```

FILES

The `/etc/passwd` password database file `/etc/utmp` (or `/var/adm/utmp`, or wherever your `utmp` file lives these days—the proper location depends on your `libc` version)

CONFORMS TO

POSIX.1. System V has a `cuserid` function that uses the real user ID rather than the effective user ID. The `cuserid` function was included in the 1988 version of POSIX, but was removed from the 1990 version.

BUGS

Unfortunately, it is often rather easy to fool `getlogin()`. Sometimes it does not work at all, because some program messed up the `utmp` file. Often, it gives only the first eight characters of the login name. The user currently logged in on the controlling tty of your program need not be the user who started it.

Nobody knows precisely what `cuserid()` does; so

- Avoid it in portable programs
- Avoid it altogether
- Use `getpwuid` (`geteuid()`) instead, if that is what you meant.

Simply, *do not use* `cuserid()`.

SEE ALSO

`geteuid(2)`, `getuid(2)`

Linux 1.2.13, 3 September 1995

getmntent, setmntent, addmntent, endmntent, hasmntopt

`getmntent`, `setmntent`, `addmntent`, `endmntent`, `hasmntopt`—Get filesystem descriptor file entry

SYNOPSIS

```
#include <stdio.h>
#include <mntent.h>
FILE *setmntent(const char *filep, const char *type);
struct mntent *getmntent(FILE *filep);
int addmntent(FILE *filep, const struct mntent *mnt);
int endmntent(FILE *filep);
char *hasmntopt(const struct mntent *mnt, const char *opt);
```

DESCRIPTION

These routines are used to access the filesystem description file `/etc/fstab` and the mounted filesystem description file `/etc/mstab`.

The `setmntent()` function opens the filesystem description file `filep` and returns a file pointer that can be used by `getmntent()`. The argument `type` is the type of access required and can take the same values as the `mode` argument of `fopen`(3).

The `getmntent()` function reads the next line from the filesystem description file `filep` and returns a pointer to a structure containing the broken-out fields from a line in the file. The pointer points to a static area of memory that is overwritten by subsequent calls to `getmntent()`.

The `addmntent()` function adds the `mntent` structure `mnt` to the end of the open file filep.

The `endmntent()` function closes the filesystem description file `filep`.

The `hasmntopt()` function scans the `mnt_opts` field of the `mntent` structure `mnt` for a substring that matches `opt`. (See `<mntent.h>` for valid mount options.)

DESCRIPTION

The `getopt()` function parses the command-line arguments. Its arguments `argc` and `argv` are the argument count and array as passed to the `main()` function on program invocation. An element of `argv` that starts with `-` (and is not exactly `-` or `--`) is an option element. The characters of this element (aside from the initial `-`) are option characters. If `getopt()` is called repeatedly, it returns successively each of the option characters from each of the option elements.

If `getopt()` finds another option character, it returns that character, updating the external variable `optind` and a static variable `nextchar` so that the next call to `getopt()` can resume the scan with the following option character or `argv` element.

If there are no more option characters, `getopt()` returns `EOF`. Then `optind` is the index in `argv` of the first `argv` element that is not an option.

`optstring` is a string containing the legitimate option characters. If such a character is followed by a colon, the option requires an argument, so `getopt` places a pointer to the following text in the same `argv` element, or the text of the following `argv` element, in `optarg`. Two colons mean an option takes an optional `arg`; if there is text in the current `argv` element, it is returned in `optarg`; otherwise, `optarg` is set to `0`.

By default, `getargs()` permutes the contents of `argv` as it scans, so that eventually all the non-options are at the end. Two other modes are also implemented. If the first character of `optstring` is `+` or the environment variable `POSIXLY_CORRECT` is set, option processing stops as soon as a non-option argument is encountered. If the first character of `optstring` is `-`, each non-option `argv` element is handled as if it were the argument of an option with character code 1. (This is used by programs that were written to expect options and other `argv` elements in any order and that care about the ordering of the two.) The special argument `--` forces an end of option scanning regardless of the scanning mode.

If `getopt()` does not recognize an option character, it prints an error message to `stderr`, stores the character in `optopt`, and returns `?`. The calling program may prevent the error message by setting `opterr` to `0`.

The `getopt_long()` function works like `getopt()`, except that it also accepts long options, started out by two dashes. Long option names may be abbreviated if the abbreviation is unique or is an exact match for some defined option. A long option may take a parameter, of the form `--arg=param` or `--arg param`.

`longopts` is a pointer to the first element of an array of `struct` option declared in `<getopt.h>`:

```
as struct option {
   const char *name;
   int has_arg;
   int *flag;
   int val;
};
```

The meanings of the different fields are

`name`	The name of the long option.
`has_arg`	`no_argument` (or `0`) if the option does not take an argument, `required_argument` (or `1`) if the option requires an argument, or `optional_argument` (or `2`) if the option takes an optional argument.
`flag`	Specifies how results are returned for a long option. If `flag` is `NULL`, `getopt_long()` returns `val`. (For example, the calling program might set `val` to the equivalent short option character.) Otherwise, `getopt_long()` returns `0`, and `flag` points to a variable that is set to `val` if the option is found, but left unchanged if the option is not found.
`val`	The value to return or to load into the variable pointed to by `flag`.

The last element of the array has to be filled with zeroes.

If `longindex` is not `NULL`, it points to a variable that is set to the index of the long option relative to `longopts`.

`getopt_long_only()` is like `getopt_long()`, but `-` as well as `--` can indicate a long option. If an option that starts with `-` (not `--`) doesn't match a long option but does match a short option, it is parsed as a short option instead.

The members of the protoent structure are

p_name	The official name of the protocol.
p_aliases	A zero-terminated list of alternative names for the protocol.
p_proto	The protocol number.

RETURN VALUE

The getprotoent(), getprotobyname(), and getprotobynumber() functions return the protoent structure, or a NULL pointer if an error occurs or the end of the file is reached.

FILES

/etc/protocols protocol database file

CONFORMS TO

BSD 4.3

SEE ALSO

getservent(3), getnetent(3), protocols(5)

BSD, 24 April 1993

getpw

getpw—Reconstructs password line entry

SYNOPSIS

```
#include <pwd.h>
#include <sys/types.h>
int getpw(uid_t uid, char *buf);
```

DESCRIPTION

The getpw() function reconstructs the password line entry for the given user UID uid in the buffer buf. The returned buffer contains a line of format

```
name:passwd:uid:gid:gecos:dir:shell
```

The passwd structure is defined in <pwd.h> as follows:

```
struct passwd {
        char      *pw_name;     /*username*/
        char      *pw_passwd;   /* user password */
        uid_t     pw_uid;       /* user id */
        gid_t     pw_gid;       /* group id */
        char     *pw_gecos;     /* real name */
        char     *pw_dir;       /* home directory */
        char     *pw_shell;     /* shell program */
};
```

RETURN VALUE

The getpw()function returns 0 on success, or -1 if an error occurs.

ERRORS

ENOMEM	Insufficient memory to allocate passwd structure.

```
void setusershell(void);
void endusershell(void);
```

DESCRIPTION

The `getusershell()` function returns the next line from the file `/etc/shells`, opening the file if necessary. The line should contain the pathname of a valid user shell. If `/etc/shells` does not exist or is unreadable, `getusershell()` behaves as if `/bin/sh` and `/bin/csh` were listed in the file.

The `setusershell()` function rewinds `/etc/shells`.

The `endusershell()` function closes `/etc/shells`.

RETURN VALUE

The `getusershell()` function returns a NULL pointer on end of file.

FILES

`/etc/shells`

CONFORMS TO

BSD 4.3

SEE ALSO

`shells`(5)

getutent, getutid, getutline, pututline, setutent, endutent, utmpname

`getutent`, `getutid`, `getutline`, `pututline`, `setutent`, `endutent`, `utmpname`—Access `utmp` file entries

SYNOPSIS

```
#include <utmp.h>
struct utmp *getutent(void);
struct utmp *getutid(struct utmp *ut);
struct utmp *getutline(struct utmp *ut);
void pututline(struct utmp *ut);
void setutent(void);
void endutent(void);
void utmpname(const char *file);
```

DESCRIPTION

`utmpname()` sets the name of the `utmp`-format file for the other utmp functions to access. If `utmpname()` is not used to set the filename before the other functions are used, they assume `PATH_UTMP`, as defined in `<paths.h>`.

`setutent()` rewinds the file pointer to the beginning of the `utmp` file. It is generally a good idea to call it before any of the other functions.

`endutent()`closes the `utmp` file. It should be called when the user code is done accessing the file with the other functions.

`getutent()` reads a line from the current file position in the `utmp` file. It returns a pointer to a structure containing the fields of the line.

`getutid()`searches forward from the current file position in the `utmp` file based on `ut`. If `ut->ut_type` is `RUN_LVL`, `BOOT_TIME`, `NEW_TIME`, or `OLD_TIME`, `getutid()` will find the first entry whose `ut_type` field matches `ut->ut_type`. If `ut->ut_type` is

glob, globfree

glob, globfree—Find pathnames matching a pattern; free memory from glob()

SYNOPSIS

```
#include <glob.h>
int glob(const char *pattern, int flags,
int errfunc(const char * epath, int eerrno),
glob_t *pglob);
void globfree(glob_t *pglob);
```

DESCRIPTION

The glob() function searches for all the pathnames matching pattern according to the rules used by the shell (see glob(7)). No tilde expansion or parameter substitution is done.

The globfree() function frees the dynamically allocated storage from an earlier call to glob().

The results of a glob() call are stored in the structure pointed to by pglob, which is a glob_t that is declared in <glob.h> as

```
typedef struct
{
    int gl_pathc;        /* Count of paths matched so far */
    char **gl_pathv;     /* List of matched pathnames. */
    int gl_offs;         /* Slots to reserve in 'gl_pathv'. */
    int gl_flags;        /* Flags for globbing. */
} glob_t;
```


Results are stored in dynamically allocated storage.

The parameter flags is made up of bitwise OR of zero or more the following symbolic constants, which modify the of behavior of glob():

GLOB_ERR	Return on read error (because a directory does not have read permission, for example).
GLOB_MARK	Append a slash to each path which corresponds to a directory.
GLOB_NOSORT	Don't sort the returned pathnames (they are by default).
GLOB_DOOFS	pglob->gl_offs slots will be reserved at the beginning of the list of strings in pglob->pathv.
GLOB_NOCHECK	If no pattern matches, return the original pattern.
GLOB_APPEND	Append to the results of a previous call. Do not set this flag on the first invocation of glob().
GLOB_NOESCAPE	Meta characters cannot be quoted by backslashes.
GLOB_PERIOD	A leading period can be matched by meta characters.

If errfunc is not NULL, it will be called in case of an error with the arguments epath, a pointer to the path that failed, and eerrno, the value of errno as returned from one of the calls to opendir(), readdir(), or stat(). If errfunc returns nonzero, or if GLOB_ERR is set, glob() will terminate after the call to errfunc.

Upon successful return, pglob->gl_pathc contains the number of matched pathnames and pglob->gl_pathv a pointer to the list of matched pathnames. The first pointer after the last pathname is NULL.

It is possible to call glob() several times. In that case, the GLOB_APPEND flag has to be set in flags on the second and later invocations.

RETURN VALUES

On successful completion, glob() returns 0. Other possible returns are

GLOB_NOSPACE	For running out of memory,
GLOB_ABEND	For a read error, and
GLOB_NOMATCH	For no found matches.

DESCRIPTION

The `infnan()` function returns a suitable value for infinity and not-a-number (NaN) results. The value of `error` can be `ERANGE` to represent infinity, or anything else to represent NaN. `errno` is also set.

RETURN VALUE

If `error` is `ERANGE` (Infinity), `HUGE_VAL` is returned.

If `error` is `-ERANGE` (-Infinity), `-HUGE_VAL` is returned.

If `error` is anything else, `NaN` is returned.

ERRORS

`ERANGE`	The value of `error` is positive or negative infinity.
`EDOM`	The value of `error` is not-a-number (NaN).

CONFORMS TO

BSD 4.3

GNU, 2 June 1993

initgroups

`initgroup` — Initializes the supplemental group access list

SYNOPSIS

```
#include <grp.h>
#include <sys/types.h>
int initgroups(const char *user, gid_t group);
```

DESCRIPTION

The `initgroups()` function initializes the group access list by reading the group database `/etc/group` and using all groups of which `user` is a member. The additional group `group` is also added to the list.

RETURN VALUE

The `initgroups()` function returns `0` on success, or `-1` if an error occurs.

ERRORS

`EPERM`	The calling process does not have sufficient privileges.
`ENOMEM`	Insufficient memory to allocate group information structure.

FILES

`/etc/group` group database file

CONFORMS TO

SVID 3, BSD 4.3

SEE ALSO

`getgroups`(2), `setgroups`(2)

GNU, 5 April 1993

lgamma

lgamma—Logs gamma function

SYNOPSIS

```
#include <math.h>
double lgamma(double x);
```

DESCRIPTION

The lgamma() function returns the log of the absolute value of the Gamma function. The sign of the Gamma function is returned in the external integer signgam.

For negative integer values of x, lgamma() returns HUGE_VAL, and errno is set to ERANGE.

ERRORS

ERANGE Invalid argument—negative integer value of x.

CONFORMS TO

SVID 3, BSD 4.3

SEE ALSO

infnan(3)

BSD, 25 June 1993

libinn

libinn—InterNetNews library routines

SYNOPSIS

```
#include "libinn.h"
typedef struct _TIMEINFO {
  time_t time;
  long usec;
  long tzone;
} TIMEINFO;

char *GenerateMessageID()

void HeaderCleanFrom(from)
char *from;

char *HeaderFind(Article, Header, size)
char *Article;
char *Header;
int size;

FILE *CAopen(FromServer, ToServer)
FILE *FromServer;
FILE *ToServer;

FILE *CAlistopen(FromServer, ToServer, request)
FILE *FromServer;
FILE *ToServer;
char *request;
```

`CloseOnExec` can make a descriptor *close-on-exec* so that it is not shared with any child processes. If the flag is nonzero, the file is so marked; if it is `0`, the close-on-exec mode is cleared.

`DDstart`, `DDcheck`, and `DDend` are used to set the `Distribution` header; the `DD` stands for Default Distribution. The `distrib.pats`(5) file is consulted to determine the proper value for the `Distribution` header after all newsgroups have been checked. `DDstart` begins the parsing. It returns a pointer to an opaque handle that should be used on subsequent calls. The `FromServer` and `ToServer` parameters should be `FILE`s connected to the NNTP server for input and output, respectively. If either parameter is `NULL`, an empty default will ultimately be returned if the file is not locally available.

`DDcheck` should be called with the handle, `h`, returned by `DDstart` and a new group, `group`, to check. It can be called as often as necessary.

`DDend` releases any state maintained in the handle and returns an allocated copy of the text that should be used for the `Distribution` header.

`SetNonBlocking` enables (if `flag` is nonzero) or disables (if `flag` is `0`) non-blocking I/O on the indicated descriptor. It returns `-1` on failure and `0` on success.

`LockFile` tries to lock the file descriptor `fd`. If `flag` is nonzero it will block until the lock can be made; otherwise it will return `-1` if the file cannot be locked. It returns `-1` on failure and `0` on success.

`GetConfigValue` returns the value of the specified configuration parameter. (See `inn.conf`(5) for details on the parameters and their interpretation.) The returned value points to static space that is reused on subsequent calls.

`GetFileConfigValue` returns the specified configuration parameter from the `inn.conf` file without checking for any defaults. The returned value points to static space that is reused on subsequent calls, or `NULL` if the value is not present.

`GetFQDN` returns the fully qualified domain name of the local host. The returned value points to static space that is reused on subsequent calls, or `NULL` on error.

`GetModeratorAddress` returns the mailing address of the moderator for the specified `group` or `NULL` on error. (See `moderators`(5) for details on how the address is determined.) `GetModeratorAddress` does no checking to see if the specified group is actually moderated. The returned value points to static space that is reused on subsequent calls.

`GetResourceUsage` fills in the `usertime` and `systime` parameters with the total user and system time used by the current process and any children it may have spawned. It gets the values by doing a `times`(2) system call. It returns `-1` on failure, or `0` on success.

`GetTimeInfo` fills in the `now` parameter with information about the current time and `tzone`. The `time` and `usec` fields will be filled in by a call to `gettimeofday`(2). The `time` field will be filled in by a call to `time`(2), and the `usec` field will be set to `0`. The `tzone` field will be filled in with the current offset from GMT. This is done by calling `localtime`(3) and taking the value of the `tm_gmtoff` field, negating it, and dividing it by 60. This is done by calling `localtime`(3) and comparing the value with that returned by a call to `gmtime`(3). For efficiency, the `tzone` field is only recalculated if more than an hour has passed since the last time `GetTimeInfo` was called. This routine returns `-1` on failure, and `0` on success.

`NNTPlocalopen` opens a connection to the private port of an InterNetNews server running on the local host. It returns `-1` on failure, or `0` on success. `FromServerp` and `ToServerp` will be filled in with `FILE`s that can be used to communicate with the server. `errbuff` can either be `NULL` or a pointer to a buffer at least 512 bytes long. If it is not `NULL`, and the server refuses the connection, it will be filled in with the text of the server's reply. This routine is not for general use; it is a subroutine for compatibility with systems that have UNIX-domain stream sockets. It always returns `-1`.

`NNTPremoteopen` does the same as `NNTPlocalopen`, except that it calls `GetConfigValue` to find the name of the local server and opens a connection to the standard NNTP port. Any client program can use this routine. It returns `-1` on failure, or `0` on success.

`NNTPconnect` is the same as `NNTPremoteopen`, except that the desired host is given as the `host` parameter.

`NNTPcheckarticle` verifies that the text meets the NNTP limitations on line length. It returns `-1` on failure, or `0` if the text is valid.

opendir

`opendir`—Opens a directory

SYNOPSIS

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir(const char *name);
```

DESCRIPTION

The `opendir()` function opens a directory stream corresponding to the directory name, and returns a pointer to the directory stream. The stream is positioned at the first entry in the directory.

RETURN VALUE

The `opendir()` function returns a pointer to the directory stream or `NULL` if an error occurred.

ERRORS

`EACESS`	Permission denied
`EMFILE`	Too many file descriptors in use by process
`ENFILE`	Too many files are currently open in the system
`ENOENT`	Directory does not exist, or `name` is an empty string
`ENOMEM`	Insufficient memory to complete the operation
`ENOTDIR`	`name` is not a directory

CONFORMS TO

SVID 3, POSIX, BSD 4.3

SEE ALSO

`open`(2), `readdir`(3), `closedir`(3), `rewinddir`(3), `seekdir`(3), `telldir`(3), `scandir`(3)

11 June 1995

parsedate

`parsedate`—Converts time and date string to number

SYNOPSIS

```
#include <sys/types.h>
typedef struct_TIMEINFO f
time_t time;
long usec;
long tzone;
} TIMEINFO;
time_t
parsedate(text, now)
char *text;
TIMEINFO *now;
```

DESCRIPTION

`parsedate` converts many common time specifications into the number of seconds since the epoch, that is, a `time_t`; see `time`(2).

```
char * QIOread(qp)
QIOSTATE *qp;
int QIOlength(qp)
QIOSTATE *qp;
int QIOtoolong(qp)
QIOSTATE *qp;
int QIOerror(qp)
QIOSTATE *qp;
int QIOtell(qp)
QIOSTATE *qp;
int QIOrewind(qp)
QIOSTATE *qp;
int QIOfileno(qp)
QIOSTATE *qp;
```

DESCRIPTION

The routines described in this manual page are part of the InterNetNews library, `libinn.a`. They are used to provide quick read access to files. The letters `QIO` stand for Quick I/O.

`QIOopen` opens the file `name` for reading. It uses a buffer of size `bytes` which must also be larger than the longest expected line. The header file defines the constant `QIO_BUFFER` as a reasonable default. If size is zero, then `QIOopen` will call `stat`(2) and use the returned block size; if that fails it will use `QIO_BUFFER`. It returns `NULL` on error or a pointer to a handle to be used in other calls. `QIOfdopen` performs the same function except that `fd` refers to an already-open descriptor.

`QIOclose` closes the open file and releases any resources used by it.

`QIOread` returns a pointer to the next line in the file. The trailing newline will be replaced with a `\0`. If `EOF` is reached, an error occurs, or if the line is longer than the buffer, `QIOread` returns `NULL`.

After a successful call to `QIOread`, `QIOlength` will return the length of the current line.

The functions `QIOtoolong` and `QIOerror` can be called after `QIOread` returns `NULL` to determine if there was an error, or if the line was too long. If `QIOtoolong` returns non-zero, then the current line did not fit in the buffer, and the next call to `QIOread` will try read the rest of the line. Long lines can only be discarded. If `QIOerror` returns non-zero, then a serious I/O error occurred.

`QIOtell` returns the `lseek`(2) offset at which the next line will start.

`QIOrewind` sets the read pointer back to the beginning of the file.

`QIOfileno` returns the descriptor of the open file.

`QIOlength`, `QIOtoolong`, `QIOerror`, `QIOtell`, and `QIOfileno` are implemented as macros defined in the header file.

EXAMPLE

```
QIOSTATE *h;
long offset;
char *p;
h = QIOopen("/etc/motd", QIO_BUFFER);
for (offset = QIOtell(h); (p = QIOread(h)) != NULL; offset = QIOtell(h))
printf("At %ld, %s\n", offset, p);
if (QIOerror(h)) {
   perror("Read error");
   exit(1);
   }
QIOclose(h);
```

HISTORY

Written by Rich $alz (`rsalz@uunet.uu.net`) for InterNetNews.

qsort

`qsort`—Sorts an array

SYNOPSIS

```
#include <stdlib.h>
void qsort(void *base, size_t nmemb, size_t size,int(*compar)
(const void *, const void *));
```

DESCRIPTION

The `qsort()` function sorts an array with `nmemb` elements of size `size`.The base argument points to the start of the array.

The contents of the array are sorted in ascending order according to a comparison function pointed to by `compar`, which is called with two arguments that point to the objects being compared.

The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second. If two members compare as equal, their order in the sorted array is undefined.

RETURN VALUE

The `qsort()` function returns no value.

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO

`sort(1)`

GNU, 29 March 1993

raise

`raise`—Sends a signal to the current process

SYNOPSIS

```
#include <signal.h>
int raise (int sig);
```

DESCRIPTION

The `raise` function sends a signal to the current process. It is equivalent to

```
kill(getpid(),sig)
```

RETURN VALUE

Zero on success, non-zero for failure.

CONFORMS TO

ANSI C

SEE ALSO

`kill`(2), `signal`(2), `getpid`(2)

GNU, 31 August 1995

rand, srand

`rand`, `srand`—Random number generator

SYNOPSIS

```
#include <stdlib.h>
int rand(void);
void srand(unsigned int seed);
```

DESCRIPTION

The `rand()` function returns a pseudo-random integer between `0` and `RAND_MAX`.

The `srand()` function sets its argument as the seed for a new sequence of pseudo-random integers to be returned by `rand()`. These sequences are repeatable by calling `srand()` with the same `seed` value.

If no `seed` value is provided, the `rand()` function is automatically seeded with a value of `1`.

RETURN VALUE

The `rand()` function returns a value between `0` and `RAND_MAX`. The `srand()` returns no value.

NOTES

The versions of `rand()` and `srand()` in the Linux C Library use the same random number generator as `random()` and `srandom()`, so the lower-order bits should be as random as the higher-order bits. However, on older `rand()` implementations, the lower-order bits are much less random than the higher-order bits.

In *Numerical Recipes in C: The Art of Scientific Computing* (William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling; New York: Cambridge University Press, 1990, first ed, p. 207), the following comments are made:

"If you want to generate a random integer between 1 and 10, you should always do it by

```
j=1+(int) (10.0*rand()/(RAND+MAX+1.0));
```

and never by anything resembling

```
j=1+((int) (1000000.0*rand()) % 10);
```

(which uses lower-order bits)."

Random-number generation is a complex topic. The `Numerical Recipes in C` book (see preceding reference) provides an excellent discussion of practical random-number generation issues in Chapter 7, "Random Numbers."

For a more theoretical discussion that also covers many practical issues in depth, please see Chapter 3, "Random Numbers," in Donald E. Knuth's *The Art of Computer Programming, Volume 2* (Seminumerical Algorithms), 2nd ed.; Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

CONFORMS TO

SVID 3, BSD 4.3, ISO 9899

SEE ALSO

`random`(3), `srandom`(3), `initstate`(3), `setstate`(3)

GNU, 18 May 1995

random, srandom, initstate, setstate

`random`, `srandom`, `initstate`, `setstate`—Random number generator

cflags may be the bitwise or of one or more of the following:

REG_EXTENDED	Use POSIX extended regular expression syntax when interpreting regex. If not set, POSIX basic regular expression syntax is used.
REG_ICASE	Do not differentiate case. Subsequent regexec searches using this pattern buffer will be case-insensitive.
REG_NOSUB	Support for substring addressing of matches is not required. The nmatch and pmatch parameters to regexec are ignored if the pattern buffer supplied was compiled with this flag set.
REG_NEWLINE	Match-any-character operators don't match a newline. A nonmatching list ([^...]) not containing a newline matches a newline. Match-beginning-of-line operator (^) matches the empty string immediately after a newline, regardless of whether eflags, the execution flags of regexec, contains REG_NOTBOL. Match-end-of-line operator ($) matches the empty string immediately before a newline, regardless of whether eflags contains REG_NOTEOL.

POSIX REGEX MATCHING

regexec is used to match a null-terminated string against the precompiled pattern buffer, preg. nmatch and pmatch are used to provide information regarding the location of any matches. eflags may be the bitwise or of one or both of REG_NOTBOL and REG_NOTEOL, which cause changes in matching behavior described in the following list:

REG_NOTBOL	The match-beginning-of-line operator always fails to match (but see the compilation flag REG_NEWLINE, in the preceding subsection). This flag may be used when different portions of a string are passed to regexec and the beginning of the string should not be interpreted as the beginning of the line.
REG_NOTEOL	The match-end-of-line operator always fails to match (but see the compilation flag REG_NEWLINE, in the preceding subsection).

BYTE OFFSETS

Unless REG_NOSUB was set for the compilation of the pattern buffer, it is possible to obtain substring match addressing information. pmatch must be dimensioned to have at least nmatch elements. These are filled in by regexec with substring match addresses. Any unused structure elements will contain the value -1.

The regmatch_t structure that is the type of pmatch is defined in regex.h:

```
typedef struct
{
  regoff_t rm_so;
  regoff_t rm_eo;
} regmatch_t;
```

Each rm_so element that is not -1 indicates the start offset of the next largest substring match within the string. The relative rm_eo element indicates the end offset of the match.

POSIX ERROR REPORTING

regerror is used to turn the error codes that can be returned by both regcomp and regexec into error message strings.

regerror is passed the error code, errcode; the pattern buffer, preg; a pointer to a character string buffer, errbuf; and the size of the string buffer, errbuf_size. It returns the size of the errbuf required to contain the null-terminated error message string. If both errbuf and errbuf_size are non-zero, errbuf is filled in with the first errbuf_size - 1 characters of the error message and a terminating null.

SYNOPSIS

```
#include <signal.h>
int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);
int sigaddset(sigset_t *set,int signum);
int sigdelset(sigset_t *set,int signum);
int sigismember(const sigset_t *set,int signum);
```

DESCRIPTION

The sigsetops(3) functions allow the manipulation of POSIX signal sets.

sigemptyset initializes the signal set given by set to empty, with all signals excluded from the set.

sigfillset initializes set to full, including all signals.

sigaddset and sigdelset add and delete, respectively, signal signum from set.

sigismember tests whether signum is a member of set.

RETURN VALUES

sigemptyset, sigfullset, sigaddset, and sigdelset return 0 on success and -1 on error.

sigismember returns 1 if signum is a member of set, 0 if signum is not a member, and -1 on error.

ERRORS

EINVAL signum is not a valid signal.

CONFORMS TO

POSIX

SEE ALSO

sigaction(2), sigpending(2), sigprocmask(2), sigsuspend(2)

Linux 1.0, 24 September 1994

sin

sin—Sine function

SYNOPSIS

```
#include <math.h>
double sin(double x);
```

DESCRIPTION

The sin() function returns the sine of x, where x is given in radians.

RETURN VALUE

The sin() function returns a value between -1 and 1.

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

The va_arg macro expands to an expression that has the type and value of the next argument in the call. The parameter ap is the va_list ap initialized by va_start. Each call to va_arg modifies ap so that the next call returns the next argument. The parameter type is a type name specified so that the type of a pointer to an object that has the specified type can be obtained simply by adding a * to type.

If there is no next argument, or if type is not compatible with the type of the actual next argument (as promoted according to the default argument promotions), random errors will occur.

The first use of the va_arg macro after that of the va_start macro returns the argument after last. Successive invocations return the values of the remaining arguments.

The va_end macro handles a normal return from the function whose variable argument list was initialized by va_start.

The va_end macro returns no value.

EXAMPLE

The function foo takes a string of format characters and prints out the argument associated with each format character based on the type.

```
void foo(char *fmt, ...)
{
     va_list ap;
     int d;
     char c, *p, *s;

     va_start(ap, fmt);
     while (*fmt)
          switch(*fmt++) {
          case 's':              /* string */
               s = va_arg(ap, char *);
               printf("string %s\n", s);
               break;
          case 'd':              /* int */
               d = va_arg(ap, int);
               printf("int %d\n", d);
               break;
          case 'c':              /* char */
               c = va_arg(ap, char);
               printf("char %c\n", c);
               break;
          }
     va_end(ap);
}
```


STANDARDS

The va_start, va_arg, and va_end macros conform to ANSI C3.159-1989 (ANSI C).

COMPATIBILITY

These macros are not compatible with the historic macros they replace. A backwards-compatible version can be found in the include file varargs.h.

BUGS

Unlike the varargs macros, the stdarg macros do not permit programmers to code a function with no fixed arguments. This problem generates work mainly when converting varargs code to stdarg code, but it also creates difficulties for variadic functions that wish to pass all of their arguments on to a function that takes a va_list argument, such as vfprintf(3).

stdio

`stdio`—Standard input/output library functions

SYNOPSIS

```
#include <stdio.h>
FILE *stdin;
FILE *stdout;
FILE *stderr;
```

DESCRIPTION

The standard I/O library provides a simple and efficient buffered stream I/O interface. Input and output is mapped into logical data streams and the physical I/O characteristics are concealed. The functions and macros are listed in this section; more information is available from the individual man pages.

A stream is associated with an external file (which may be a physical device) by opening a file, which may involve creating a new file. Creating an existing file causes its former contents to be discarded. If the file can support positioning requests (such as a disk file, as opposed to a terminal), then a file position indicator is associated with the stream is positioned at the start of the file (byte zero), unless the file is opened with append mode. If append mode is used, the position indicator will be placed the end-of-file. The position indicator is maintained by subsequent reads, writes, and positioning requests. All input occurs as if the characters were read by successive calls to the `fgetc`(3) function; all output takes place as if all characters were read by successive calls to the `fput`(3) function.

A file is disassociated from a stream by closing the file. Output streams are flushed (any unwritten buffer contents are transferred to the host environment) before the stream is disassociated from the file. The value of a pointer to a `FILE` object is indeterminate after a file is closed (garbage).

A file may be subsequently reopened, by the same or another program execution, and its contents reclaimed or modified (if it can be repositioned at the start). If the main function returns to its original caller, or the `exit`(3) function is called, all open files are closed (hence all output streams are flushed) before program termination. Other methods of program termination, such as `abort`(3) do not bother about closing files properly.

At program startup, three text streams are predefined and need not be opened explicitly: standard input (for reading conventional input), standard output (for writing conventional input), and standard error (for writing diagnostic output). These streams are abbreviated `stdin`, `stdout`, and `stderr`. When opened, the standard error stream is not fully buffered; the standard input and output streams are fully buffered if and only if the streams do not to refer to an interactive device.

Output streams that refer to terminal devices are always line buffered by default; pending output to such streams is written automatically whenever an input stream that refers to a terminal device is read. In cases where a large amount of computation is done after printing part of a line on an output terminal, it is necessary to `fflush`(3) the standard output before going off and computing so that the output will appear.

The `stdio` library is a part of the library `libc` and routines are automatically loaded as needed by the compilers `cc`(1) and `pc`(1). The `SYNOPSIS` sections of the following manual pages indicate which include files are to be used, what the compiler declaration for the function looks like, and which external variables are of interest.

The following are defined as macros; these names may not be reused without first removing their current definitions with `#undef`: `BUFSIZ`, `EOF`, `FILENAME_MAX`, `FOPEN_MAX`, `L_cuserid`, `L_ctermid`, `L_tmpnam`, `NULL`, `SEEK_END`, `SEEK_SET`, `SEE_CUR`, `TMP_MAX`, `clearerr`, `feof`, `ferror`, `fileno`, `fropen`, `fwopen`, `getc`, `getchar`, `putc`, `putchar`, `stderr`, `stdin`, `stdout`. Function versions of the macro functions `feof`, `ferror`, `clearerr`, `fileno`, `getc`, `getchar`, `putc`, and `putchar` exist and will be used if the macros definitions are explicitly removed.

SEE ALSO

`open`(2), `close`(2), `read`(2), `write`(2)

BUGS

The standard buffered functions do not interact well with certain other library and system functions, especially `vfork` and `abort`. This may not be the case under Linux.

STANDARDS

The `stdio` library conforms to ANSI C3.159-1989 (ANSI C).

LIST OF FUNCTIONS

Function	*Description*
`clearerr`	Check and reset stream status
`fclose`	Close a stream
`fdopen`	Stream open functions
`feof`	Check and reset stream status
`ferror`	Check and reset stream status
`fflush`	Flush a stream
`fgetc`	Get next character or word from input stream
`fgetline`	Get a line from a stream
`fgetpos`	Reposition a stream
`fgets`	Get a line from a stream
`fileno`	Check and reset stream status
`fopen`	Stream open functions
`fprintf`	Formatted output conversion
`fpurge`	Flush a stream
`fputc`	Output a character or word to a stream
`fputs`	Output a line to a stream
`fread`	Binary stream input/output
`freopen`	Stream open functions
`fropen`	Open a stream
`fscanf`	Input format conversion
`fseek`	Reposition a stream
`fsetpos`	Reposition a stream
`ftell`	Reposition a stream
`fwrite`	Binary stream input/output
`getc`	Get next character or word from input stream
`getchar`	Get next character or word from input stream
`gets`	Get a line from a stream
`getw`	Get next character or word from input stream
`mktemp`	Make temporary filename (unique)
`perror`	System error messages
`printf`	Formatted output conversion
`putc`	Output a character or word to a stream
`putchar`	Output a character or word to a stream
`puts`	Output a line to a stream
`putw`	Output a character or word to a stream

The members of the tm structure are

tm_sec	The number of seconds after the minute, normally in the range 0 to 59, but can be up to 61 to allow for leap seconds.
tm_min	The number of minutes after the hour, in the range 0 to 59.
tm_hour	The number of hours past midnight, in the range 0 to 23.
tm_mday	The day of the month, in the range 1 to 31.
tm_mon	The number of months since January, in the range 0 to 11.
tm_year	The number of years since 1900.
tm_wday	The number of days since Sunday, in the range 0 to 6.
tm_yday	The number of days since January 1, in the range 0 to 365.
tm_isdst	A flag that indicates whether daylight saving time is in effect at the time described. The value is positive if daylight saving time is in effect, zero if it is not, and negative if the information is not available.

RETURN VALUE

The strftime() function returns the number of characters placed in the array s, not including the terminating NULL character. If the value equals max, it means that the array was too small.

CONFORMS TO

SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO

date(1), time(2), ctime(3), setlocale(3), sprintf(3)

NOTES

The function supports only those locales specified in locale(7)

GNU, 2 July 1993

strcasecmp, strcat, strchr, strcmp, strcoll, strcpy, strcspn, strdup, strfry, strlen, strncat, strncmp, strncpy, strncasecmp, strpbrk, strrchr, strsep, strspn, strstr, strtok, strxfrm, index, rindex

strcasecmp, strcat, strchr, strcmp, strcoll, strcpy, strcspn, strdup, strfry, strlen, strncat, strncmp, strncpy, strncasecmp, strpbrk, strrchr, strsep, strspn, strstr, strtok, strxfrm, index, rindex—String operations

SYNOPSIS

```
#include <string.h>
int strcasecmp(const char *s1, const char *s2);
char *strcat(char *dest, const char *src);
char *strchr(const char *s,int c);
int strcmp(const char *s1, const char *s2);
int strcoll(const char *s1, const char *s2);
char *strcpy(char *dest, const char *src);
size_t strcspn(const char *s, const char *reject);
char *strdup(const char *s);
```

DESCRIPTION

The `strsep()` function returns the next token from the string `stringp` which is delimited by `delim`. The token is terminated with a `\0` character and `stringp` is updated to point past the token.

RETURN VALUE

The `strsep()` function returns a pointer to the token, or `NULL` if `delim` is not found in `stringp`.

CONFORMS TO

BSD 4.3

SEE ALSO

`index`(3), `memchr`(3), `rindex`(3), `strchr`(3), `strpbrk`(3), `strspn`(3), `strstr`(3), `strtok`(3)

GNU, 13 April 1993

strsignal

`strsignal`—Returns string describing signal

SYNOPSIS

```
#include <string.h>
char *strsignal(int sig);
extern const char * const sys_siglist[];
```

DESCRIPTION

The `strsignal()` function returns a string describing the signal number passed in the argument `sig`. The string can only be used until the next call to `strsignal()`.

The array `sys_siglist` holds the signal description strings indexed by signal number.

RETURN VALUE

The `strsignal()` function returns the appropriate description string, or an unknown signal message if the signal number is invalid.

SEE ALSO

`psignal`(3), `strerror`(3)

GNU, 13 April 1993

strspn, strcspn

`strspn`, `strcspn`—Search a string for a set of characters

SYNOPSIS

```
#include <string.h>
size t strspn(const char *s, const char *accept);
size t strcspn(const char *s, const char *reject);
```

DESCRIPTION

The `strspn()` function calculates the length of the initial segment of `s`, which consists entirely of characters in `accept`.

The `strcspn()` function calculates the length of the initial segment of `s`, which consists entirely of characters not in `reject`.

toascii

`toascii`—Converts character to ASCII

SYNOPSIS

```
#include <ctype.h>
int toascii (int c);
```

DESCRIPTION

`toascii()` converts `c` to a 7-bit `unsigned char` value that fits into the ASCII character set, by clearing the high-order bits.

RETURN VALUE

The value returned is that of the converted character.

CONFORMS TO

SVID, BSD

BUGS

Many people will be unhappy if you use this function. This function will convert accented letters into random characters.

SEE ALSO

`isascii`(3), `toupper`(3), `tolower`(3)

GNU, 16 September 1995

toupper, tolower

`toupper`, `tolower`—Convert letter to uppercase or lowercase

SYNOPSIS

```
#include <ctype.h>
int toupper (int c);
int tolower (int c);
```

DESCRIPTION

`toupper()` converts the letter `c` to uppercase, if possible.

`tolower()` converts the letter `c` to lowercase, if possible.

RETURN VALUE

The value returned is that of the converted letter, or `c` if the conversion was not possible.

CONFORMS TO

ANSI C, BSD 4.3

BUGS

The details of what constitutes an uppercase or lowercase letter depend on the current locale. For example, the default locale does not know about umlauts, so no conversion is done for them.

In some non-English locales, there are lowercase letters with no corresponding uppercase equivalent; the German sharp `s` is one example.

Common ways to switch consoles are the following:

- Use Alt+Fn or Ctrl+Alt+Fn to switch to console n; AltGr+Fn might bring you to console n+12 [here Alt and AltGr refer to the left and right Alt keys, respectively]
- Use Alt+RightArrow or Alt+LeftArrow to cycle through the presently allocated consoles
- Use the program `chvt`(1). (The key mapping can be set by the user; see `loadkeys`(1); the preceding key combinations are according to the default settings.)

The command `disalloc`(8) will free the memory taken by the screen buffers for consoles that no longer have any associated process.

PROPERTIES

Consoles carry a lot of state. I hope to document that some other time. The most important fact is that the consoles simulate vt100 terminals. In particular, a console is reset to the initial state by printing the two characters ESC c. All escape sequences can be found in console `codes`(4).

FILES

```
/dev/console
/dev/tty*
```

SEE ALSO

`charsets`(4), `console_codes`(4), `console_ioctl`(4), `mknod`(1), `tty`(4), `ttys`(4), `getty`(8), `init`(8), `chvt`(1), `open`(1), `disalloc`(8), `loadkeys`(1), `resizecons`(8), `setfont`(8), `mapscrn`(8)

Linux, 31 October 1994

console_codes

`console_codes`—Linux console escape and control sequences

DESCRIPTION

The Linux console implements a large subset of the VT102 and ECMA-48/ISO 6429/ANSI X3.64 terminal controls, plus certain private-mode sequences for changing the color palette, character-set mapping, and so on. In the following tabular descriptions, the second column gives ECMA-48 or DEC mnemonics (the latter if prefixed with DEC) for the given function. Sequences without a mnemonic are neither ECMA-48 nor VT102.

After all the normal output processing has been done, and a stream of characters arrives at the console driver for actual printing, the first thing that happens is a translation from the code used for processing to the code used for printing.

If the console is in UTF-8 mode, then the incoming bytes are first assembled into 16-bit Unicode codes. Otherwise, each byte is transformed according to the current mapping table (which translates it to a Unicode value). (See the “Character Sets” subsection for discussion.)

In the normal case, the Unicode value is converted to a font index, and this is stored in video memory, so that the corresponding glyph (as found in video ROM) appears on the screen. Note that the use of Unicode (and the design of the PC hardware) allows the use of 512 different glyphs simultaneously.

If the current Unicode value is a control character, or you are currently processing an escape sequence, the value will treated specially. Instead of being turned into a font index and rendered as a glyph, it may trigger cursor movement or other control functions. (See the “Linux Console Controls” subsection.)

It is generally not good practice to hardwire terminal controls into programs. Linux supports a `terminfo`(5) database of terminal capabilities. Rather than emitting console escape sequences by hand, you will almost always want to use a `terminfo`-aware screen library or utility such as `ncurses`(3), `tput`(1), or `reset`(1).

`ESC ( U`		Select null mapping—straight to character ROM.
`ESC ( K`		Select user mapping, the map that is loaded by the utility `mapscrn`(8).
`ESC )`		Start sequence defining G1 (followed by one of `B`, `0`, `U`, `K`, as above).
`ESC >`	`DECPNM`	Set numeric keypad mode.
`ESC =`	`DECPAM`	Set application keypad mode.
`ESC ]`	`OSC`	(Should be: Operating system command) `ESC ] P nrrggbb`: set palette, with parameter given in 7 hexadecimal digits after the final `P` `:-(`. Here `n` is the color (0–16), and `rrggbb` indicates the red/green/blue values (0–255). `ESC ] R`: reset palette.

ECMA-48 CSI SEQUENCES

`CSI` (or `ESC [`) is followed by a sequence of parameters, at most `NPAR`(16), that are decimal numbers separated by semicolons. An empty or absent parameter is taken to be `0`. The sequence of parameters may be preceded by a single question mark.

However, after `CSI [` (or `ESC [ [`) a single character is read and this entire sequence is ignored. (The idea is to ignore an echoed function key.)

The action of a `CSI` sequence is determined by its final character.

Character	*Function*	*Description*
`@`	`ICH`	Insert the indicated # of blank characters.
`A`	`CUU`	Move cursor up the indicated # of rows.
`B`	`CUD`	Move cursor down the indicated # of rows.
`C`	`CUF`	Move cursor right the indicated # of columns.
`D`	`CUB`	Move cursor left the indicated # of columns.
`E`	`CNL`	Move cursor down the indicated # of rows, to column 1.
`F`	`CPL`	Move cursor up the indicated # of rows, to column 1.
`G`	`CHA`	Move cursor to indicated column in current row.
`H`	`CUP`	Move cursor to the indicated row, column (origin at 1,1).
`J`	`ED`	Erase display (default: from cursor to end of display).
		`ESC [ 1 J`: erase from start to cursor.
		`ESC [ 2 J`: erase whole display.
`K`	`EL`	Erase line (default: from cursor to end of line).
		`ESC [ 1 K`: erase from start of line to cursor.
		`ESC [ 2 K`: erase whole line.
`L`	`IL`	Insert the indicated # of blank lines.
`M`	`DL`	Delete the indicated # of lines.
`P`	`DCH`	Delete the indicated # of characters on the current line.
`X`	`ECH`	Erase the indicated # of characters on the current line.
`a`	`HPR`	Move cursor right the indicated # of columns.
`c`	`DA`	Answer `ESC [ ? 6 c: 'I am a VT102'.`
`d`	`VPA`	Move cursor to the indicated row, current column.
`e`	`VPR`	Move cursor down the indicated # of rows.
`f`	`HVP`	Move cursor to the indicated row, column.

continues

`KDGETLED` Get state of LEDs. `argp` points to a long `int`. The lower three bits of `*argp` are set to the state of the LEDs, as follows:

`LED_CAP`	`0x04`	caps lock LED
`LEC_NUM`	`0x02`	num lock LED
`LED_SCR`	`0x01`	scroll lock LED

`KDSETLED` Set the LEDs. The LEDs are set to correspond to the lower three bits of `argp`. However, if a higher order bit is set, the LEDs revert to normal, displaying the state of the keyboard functions of caps lock, num lock, and scroll lock.

Before 1.1.54, the LEDs just reflected the state of the corresponding keyboard flags, and `KDGETLED`/`KDSETLED` would also change the keyboard flags. Since 1.1.54 the LEDs can be made to display arbitrary information, but by default they display the keyboard flags. The following two ioctls are used to access the keyboard flags.

`KDGKBLED` Get keyboard flags `CapsLock`, `NumLock`, `ScrollLock` (not lights). `argp` points to a `char` that is set to the flag state. The low order three bits (mask `0x7`) get the current flag state, and the low order bits of the next nibble (mask `0x70`) get the default flag state (since 1.1.54).

`KDSKBLED` Set keyboard flags `CapsLock`, `NumLock`, `ScrollLock` (not lights). `argp` has the desired flag state. The low order three bits (mask `0x7`) have the flag state, and the low order bits of the next nibble (mask `0x70`) have the default flag state (since 1.1.54).

`KDGKBTYPE` Get keyboard type. This returns the value `KB 101`, defined as `0x02`.

`KDADDIO` Add I/O port as valid. Equivalent to `ioperm(arg,1,1)`.

`KDDELIO` Delete I/O port as valid. Equivalent to `ioperm(arg,1,0)`.

`KDENABIO` Enable I/O to video board. Equivalent to `ioperm(0x3b4, 0x3df-0x3b4+1, 1)`.

`KDDISABIO` Disable I/O to video board. Equivalent to `ioperm(0x3b4, 0x3df-0x3b4+ 1, 0)`.

`KDSETMODE` Set text/graphics mode. `argp` is one of these:

`KD_TEXT`	`0x00`
`KD_GRAPHICS`	`0x01`

`KDGETMODE` Get text/graphics mode. `argp` points to a long which is set to one of the above values.

`KDMKTONE` Generate tone of specified length. The lower 16 bits of `argp` specify the period in clock cycles, and the upper 16 bits give the duration in `msec`. If the duration is zero, the sound is turned off. Control returns immediately. For example, `argp = (125<<16) + 0x637` would specify the beep normally associated with a `ctrl-G`.

`KIOCSOUND` Start or stop sound generation. The lower 16 bits of `argp` specify the period in clock cycles (that is, argp = 1193180/frequency). `argp = 0` turns sound off. In either case, control returns immediately.

`GIO_CMAP` Get the current default color map from kernel. `argp` points to a 48-byte array. (Since 1.3.3.)

`PIO_CMAP` Change the default text-mode color map. `argp` points to a 48-byte array that contains, in order, the red, green, and blue values for the 16 available screen colors: `0` is off, and `255` is full intensity. The default colors are, in order: black, dark red, dark green, brown, dark blue, dark purple, dark cyan, light grey, dark grey, bright red, bright green, yellow, bright blue, bright purple, bright cyan, and white. (Since 1.3.3.)

`GIO_FONT` Gets 256-character screen font in expanded form. `argp` points to an 8192-byte array. Fails with error code `EINVAL` if the currently loaded font is a 512-character font, or if the console is not in text mode.

ispell

ispell—Format of ispell dictionaries and affix files

DESCRIPTION

ispell(1) requires two files to define the language that it is spell checking. The first file is a dictionary containing words for the language, and the second is an affix file that defines he meaning of special flags in the dictionary. The two files are combined by buildhash (see spell(1)) and written to a hash file that is not described here.

A raw ispell dictionary (either the main dictionary or your own personal dictionary) contains a list of words, one per line. Each word may optionally be followed by a slash (/) and one or more flags, which modify the root word as explained later. Depending on the options with which ispell was built, case may or may not be significant in either the root word or the flags, independently. Specifically, if the compile-time option CAPITALIZATION is defined, case is significant in the root word; if not, case is ignored in the root word. If the compile-time option MASKBITS is set to a value of 32, case is ignored in the flags; otherwise, case is significant in the flags. Contact your system administrator or ispell maintainer for more information (or use the -vv flag to find out). The dictionary should be sorted with the -f flag of sort(1) before the hash file is built; this is done automatically by unchlist(1), which is the normal way of building dictionaries.

If the dictionary contains words that have string characters (see the affix file documentation, following), they must be written in the format given by the defstringtype statement in the affix file. This will be the case for most non-English languages. Be careful to use this format, rather than that of your favorite formatter, when adding words to a dictionary. If you add words to your personal dictionary during an ispell session, they will automatically be converted to the correct format. This feature can be used to convert an entire dictionary if necessary:

```
echo qqqqq > dummy.dict
buildhash dummy.dict affix-file dummy.hash
awk 'fprint "*"gENDfprint "#"g' old-dict-file \
¦ ispell -a -T old-dict-string-type \
-d ./dummy.hash -p ./new-dict-file \
> /dev/null
rm dummy.*
```

The case of the root word controls the case of words accepted by ispell, as follows:

1. If the root word appears only in lowercase (for example, bob), it will be accepted in lowercase, capitalized, or all capitals.
2. If the root word appears capitalized (for example, Robert), it will not be accepted in all lowercase, but will be accepted capitalized or all in capitals.
3. If the root word appears all in capitals (for example, UNIX), it will only be accepted all in capitals.
4. If the root word appears with a “funny” capitalization (for example, ITCorp), a word will be accepted only if it follows that capitalization, or if it appears all in capitals.
5. More than one capitalization of a root word may appear in the dictionary. Flags from different capitalizations are combined using OR.

Redundant capitalizations (for example, bob and Bob) will be combined by buildhash and by ispell (for personal dictionaries), and can be removed from a raw dictionary by munchlist.

For example, the dictionary

```
bob
Robert
UNIX
ITcorp
ITCorp
```

will accept bob, Bob, BOB, Robert, ROBERT, UNIX, ITcorp, ITCorp, and ITCORP, and will reject all others. Some of the unacceptable forms are bOb, robert, Unix, and ItCorp.

As mentioned, root words in any dictionary may be extended by flags. Each flag is a single alphabetic character, which represents a prefix or suffix that may be added to the root to form a new word. For example, in an English dictionary the `D` flag can be added to bathe to make bathed. Because flags are represented as a single bit in the hashed dictionary, this results in significant space savings. The `munchlist` script will reduce an existing raw dictionary by adding flags when possible.

When a word is extended with an affix, the affix will be accepted only if it appears in the same case as the initial (prefix) or final (suffix) letter of the word. Thus, for example, the entry `UNIX/M` in the main dictionary (M means add an apostrophe and an `s` to make a possessive) would accept UNIX'S but would reject UNIX's. If UNIX's is legal, it must appear as a separate dictionary entry, and it will not be combined by `munchlist`. (In general, you don't need to worry about these things; `munchlist` guarantees that its output dictionary will accept the same set of words as its input, so all you have to do is add words to the dictionary and occasionally run `munchlist` to reduce its size.)

As mentioned, the affix definition file describes the affixes associated with particular flags. It also describes the character sets used by the language.

Although the affix-definition grammar is designed for a line-oriented layout, it is actually a free-format grammar and can be laid out weirdly if you want. Comments are started by a pound (sharp) sign (`#`), and continue to the end of the line. Backslashes are supported in the usual fashion (`\nnn`, plus specials `\n`, `\r`, `\t`, `\v`, `\f`, and `\a`, and the new hex format `\xnn`). Any character with special meaning to the parser can be changed to an uninterpreted token by backslashing it; for example, you can declare a flag named `asterisk` or `colon` with flag `*:` `\*flag\n::`.

The grammar will be presented in a top-down fashion, with discussion of each element. An affix-definition file must contain exactly one table:

```
table :[headers][prefixes][suffixes]
```

At least one of prefixes and suffixes is required. They can appear in either order.

```
headers :[options ] char-sets
```

The headers describe options global to this dictionary and language. These include the character sets to be used and the formatter, and the defaults for certain *ispell* flags.

```
options : { fmtr-stmt ¦ opt-stmt ¦ flag-stmt ¦ num-stmt }
```

The options statements define the defaults for certain `ispell` flags and for the character sets used by the formatters.

```
fmtr-stmt : { nroff-stmt ¦ tex-stmt }
```

A `fmtr-stmt` statement describes characters that have special meaning to a formatter. Normally, this statement is not necessary, but some languages may have preempted the usual defaults for use as language-specific characters. In this case, these statements may be used to redefine the special characters expected by the formatter.

```
nroff-stmt : { nroffchars ¦ troffchars } string
```

The `nroffchars` statement allows redefinition of certain `nroff` control characters. The string given must be exactly five characters long, and must list substitutions for the left and right parentheses, the period, the backslash, and the asterisk. (The right parenthesis is not currently used, but is included for completeness.) For example, the statement:

```
nroffchars {}.\\*
```

would replace the left and right parentheses with left and right curly braces for purposes of parsing `nroff`/`troff` strings, with no effect on the others (admittedly a contrived example). Note that the backslash is escaped with a backslash.

```
tex-stmt : { TeXchars ¦ texchars } string
```

The `TeXchars` statement allows redefinition of certain TeX/LaTeX control characters. The string given must be exactly thirteen characters long, and must list substitutions for the left and right parentheses, the left and right square brackets, the left and right curly braces, the left and right angle brackets, the backslash, the dollar sign, the asterisk, the period or dot, and the percent sign. For example, the statement:

```
texchars ()\[]<\><\>\\$*.%
```

A single character-set statement can declare either a single character or a contiguous range of characters. A range is given as in `egrep` and the shell: `[a-z]` means lowercase alphabetics; `[^a-z]` means all but lowercase, and so on. All character-set statements are combined (unioned) to produce the final list of characters that may be part of a word. The collating order of the characters is defined by the order of their declaration; if a range is used, the characters are considered to have been declared in ASCII order. Characters that have case are collated next to each other, with the uppercase character first.

The character-declaration statements have a rather strange behavior caused by the need to match each lowercase character with its uppercase equivalent. In any given `wordchars` or `boundarychars` statement, the characters in each range are first sorted into ASCII collating sequence, then matched one-for-one with the other range. (The two ranges must have the same number of characters). Thus, for example, the two statements:

```
wordchars [aeiou] [AEIOU]
wordchars [aeiou] [UOIEA]
```

would produce exactly the same effect. To get the vowels to match up "wrong," you would have to use separate statements:

```
wordchars a U
wordchars e O
wordchars i I
wordchars o E
wordchars u A
```

which would cause uppercase *e* to be *O*, and lowercase *e* to be *e*. This should normally be a problem only with languages that have been forced to use a strange ASCII collating sequence. If your uppercase and lowercase letters both collate in the same order, you shouldn't have to worry about this "feature."

The prefixes and suffixes sections have exactly the same syntax, except for the introductory keyword:

```
prefixes : prefixes flagdef*
suffixes : suffixes flagdef*
flagdef : flag [*jÚ] char : repl *
```

A prefix or suffix table consists of an introductory keyword and a list of flag definitions. Flags can be defined more than once, in which case the definitions are combined. Each flag controls one or more `repls` (replacements), which are conditionally applied to the beginnings or endings of various words.

Flags are named by a single character `char`. Depending on a configuration option, this character can be either any uppercase letter (the default configuration) or any 7-bit ASCII character. Most languages should be able to get along with just 26 flags.

A flag character may be prefixed with one or more option characters. (If you wish to use one of the option characters as a flag character, simply enclose it in double quotes.)

The asterisk (*) option means that this flag participates in cross-product formation. This only matters if the file contains both prefix and suffix tables. If so, all prefixes and suffixes marked with an asterisk will be applied in all cross-combinations to the root word. For example, consider the root *fix* with prefixes *pre* and *in*, and suffixes *es* and *ed*. If all flags controlling these prefixes and suffixes are marked with an asterisk, then the single root *fix* would also generate *prefix*, *prefixes*, *prefixed*, *infix*, *infixes*, *infixed*, *fix*, *fixes*, and *fixed*. Cross-product formation can produce a large number of words quickly, some of which may be illegal, so watch out. If cross-products produce illegal words, `munchlist` will not produce those flag combinations, and the flag will not be useful.

```
repl : condition* > [ - strip-string , ] append-string
```

The ~ option specifies that the associated flag is only active when a compound word is being formed. This is useful in a language like German, in which the form of a word sometimes changes inside a compound.

A `repl` is a conditional rule for modifying a root word. Up to eight conditions may be specified. If the conditions are satisfied, the rules on the rightside of the `repl` are applied, as follows:

1. If a `strip-string` is given, it is first stripped from the beginning or ending (as appropriate) of the root word.
2. The `append-string` is added at that point.

Byte addresses in mem are interpreted as physical memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching is likely to lead to unexpected results when read-only or write-only bits are present.

It is typically created by

```
mknod -m 660 /dev/mem c 1 1
chown root.mem /dev/mem
```

The file kmem is the same as mem, except that the kernel virtual memory rather than physical memory is accessed.

It is typically created by

```
mknod -m 640 /dev/kmem c 1 2
chown root.mem /dev/kmem
```

port is similar to mem, but the IO ports are accessed.

It is typically created by

```
mknod -m 660 /dev/port c 1 4
chown root.mem /dev/port
```


FILES

/dev/mem

/dev/kmem

/dev/port

SEE ALSO

mknod(1), chown(1), ioperm(2)

Linux, 21 November 1992

mouse

mouse—Serial mouse interface.

CONFIG

Serial mice are connected to a serial RS232/V24 dialout line; see cua(4) for a description.

DESCRIPTION

The pinout of the usual 9 pin plug as used for serial mice is

Pin	*Name*	*Used for*
2	RX	Data
3	TX	-12 V, Imax = 10 mA
4	DTR	+12 V, Imax = 10 mA
7	RTS	+12 V, Imax = 10 mA
5	GND	Ground

This is the specification; in fact, 9 V suffices with most mice.

The mouse driver can recognize a mouse by dropping RTS to low. About 14ms later, the mouse will send 0x4D on the data line. After a further 63ms, Microsoft-compatible mice will send 0x33. Other mice send different values.

These replace the screendump `ioctl`s of `console`(4), so the system administrator can control access using filesystem permissions.

The devices for the first eight virtual consoles may be created by

```
for x in 0 1 2 3 4 5 6 7 8; do
mknod -m 644 /dev/vcs$x c 7 $x;
mknod -m 644 /dev/vcsa$x c 7 $[$x+128];
done
chown root.tty /dev/vcs*
```

No `ioctl`() requests are supported.

EXAMPLES

You can do a screendump on vt3 by switching to vt1 and typing `cat /dev/vcs3 >foo`.

This program displays the character and screen attributes under the cursor of the second virtual console and then changes the background color there:

```
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>

void main()
{ int fd;
    struct {char lines, cols, x, y;} scrn;
    char ch, attrib;

    fd = open("/dev/vcsa2", O_RDWR);
    (void)read(fd, &scrn, 4);
    (void)lseek(fd, 4 + 2*(scrn.y*scrn.cols + scrn.x), 0);
    (void)read(fd, &ch, 1);
    (void)read(fd, &attrib, 1);
    printf("ch='%c' attrib=0x%02x\n", ch, attrib);
    attrib ^= 0x10;
    (void)lseek(fd, -1, 1);
    (void)write(fd, &attrib, 1);
}
```


FILES

`/dev/vcs[0-63]`

`/dev/vcsa[0-63]`

AUTHOR

Andries Brouwer (`aeb@cwi.nl`)

HISTORY

Introduced with version 1.1.92 of the Linux kernel.

SEE ALSO

`console`(4), `tty`(4), `ttys`(4), `selection`(1)

aliases

aliases—Aliases file for sendmail.

SYNOPSIS

```
aliases
```

DESCRIPTION

This file describes user ID aliases used by . The file resides in and is formatted as a series of lines of the form:

```
name: name_1, name_2, name_3, ...
```

The name is the name to alias, and the name_n are the aliases for that name. Lines beginning with whitespace are continuation lines. Lines beginning with # are comments.

Aliasing occurs only on local names. Loops cannot occur because no message will be sent to any person more than once.

After aliasing has been done, local and valid recipients who have a .forward file in their home directory have messages forwarded to the list of users defined in that file.

This is only the raw data file; the actual aliasing information is stored in a binary format in the file, using the program newaliases(1). A newaliases command should be executed each time the aliases file is changed for the change to take effect.

SEE ALSO

newaliases(1), dbm(3), sendmail(8), "Sendmail Installation and Operation Guide," "Sendmail: An Internetwork Mail Router."

BUGS

Because of restrictions in dbm(3), a single alias cannot contain more than about 1000 bytes of information. You can get longer aliases by "chaining"—that is, making the last name in the alias a dummy name that is a continuation alias.

HISTORY

The aliases file format appeared in BSD 4.0.

BSD 4, 10 May 1991

cfingerd

cfingerd—Configurable finger daemon.

SYNOPSIS

```
cfingerd [-c¦-d¦-e¦-o¦-v]
```

-c	Check configuration
-d	Run as daemon, not inetd
-e	Emulate local finger without inetd
-o	Turn off all finger queries
-v	Request version information

-c checks your installed configuration. This makes sure there are no existing errors in the current cfingerd.conf file.

-d runs cfingerd as a daemon. Don't run cfingerd this way if you're using inetd.

-e allows you to emulate a local finger on a user that exists on your system. This makes it so that you can test cfingerd on your system before installing it. Using the -e directive is the same as installing the software, typing finger username@ and getting the output. Using -e username does the same.

The supported caught signals are as follows:

```
SIGHUP, SIGINT, SIGQUIT, SIGILL, SIGTRAP, SIGABRT, SIGFPE, SIGUSR1, SIGSEGV, SIGUSR2, SIGPIPE, SIGALRM, SIGTERM, SIGCONT,
SIGTSTP, SIGTTIN, SIGTTOU, SIGIO, SIGXCPU, SIGXFSZ, SIGVTALRM, SIGPROF, SIGWINCH
```

FINGER PROGRAMS FILES SECTION (FILES finger programs)

These are the programs that are called when a specific action is take on the finger display.

`FINGER` is the file that is used when a user listing is requested from your machine. This is used in the standard user list and in the sorted user list, so it is wise to use the standard here: `/usr/sbin/userlist`.

`WHOIS` is the program that is used when a WHOIS request is done on a specific user.

FINGER FAKE USERS FILES SECTION (FILES finger fakeusers)

These are the ever–popular fake users that you can create on your system. These users are ones that don't exist (and should not exist, for that matter). These are, instead, treated as normal scripts that can be called for your use.

The format is as follows for fake users:

```
fake_username Script_name SEARCHBOOL script
```

`fake_username` is the name of the fake user you want to request. Make sure that this is a user that does not exist on your system. Keep in mind that if you create a fake user name and that user already exists, the fake user name will be shown.

`Script_name` is the standard name of your script. This is used in the display of your services listing.

`SEARCHBOOL` specifies whether parameters can be sent to that specific fake user. If you decide to use the `SEARCHBOOL` option (`TRUE` in this case), the passed variables are

`$1`	First passed option
`$2`	Second passed option
`$3`	Third passed option
`$4`	Fourth passed option

(If more than four options were passed to this, the request will be ignored, and an error message will be returned to the user who requested the finger request.)

`script` is the location of your script. It should be `chmod 700` and readable only by root.

If you do not specify any fake users, a fake user called `None` will be created. This is a fake user that does nothing and calls `/dev/null` for the script.

SERVICES HEADER CONFIGURE SECTION (CONFIG services header)

This is the display that is given during a services finger. It should be formatted the same way that you want it to display on the screen.

When specifying the finger formatted options, you should specify them as C formatted strings as well, with the standard options. This should always be given last in the display.

An example of this is

```
Welcome to this system's services!
User: Service name: Searchable:
--- -------- ----
%-8s %-20s %-s
```

Remember to keep the format string last or a `SIGSEGV` will result.

SERVICES POSITIONS CONFIGURE SECTION (CONFIG services positions)

This specifies where in the preceding display string that the information from a service listing is to appear. These numbers can be anywhere between `1` and `3`.

There is a special syntax for creating the entire banks of devices for a hard drive:

```
hd[a-d] 8/64
```

What this means is as follows: Create `hda`, and eight partitions on `hda` (`hda1` through `hda8`), starting with minor number 0. Then create `hdb`, and eight partitions, starting with minor number 64. Then `hdc`, and so on, with minor number 64*2 = 128—and so forth. These are automatically placed in the class `disk`. The necessary groups and batches are created so you can ask `MAKEDEV` to create `hd` or `hda` or `hda1` and expect it to do the correct thing.

Note that simple arithmetic is permitted for specifying the minor device number, as this often makes things much clearer and less likely to be accidentally broken.

SEE ALSO

`MAKEDEV`(8), `MAKEDEV.cfg`(5)

Version 1.4, January 1995

environ

`environ`—User environment.

SYNOPSIS

```
extern char **environ;
```

DESCRIPTION

An array of strings called the `environment` is made available by `exec`(2) when a process begins. By convention, these strings have the form `name=value`. Common examples are

`USER`	The name of the logged-in user (used by some BSD-derived programs).
`LOGNAME`	The name of the logged-in user (used by some System-V derived programs).
`HOME`	A user's login directory, set by `login`(1) from the password file `passwd`(5).
`LANG`	The name of a locale to use for locale categories when not overridden by `LC_ALL` or more specific environment variables.
`PATH`	The sequence of directory prefixes that `sh`(1) and many other programs apply in searching for a file known by an incomplete pathname. The prefixes are separated by `:`.
`SHELL`	The filename of the user's login shell.
`TERM`	The terminal type for which output is to be prepared.

Further names maybe placed in the environment by the `export` command and `name=value` in `sh`(1) or by the `setenv` command if you use `csh`(1). Arguments may also be placed in the environment at the point of an `exec`(2).

It is risky practice to set `name=value` pairs that conflict with well-known shell variables. Setting these could cause surprising behavior in subshells or `system`(3) commands.

SEE ALSO

`login`(1), `sh`(1), `bash`(1), `csh`(1), `tcsh`(1), `exec`(2), `system`(3)

Linux, 21 October 1996

expire.ctl

`expire.ctl`—Control file for Usenet article expiration.

Here's the complete list of mapping options:

`root_squash`	Map requests from UID/GID 0 to the anonymous UID/GID. Note that this does not apply to any other UIDs that might be equally sensitive, such as user `bin`.
`no_root_squash`	Turn off root squashing. This option is mainly useful for diskless clients.

`squash_uids` and `squash_gids`	This option specifies a list of UIDs or GIDs that should be subject to anonymous mapping. A valid list of IDs looks like this:
	`squash_uids=0-15,20,25-50`
	Usually, your squash lists will look a lot simpler, such as
	`squash_uids=0-100`

`all_squash`	Map all UIDs and GIDs to the anonymous user. Useful for NFS-exported public FTP directories, newsspool directories, and so on. The opposite option is `no_all_squash`, which is the default setting.
`map_daemon`	This option turns on dynamic UID/GID mapping. Each UID in an NFS request will be translated to the equivalent server UID, and each UID in an NFS reply will be mapped the other way round. This option requires that `ypbind` (8) runs on the client host. The default setting is `map_identity`, which leaves all UIDs untouched. The normal squash options apply regardless of whether dynamic mapping is requested.
`anonuid` and `anongid`	These options explicitly set the UID and GID of the anonymous account. This option is primarily useful for PC/NFS clients, where you might want all requests appear to be from one user. As an example, consider the export entry for `/home/joe` in the section "Example," which maps all requests to UID 150 (which is supposedly that of user `joe`).

EXAMPLE

```
# sample /etc/exports file
/ master(rw) trusty(rw,no_root_squash)
/projects proj*.local.domain(rw)
/usr *.local.domain(ro) @trusted(rw)
/home/joe pc001(rw,all_squash,anonuid=150,anongid=100)
/pub (ro,insecure,all_squash)
```

The first line exports the entire filesystem to machines `master` and `trusty`. In addition to write access, all UID squashing is turned off for host `trusty`. The second and third entry show examples for wildcard hostnames and netgroups (this is the entry `@trusted`). The fourth line shows the entry for the PC/NFS client discussed previously. The last line exports the public FTP directory to every host in the world, executing all requests under the nobody account. The `insecure` option in this entry also allows clients with NFS implementations that don't use a reserved port for NFS.

CAVEATS

Unlike other NFS server implementations, this `nfsd` allows you to export both a directory and a subdirectory thereof to the same host, for instance `/usr` and `/usr/X11R6`. In this case, the mount options of the most specific entry apply. For instance, when a user on the client host accesses a file in `/usr/X11R6`, the mount options given in the `/usr/X11R6` entry apply. This is also true when the latter is a wildcard or netgroup entry.

FILES

`/etc/exports`	Configuration file for `nfsd`(8)
`/etc/passwd`	The password file

DIAGNOSTICS

An error parsing the file is reported using `syslogd`(8) as level `NOTICE` from a `DAEMON` whenever `nfsd`(8) or `mountd`(8) is started. Any unknown host is reported at that time, but often not all hosts are not yet known to `named`(8) at boot time, so as hosts are found, they are reported with the same `syslogd`(8) parameters.

smb	A network filesystem that supports the SMB protocol, used by Windows for Workgroups, Windows NT, and LAN Manager. To use smb, you need a special mount program, which can be found in the ksmbfs package at ftp://sunsite.unc.edu/pub/Linux/system/Filesystems/smbfs.
ncpfs	A network filesystem that supports the NCP protocol, used by Novell NetWare. To use ncpfs, you need special programs found at ftp://linux01.gwdg.de/pub/ncpfs.

SEE ALSO

proc(5), fsck(8), mkfs(8), mount(8)

25 March 1996

fstab

fstab—Static information about the filesystems.

SYNOPSIS

```
#include <fstab.h>
```

DESCRIPTION

The file fstab contains descriptive information about the various filesystems. fstab is only read by programs and not written; it is the duty of the system administrator to properly create and maintain this file. Each filesystem is described on a separate line; fields on each line are separated by tabs or spaces. The order of records in fstab is important because fsck(8), mount(8), and umount(8) sequentially iterate through fstab doing their thing.

The first field (fs_spec) describes the block special device or remote filesystem to be mounted.

The second field (fs_file) describes the mount point for the filesystem. For swap partitions, this field should be specified as none.

The third field (fs_vfstype) describes the type of the filesystem. The system currently supports three types of filesystems:

minix	A local filesystem, supporting filenames of length 14 or 30 characters.
ext	A local filesystem with longer filenames and larger inodes. This filesystem has been replaced by the ext2 filesystem and should no longer be used.
ext2	A local filesystem with longer filenames, larger inodes, and a lot of other features.
xiafs	A local filesystem with longer filenames, larger inodes, and a lot of other features.
msdos	A local filesystem for MS-DOS partitions.
hpfs	A local filesystem for HPFS partitions.
iso9660	A local filesystem used for CD-ROM drives.
nfs	A filesystem for mounting partitions from remote systems.
swap	A disk partition to be used for swapping.

If vfs_fstype is specified as ignore, the entry is ignored. This is useful to show disk partitions that are currently unused.

The fourth field (fs_mntops) describes the mount options associated with the filesystem.

It is formatted as a comma-separated list of options. It contains at least the type of mount plus any additional options appropriate to the filesystem type. For documentation on the available options for non-NFS file systems, see mount(8). For documentation on all NFS-specific options, take a look at nfs(5). Common for all types of filesystems are the options noauto (do not mount when mount -a is given, such as at boot time) and user (allow a user to mount). For more details, see mount(8).

The fifth field (fs_freq) is used for these filesystems by the dump(8) command to determine which filesystems need to be dumped. If the fifth field is not present, a value of zero is returned and dump will assume that the filesystem does not need to be dumped.

are in `ditroff`. The `width` subfields gives the width of the character. The `height` subfield gives the height of the character (upwards is positive); if a character does not extend above the baseline, it should be given a zero height, rather than a negative height. The `depth` subfield gives the depth of the character, that is, the distance below the lowest point below the baseline to which the character extends (downwards is positive); if a character does not extend below above the baseline, it should be given a zero depth, rather than a negative depth. The `italic_correction` subfield gives the amount of space that should be added after the character when it is immediately to be followed by a character from a roman font. The `left_italic_correction` subfield gives the amount of space that should be added before the character when it is immediately to be preceded by a character from a roman font. The `subscript_correction` gives the amount of space that should be added after a character before adding a subscript. This should be less than the `italic_correction`.

A line in the charset section can also have the format

```
name "
```

This indicates that `name` is just another name for the character mentioned in the preceding line.

The word `kernpairs` starts the `kernpairs` section. This contains a sequence of lines of the form:

```
c1 c2 n
```

This means that when character `c1` appears next to character `c2`, the space between them should be increased by `n`. Most entries in `kernpairs` section will have a negative value for `n`.

FILES

`/usr/lib/groff/font/devname/DESC`	Device description file for device name.
`/usr/lib/groff/font/devname/F`	Font file for font `F` of device name.

SEE ALSO

`groff_out`(5), `gtroff`(1)

Groff Version 1.09, 14 February 1994

groff_out

`groff_out`—`groff` intermediate output format.

DESCRIPTION

This manual page describes the format output by GNU `troff`. The output format used by GNU `troff` is very similar to that used by UNIX device-independent `troff`. Only the differences are documented here.

The argument to the `s` command is in scaled points (units of points/`n`, where `n` is the argument to the `sizescale` command in the `DESC` file.) The argument to the `x` Height command is also in scaled points.

The first three output commands are guaranteed to be

```
x T device
x res n h v
x init
```

If the `tcommand` line is present in the `DESC` file, `troff` will use the following two commands:

`txxx` — `xxx` is any sequence of characters terminated by a space or a newline; the first character should be printed at the current position, the current horizontal position should be increased by the width of the first character, and so on for each character. The width of the character is that given in the font file, appropriately scaled for the current point size and rounded so that it is a multiple of the horizontal resolution. Special characters cannot be printed using this command.

```
time_t sem_ctime; /* last change time */
ushort sem_nsems; /* count of sems in set */
```

sem_perm	ipc_perm structure that specifies the access permissions on the semaphore set.
sem_otime	Time of last semop system call.
sem_ctime	Time of last semctl system call that changed a member of the above structure or of one semaphore belonging to the set.
sem_nsems	Number of semaphores in the set. Each semaphore of the set is referenced by a non-negative integer ranging from 0 to sem_nsems-1.

A semaphore is a data structure of type struct sem containing the following members:

```
ushort semval; /* semaphore value */
short sempid; /* pid for last operation */
ushort semncnt; /* no. of awaiting semval to increase */
ushort semzcnt; /* no. of awaiting semval = 0 */
```

semval	Semaphore value: a non-negative integer.
sempid	ID of the last process that performed a semaphore operation on this semaphore.
semncnt	Number of processes suspended awaiting for semval to increase.
semznt	Number of processes suspended awaiting for semval to become zero.

SHARED MEMORY SEGMENTS

A shared memory segment is uniquely identified by a positive integer (its shmid) and has an associated data structure of type struct shmid_ds, defined in <sys/shm.h>, containing the following members:

```
struct ipc_perm shm_perm;
int shm_segsz; /* size of segment */
ushort shm_cpid; /* pid of creator */
ushort shm_lpid; /* pid, last operation */
short shm_nattch; /* no. of current attaches */
time_t shm_atime; /* time of last attach */
time_t shm_dtime; /* time of last detach */
time_t shm_ctime; /* time of last change */
```

shm_perm	ipc_perm structure that specifies the access permissions on the shared memory segment.
shm_segsz	Size in bytes of the shared memory segment.
shm_cpid	ID of the process that created the shared memory segment.
shm_lpid	ID of the last process that executed a shmat or shmdt system call.
shm_nattch	Number of current alive attaches for this shared memory segment.
shm_atime	Time of the last shmat system call.
shm_dtime	Time of the last shmdt system call.
shm_ctime	Time of the last shmctl system call that changed shmid_ds.

SEE ALSO

ftok(3), msgctl(2), msgget(2), msgrcv(2), msgsnd(2), semctl(2), semget(2), semop(2), shmat(2), shmctl(2), shmget(2), shmdt (2)

Linux 0.99.13, 1 November 1993

issue

issue—Issue identification file.

Here is a sample file:

```
foo.important:announce-request@foo.com
foo.*:%s@mailer.foo.com
gnu.*:%s@prep.ai.mit.edu
:%s@uunet.uu.net
```

Using this file, postings to the moderated newsgroup in the left column will be sent to the address shown in the right column:

```
foo.important announce-request@foo.com
foo.x.announce foo-x-announce@mailer.foo.com
gnu.emacs.sources gnu-emacs-sources@prep.ai.mit.edu
comp.sources.unix comp-sources-unix@uunet.uu.net
```

HISTORY

Written by Rich $alz (`rsalz@uunet.uu.net`) for InterNetNews.

SEE ALSO

inews(1), inn.conf(5), libinn(3), wildmat(3)

/etc/modules

`/etc/modules`—Kernel modules to load at boot time.

DESCRIPTION

The `/etc/modules` file contains the names of kernel modules that are to be loaded at boot time, one per line. Comments begin with a `#`, and everything on the line after them are ignored.

Debian GNU/Linux version 0.93

motd

`motd`—Message of the day.

DESCRIPTION

The contents of `/etc/motd` are displayed by `login`(1) after a successful login but just before it executes the login shell.

The `motd` stands for “message of the day,” and this file has been traditionally been used for exactly that. (It requires much less disk space than mail to all users.)

FILES

/etc/motd

SEE ALSO

`login`(1) `issue`(5)

Linux, 29 December 1992

mtools

`mtools`—Table of DOS devices.

DESCRIPTION

`/etc/mtools.conf` and `~/.mtoolsrc` are the configuration files for `mtools`. These configuration files describe the following items:

Global configuration flags and variables

Per-drive flags and variables

Character translation tables

`/etc/mtools.conf` is the system-wide configuration file, and `~/.mtoolsrc` is the user's private configuration file.

GENERAL SYNTAX

The configuration files is made up of sections. Each section starts with a keyword identifying the section followed by a colon. Then follow variable assignments and flags. Variable assignments take the following form:

```
name=value
```

Flags are lone keywords without an equal sign and `value` following them. A section either ends at the end of the file or where the next section begins.

Lines starting with a hash (`#`) are comments. Newline characters are equivalent to whitespace (except where ending a comment). The configuration file is case insensitive except for items enclosed in quotes (such as filenames).

DEFAULT VALUES

For most platforms `mtools` contains reasonable compiled-in defaults. You usually don't need to bother with the configuration file, if all you want to do with `mtools` is access your floppy drives. On the other hand, the configuration file is needed if you also want to use `mtools` to access your hard disk partitions and `dosemu` image files.

GLOBAL VARIABLES

Global variables may be set to `1` or to `0`.

The following global flags are recognized:

`MTOOLS_SKIP_CHECK` — If this is set to `1`, `mtools` skips most of its sanity checks. This is needed to read some Atari disks that have been made with the earlier ROMs and that would not be recognized otherwise.

`MTOOLS_FAT_COMPATIBILITY` — If this is set to `1`, `mtools` skips the FAT size checks. Some disks have a bigger FAT than they really need. These are rejected if this option is not set.

`MTOOLS_LOWER_CASE` — If this is set to `1`, `mtools` displays all-uppercase short filenames as lowercase. This has been done to allow a behavior that is consistent with older versions of `mtools`, which didn't know about the case bits.

For example, inserting the following line into your configuration file instructs `mtools` to skip the sanity checks: `MTOOLS_SKIP_CHECK=1`.

Global variables may also be set via the environment: `export MTOOLS_SKIP_CHECK=1`.

PER-DRIVE FLAGS AND VARIABLES

Per-drive flags and values may be described in a drive section. A drive section starts with `drive driveletter:`.

Then follow variable-value pairs and flags.

GENERAL PURPOSE DRIVE VARIABLES

The following variables are available:

`file` — The name of the file or device holding the disk image. This is mandatory. The filename should be enclosed in quotes.

`G count` If this flag is specified, an article will only be sent to the site if it is posted to no more than `count` newsgroups.

`H count` If this flag is specified, an article will only be sent to the site if it has `count` or fewer sites in its Path line. This flag should only be used as a rough guide because of the loose interpretation of the Path header; some sites put the poster's name in the header, and some sites that might logically be considered to be one hop become two because they put the posting workstation's name in the header. The default value for `count` is one.

`I size` The flag specifies the size of the internal buffer for a file feed. If there are more file feeds than allowed by the system, they will be buffered internally in least recently used order. If the internal buffer grows bigger than `size` bytes, however, the data will be written out to the appropriate file.

`N modifiers` The newsgroups that a site receives are modified according to the modifiers, which should be chosen from the following set:

`m`	Only moderated groups
`u`	Only unmoderated groups

`S size` If the amount of data queued for the site is set to be larger than `size` bytes, then the server will switch to spooling, appending to the file specified by the `F` flag or `/news/spool/out.going/sitename` if the `F` flag is not specified. Spooling usually happens only for channel or exploder feeds.

`T type` This flag specifies the type of feed for the site; `type` should be a letter chosen from the following set:

`c`	Channel
`f`	File
`l`	Log entry only
`m`	Funnel (multiple entries feed into one)
`p`	Program
`x`	Exploder. Each feed is described in the section on feed types.

The default is `Tf`.

`W items` If a site is fed by file, channel, or exploder, this flag controls what information is written. If a site is fed by a program, only the asterisk (`*`) has any effect. The items should be chosen from the following set:

`b`	Size of the article in bytes.
`f`	Article's full pathname.
`g`	The newsgroup the article is in; if cross-posted, then the first of the groups this site gets.
`m`	Article's Message-ID.
`n`	Article's pathname relative to the spool directory.
`p`	The site that fed the article to the server; from the Path header.
`s`	The IP address of the site that sent the article.
`t`	Time article was received as seconds since epoch.
`*`	Names of the appropriate funnel entries; or all sites that get the article.
`D`	Value of the Distribution header; `?` if none present.
`H`	All headers.
`N`	Value of the Newsgroups header.
`O`	Overview data.
`R`	Information needed for replication. More than one letter can be used; the entries will be separated by a space and written in the order in which they are specified. The default is `Wn`.

HISTORY

Written by Landon Curt Noll (`chongo@toad.com`) and Rich $alz (`rsalz@uunet.uu.net`) for InterNetNews.

SEE ALSO

`control.ctl`*(5)*, `ctlinnd`*(8)*, `expire`*(8)*, `innd`*(8)*, `news.daily`*(8)*, `nntpsend`*(8)*, `newslog`*(8)*

nfs

`nfs`—NFS `fstab` format and options.

SYNOPSIS

`/etc/fstab`

DESCRIPTION

The `fstab` file contains information about which filesystems to mount where and with what options. For NFS mounts, it contains the server name and exported server directory to mount from, the local directory that is the mount point, and the NFS-specific options that control the way the filesystem is mounted. Here is an example from an `/etc/fstab` file from an NFS mount.

```
server:/usr/local/pub /pub nfs rsize=8192,wsize=8192,timeo=14,intr
```


OPTIONS

`rsize=n` The number of bytes NFS uses when reading files from an NFS server. The default value is dependent on the kernel, currently `1024` bytes. (However, throughput is improved greatly by asking for `rsize=8192`.)

`wsize=n` The number of bytes NFS uses when writing files to an NFS server. The default value is dependent on the kernel, currently `1024` bytes. (However, throughput is improved greatly by asking for `wsize=8192`.)

`timeo=n` The value in tenths of a second before sending the first retransmission after an RPC time-out. The default value is 7 tenths of a second. After the first time-out, the time-out is doubled after each successive time-out until a maximum time-out of 60 seconds is reached or the enough retransmissions have occurred to cause a major time-out. Then, if the filesystem is hard mounted, each new time-out cascade restarts at twice the initial value of the previous cascade, again doubling at each retransmission. The maximum time-out is always 60 seconds. Better overall performance may be achieved by increasing the time-out when mounting on a busy network, to a slow server, or through several routers or gateways.

`retrans=n` The number of minor time-outs and retransmissions that must occur before a major time-out occurs. The default is `3` time-outs. When a major time-out occurs, the file operation is either aborted or a “server not responding” message is printed on the console.

`acregmin=n` The minimum time in seconds that attributes of a regular file should be cached before requesting fresh information from a server. The default is `3` seconds.

`acregmax=n` The maximum time in seconds that attributes of a regular file can be cached before requesting fresh information from a server. The default is `60` seconds.

`acdirmin=n` The minimum time in seconds that attributes of a directory should be cached before requesting fresh information from a server. The default is `30` seconds.

`acdirmax=n` The maximum time in seconds that attributes of a directory can be cached before requesting fresh information from a server. The default is `60` seconds.

`actimeo=n` Using `actimeo` sets all of `acregmin`, `acregmax`, `acdirmin`, and `acdirmax` to the same value. There is no default value.

pnm

pnm—Portable anymap file format.

DESCRIPTION

The pnm programs operate on portable bitmaps, graymaps, and pixmaps produced by the pbm, pgm, and ppm segments. There is no file format associated with pnm itself.

SEE ALSO

anytopnm(1), rasttopnm(1), tifftopnm(1), xwdtopnm(1), pnmtops(1), pnmtorast(1), pnmtotiff(1), pnmtoxwd(1), pnmarith(1), pnmcat(1), pnmconvol(1), pnmcrop(1), pnmcut(1), pnmdepth(1), pnmenlarge(1), pnmfile(1), pnmflip(1), pnmgamma(1), pnmindex(1), pnminvert(1), pnmmargin(1), pnmnoraw(1), pnmpaste(1), pnmrotate(1), pnmscale(1), pnmshear(1), pnmsmooth(1), pnmtile(1), ppm(5), pgm(5), pbm(5)

AUTHOR

27 September 1991

ppm

ppm—Portable pixmap file format.

DESCRIPTION

The portable pixmap format is a lowest common denominator color image file format. The definition is as follows:

A "magic number" for identifying the file type. A ppm file's magic number is the two characters P3.

Whitespace (blanks, Tabs, CRs, LFs).

A width, formatted as ASCII characters in decimal.

Whitespace.

A height, again in ASCII decimal.

Whitespace.

The maximum color-component value, again in ASCII decimal.

Whitespace.

Width * height pixels, each three ASCII decimal values between 0 and the specified maximum value, starting at the top-left corner of the pixmap, proceeding in normal English reading order. The three values for each pixel represent red, green, and blue; a value of 0 means that color is off, and the maximum value means that color is maxed out.

Characters from a # to the next end-of-line are ignored (comments).

No line should be longer than 70 characters.

Here is an example of a small pixmap in this format:

```
P3
# feep.ppm
4 4
15
0 0 0 0 0 0 0 0 0 0 15 0 15
0 0 0 0 15 7 0 0 0 0 0 0
0 0 0 0 0 0 0 15 7 0 0 0
15 0 150 0 0 00 0 0 0 0
```

Programs that read this format should be as lenient as possible, accepting anything that looks remotely like a pixmap.

```
newphrase ::= id word* ;
word ::= id ¦ num ¦ string ¦ :
```

Identifiers are case sensitive. Keywords are in lowercase only. The sets of keywords and identifiers can overlap. In most environments, RCS uses the ISO8859/1 encoding: visible graphic characters are codes 041–176 and 240–377, and whitespace characters are codes 010–015 and 040.

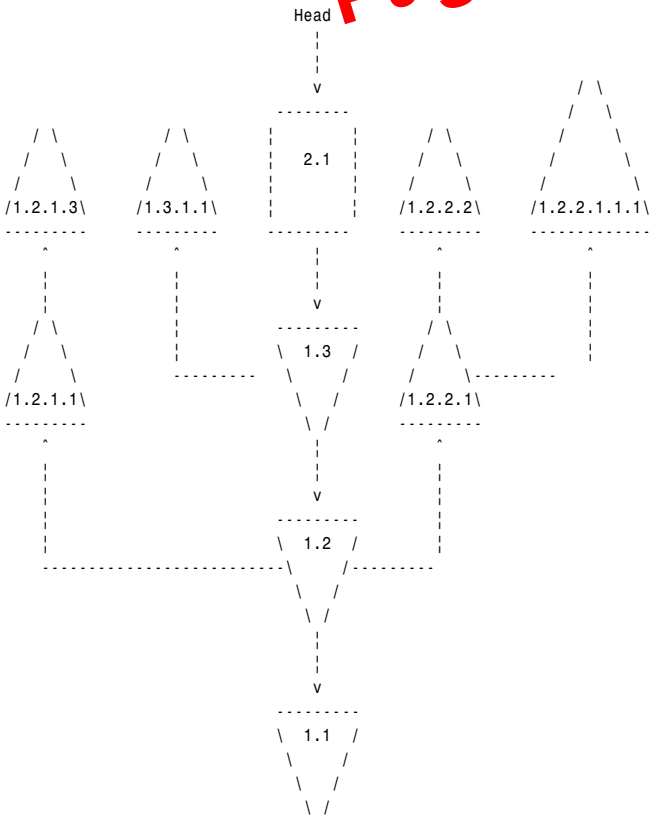
Dates, which appear after the `date` keyword, are of the form *Y.mm.dd.hh.mm.ss*, where *Y* is the year, *mm* the month (01–12), *dd* the day (01–31), *hh* the hour (00–23), *mm* the minute (00–59), and *ss* the second (00–60). *Y* contains just the last two digits of the year for years from 1900 through 1999, and all the digits of years thereafter. Dates use the Gregorian calendar; times use UTC.

The `newphrase` productions in the grammar are reserved for future extensions to the format of RCS files. No `newphrase` will begin with any keyword already in use.

The `delta` nodes form a tree. All nodes whose numbers consist of a single pair (such as 2.3, 2.1, 1.3, and so on) are on the trunk and are linked through the next field in order of decreasing numbers. The `head` field in the `admin` node points to the head of that sequence (contains the highest pair). The `branch` node in the `admin` node indicates the default branch (or revision) for most RCS operations. If empty, the default branch is the highest branch on the trunk.

All `delta` nodes whose numbers consist of 2*n* fields (*n*2) (such as 3.1.1.1, 2.2.2.2, and so on) are linked as follows. All nodes whose first 2*n*–1 number fields are identical are linked through the `next` field in order of increasing numbers. For each such sequence, the `delta` node whose number is identical to the first 2*n*–2 number fields of the deltas on that sequence is called the branchpoint. The `branches` field of a node contains a list of the numbers of the first nodes of all sequences for which it is a branchpoint. This list is ordered in increasing numbers.

The following diagram shows an example of an RCS file's organization.



protocol) for its service. The C library routines `getservent`(3), `getservbyname`(3), `getservbyport`(3), `setservent`(3), and `endservent`(3) support querying this file from programs.

Port numbers are assigned by the IANA (Internet Assigned Numbers Authority), and their current policy is to assign both TCP and UDP protocols when assigning a port number. Therefore, most entries will have two entries, even for TCP-only services.

Port numbers below 1024 (so-called low-numbered ports) can only be bound to by root (see `bind`(2), `tcp`(7), and `udp`(7).) This is so that clients connecting to low-numbered ports can trust that the service running on the port is the standard implementation and not a rogue service run by a user of the machine. Well-known port numbers specified by the IANA are normally located in this root-only space.

The presence of an entry for a service in the `services` file does not necessarily mean that the service is currently running on the machine. See `inetd.conf`(5) for the configuration of Internet services offered. Note that not all networking services are started by `inetd`(8) and so won't appear in `inetd.conf`(5). In particular, news (NNTP) and mail (SMTP) servers are often initialized from the system boot scripts.

The location of the `services` file is defined by `PATH_SERVICES` in `/usr/include/netdb.h`. This is usually set to `/etc/services`.

Each line describes one service and is of the form:

```
service-name port/protocol [aliases ...]
```

`service-name` — The friendly name the service is known by and looked up under. It is case sensitive. Often, the client program is named after the `service-name`.

`port` — The port number (in decimal) to use for this service.

`protocol` — The type of protocol to be used. This field should match an entry in the `protocols`(5) file. Typical values include `tcp` and `udp`.

`aliases` — An optional space- or tab-separated list of other names for this service (see the Bugs section below). Again, the names are case sensitive.

Either spaces or tabs may be used to separate the fields.

Comments are started by the hash sign (`#`) and continue until the end of the line. Blank lines are skipped.

The `service-name` should begin in the first column of the file because leading spaces are not stripped. `service-names` can be any printable characters excluding space and tab; however, a conservative choice of characters should be used to minimize inter-operability problems. For example, a–z, 0–9, and hyphen (–) would seem a sensible choice.

Lines not matching this format should not be present in the file. (Currently, they are silently skipped by `getservent`(3), `getservbyname`(3), and `getservbyport`(3). However, this behavior should not be relied on.)

As a backwards compatibility feature, the slash (`/`) between the port number and protocol name can in fact be either a slash or a comma (`,`). Use of the comma in modern installations is depreciated.

This file might be distributed over a network using a network-wide naming service such as Yellow Pages/NIS or BIND/Hesiod.

A sample `services` file might look like this:

```
netstat     15/tcp
qotd        17/tcp     quote
msp         18/tcp     # message send protocol
msp         18/udp     # message send protocol
chargen     19/tcp     ttytst source
chargen     19/udp     ttytst source
ftp         21/tcp
#           22 - unassigned
telnet      23/tcp
```

`Ramdac "ramdac-type"`

This optional entry specifies the type of RAMDAC used on the graphics board. This is only used by a few of the X servers, and in most cases, it is not required because the X servers will probe the hardware to determine the RAMDAC type where possible.

`DacSpeed speed`

This optional entry specifies the RAMDAC speed rating (which is usually printed on the RAMDAC chip). The speed is in MHz. This is only used by a few of the X servers and only needs to be specified when the speed rating of the RAMDAC is different from the default built in to the X server.

`Clocks clock ...`

Specifies the dotclocks that are on your graphics board. The clocks are in MHz and may be specified as a floating-point number. The value is stored internally to the nearest kHz. The ordering of the clocks is important. It must match the order in which they are selected on the graphics board. Multiple `Clocks` lines may be specified. For boards with programmable clock chips, the `ClockChip` entry should be used instead of this. A `Clocks` entry is not mandatory for boards with non-programmable clock chips but is highly recommended because it prevents the clock probing phase during server startup. This clock probing phase can cause problems for some monitors.

`ClockChip "clockchip-type"`

This optional entry is used to specify the clock chip type on graphics boards that have a programmable clock generator. Only a few X servers support programmable clock chips. For details, see the appropriate X server manual page.

`ClockProg command [textclock]`

This optional entry runs `command` to set the clock on the graphics board instead of using the internal code. The command string must consist of the full pathname (and any flags). When using this option, a `Clocks` entry is required to specify what clock values are to be made available to the server (up to 128 clocks may be specified). The optional `textclock` value is to tell the server that `command` must be run to restore the text-mode clock at server exit (or when VT switching). `textclock` must match one of the values in the `Clocks` entry. This parameter is required when the clock used for text mode is a programmable clock.

The command is run with the real user's ID with `stdin` and `stdout` set to the graphics console device. Two arguments are passed to the command. The first is the clock frequency in MHz as a floating-point number and the second is the index of the clock in the `Clocks` entry. The command should return an exit status of 0 when successful and something in the range 1–254 otherwise.

The `command` is run when the initial graphics mode is set and when changing screen resolution with the hotkey sequences. If the program fails at initialization, the server exits. If it fails during a mode switch, the mode switch is aborted but the server keeps running. It is assumed that if the command fails, the clock has not been changed.

`Option optionstring`

This optional entry allows the user to select certain options provided by the drivers. Multiple `Option` entries may be given. The supported values for `optionstring` are given in the appropriate X server manual pages.

`VideoRam mem`

This optional entry specifies the amount of video RAM that is installed on the graphics board. This is measured in kilobytes. In most cases, this is not required because the X server probes the graphics board to determine this quantity.

`BIOSBase baseaddress`

This optional entry specifies the base address of the video BIOS for the VGA board. This address is usually 0xC0000, which is the default the X servers use. Some systems, particularly those with on-board VGA hardware, have the BIOS located at an alternate address, usually 0xE0000. If your video BIOS is at an address other than 0xC0000, you must specify the base address in the `XF86Config` file. Note that some X servers don't access the BIOS at all and those that do only use the BIOS when searching for information during the hardware probe phase.

Also note that `hd=` can be used to refer to the next unspecified drive in the (`a`, …, `h`) sequence. For the following discussions, the `hd=` option will be cited for brevity. See the file `README.ide` in `linux/drivers/block` for more details.

THE hd=cyls,heads,sects[,wpcom[,irq]] OPTIONS

These options are used to specify the physical geometry of the disk. Only the first three values are required. The cylinder, head, and sectors values are those used by `fdisk`. The write precompensation value is ignored for IDE disks. The IRQ value specified is the IRQ used for the interface that the drive resides on and is not really a drive-specific parameter.

THE hd=serialize OPTION

The dual IDE interface CMD-640 chip is broken as designed such that when drives on the secondary interface are used at the same time as drives on the primary interface, it will corrupt your data. Using this option tells the driver to make sure that both interfaces are never used at the same time.

THE hd=dtc2278 OPTION

This option tells the driver that you have a DTC-2278D IDE interface. The driver then tries to do DTC-specific operations to enable the second interface and to enable faster transfer modes.

THE hd=noprobe OPTION

Do not probe for this drive. The following line

```
hdb=noprobe hdb=1166,7,17
```

disables the probe but will specify the drive geometry so that it is registered as a valid block device and hence usable.

THE hd=nowerr OPTION

Some drives apparently have the `WRERR STAT` bit stuck on permanently. This enables a work-around for these broken devices.

THE hd=cdrom OPTION

This tells the IDE driver that there is an ATAPI compatible CD-ROM attached in place of a normal IDE hard disk. In most cases, the CD-ROM is identified automatically, but if it isn't, then this might help.

STANDARD ST-506 DISK DRIVER OPTIONS (hd=)

The standard disk driver can accept geometry arguments for the disks similar to the IDE driver. Note however that it only expects three values (`C`/`H`/`S`); any more or any less and it will silently ignore you. Also, it only accepts `hd=` as an argument; `hda=` and so on are not valid here. The format is as follows:

```
hd=cyls,heads,sects
```

If there are two disks installed, the preceding line is repeated with the geometry parameters of the second disk.

XT DISK DRIVER OPTIONS (xd=)

If you are unfortunate enough to be using one of these old 8-bit cards that move data at a whopping 125KB/s, then here is the scoop. If the card is not recognized, you must use a boot arg of the form

```
xd=type,irq,iobase,dma_chan
```

The `type` value specifies the particular manufacturer of the card, and you use one of the following: `0`=generic, `1`=DTC, `2`, `3`, `4`=Western Digital, `5`, `6`, `7`=Seagate, or `8`=OMTI. The only difference between multiple types from the same manufacturer is the BIOS string used for detection, which is not used if the type is specified.

The `xd_setup()` function does no checking on the values and assumes that you entered all four values. Don't disappoint it. Here is a sample usage for a WD1002 controller with the BIOS disabled or removed, using the default XT controller parameters:

```
xd=2,5,0x320,3
```

Request	Initial Value	Cause Break	Explanation
.)x	-	Yes	End index item.
.)z	-	Yes	End floating keep.
.++ m H	-	No	Define paper section.
			m defines the part of the paper and can be C (chapter), A (appendix), P (preliminary, such as abstract, table of contents, and so on), B (bibliography), RC (chapters renumbered from page one each chapter), or RA (appendix renumbered from page one).
.+c T	-	Yes	Begin chapter (or appendix and so on as set by .++). T is the chapter title.
.1c	1	Yes	One column format on a new page.
.2c	1	Yes	Two column format on a new page.
.EN	-	Yes	Space after equation produced by eqn or neqn.
.EQ x y	-	Yes	Set up equation; break out and add space.
			The optional argument x may be I to indent equation (default), L to left-adjust the equation, or C to center the equation.
.GE	-	Yes	End gremlin picture.
.GS	-	Yes	Begin gremlin picture.
.PE	-	Yes	End pic picture.
.PS	-	Yes	Begin pic picture.
.TE	-	Yes	End table.
.TH	-	Yes	End heading section of table.
.TS x	-	Yes	Begin table; if x is H, table has repeated heading.
.b x	No	No	Print x in boldface; if no argument switch to boldface.
.ba +n	0	Yes	Augments the base indent by n.
			This indent is used to set the indent on regular text (like paragraphs).
.bc	No	Yes	Begin new column.
.bi x	No	No	Print x in bold italics (no fill only).
.bu	-	Yes	Begin bulleted paragraph.
.bx x	No	No	Print x in a box (no fill only).
.ef 'x'y'z'	""	No	Set even footer to x y z.
.eh 'x'y'z'	""	No	Set even header to x y z.
.fo 'x'y'z'	""	No	Set footer to x y z.
.hx	-	No	Suppress headers and footers on next page.
.he 'x'y'z'	""	No	Set header to x y z.
.hl	-	Yes	Draw a horizontal line.
.i x	No	No	Italicize x; if x is missing, italic text follows.
.ip x y	No	Yes	Start indented paragraph with hanging tag x. Indentation is y ens (default is 5).

SECTIONS

Sections are started with .SH followed by the heading name. If the name contains spaces and appears on the same line as .SH, then place the heading in double quotes. Traditional headings include NAME, SYNOPSIS, DESCRIPTION, OPTIONS, FILES, SEE ALSO, DIAGNOSTICS, BUGS, and AUTHOR. The only required heading is NAME, which should be followed on the next line by a one line description of the program:

```
.SH NAME
chess \- the game of chess
```

It is extremely important that this format is followed and that there is a backslash before the single dash that follows the command name. This syntax is used by the makewhatis(8) program to create a database of short command descriptions for the whatis(1) and apropos(1) commands.

OTHER MACROS

Other macros include the following:

.DT	Default tabs.
.HP	Begin hanging indent.
.IP	Begin paragraph with hanging tag. This is the same as .TP, except the tag is given on the same line, not on the following line.
.LP	Same as .PP.
.PD	Set inter-paragraph distance to argument.
.PP	Begin a new paragraph.
.RE	End relative indent (indented paragraph).
.RS	Start relative indent (indented paragraph).
.SS	Subheading (like .SH but used for a subsection).
.TP	Begin paragraph with hanging tag. The tag is given on the next line. This command is similar to .IP.

FILES

```
/usr/local/lib/groff/tmac/tmac.an
/usr/man/whatis
```

SEE ALSO

groff(1), man(1), whatis(1), apropos(1), makewhatis(8)

Linux, 25 July 1993

signal

signal—List of available signals.

DESCRIPTION

Linux supports the signals listed in this section. Several signal numbers are architecture dependent. First are the signals described in POSIX.1:

abort(3) alarm(1) Next various other signals.

(Here, – denotes that a signal is absent; there, where three values are given, the first one is usually valid for alpha and sparc, the middle one for i386 and ppc, the last one for mips. Signal 29 is SIGINFO/SIGPWR on an alpha but SIGLOST on a sparc.)

The letters in the Action column have the following meanings:

A	Default action is to terminate the process.
B	Default action is to ignore the signal.
C	Default action is to dump core.
D	Default action is to stop the process.
E	Signal cannot be caught.
F	Signal cannot be ignored.
G	Not a POSIX.1 conformant signal.

CONFORMING TO

POSIX.1

BUGS

`SIGIO` and `SIGLOST` have the same value. The latter is commented out in the kernel sources, but the build process of some software still thinks that Signal `29` is `SIGLOST`.

SEE ALSO

`kill`(1), `kill`(2), `setitimer`(2)

Linux 1.3.88, 14 April 1996

suffixes

`suffixes`—List of file suffixes.

DESCRIPTION

It is customary to indicate the contents of a file with the file suffix, which consists of a period followed by one or more letters. Many standard utilities, such as compilers, use this to recognize the type of file they are dealing with. The `make`(1) utility is driven by rules based on file suffixes.

Following is a list of suffixes that are likely to be found on a Linux system:

Suffix	*File Type*
,v	Files for RCS (Revision Control System)
-	Backup file
.C	C++ source code
.F	FORTRAN source with `cpp`(1) directives
.S	Assembler source with `cpp`(1) directives
.Z	File compressed using `compress`(1)
.[0-9]+pk	TeX font files
.[1-9]	Manual page for the corresponding section
.[1-9][a-z]	Manual page for section plus subsection
.a	Static object code library
.afm	PostScript font metrics
.arc	ARC archive
.arj	ARJ archive
.asc	PGP ASCII-armored data

continues

Suffix	*File Type*
.awk	AWK language program
.bak	Backup file
.bm	Bitmap source
.c	C source
.cat	Message catalog files
.cc	C++ source
.cf	Configuration file
.conf	Configuration file
.config	Configuration file
.cweb	Donald Knuth's WEB for C
.dat	Data file
.def	Modula-2 source for definition modules
.def	Other definition files
.diff	ASCII File differences
.doc	Documentation file
.dvi	TEX device independent output
.el	EMACS lisp source
.elc	Compiled EMACS lisp
.eps	Encapsulated PostScript
.f	FORTRAN source
.fas	Precompiled common lisp
.fi	FORTRAN include files
.gif	Graphics Interchange Format
.gsf	Ghostscript fonts
.gz	File compressed using `gzip`(1)
.h	C or C++ header files
.hlp	Help file
.htm	HTML file imported without renaming from a brain-damaged OS
.html	HTML document used with the World Wide Web
.i	C source after preprocessing
.idx	Reference or datum-index file for hypertext or database system
.icon	Bitmap source
.image	Bitmap source
.in	Configuration template, especially for GNU `autoconf`
.info	Files for the EMACS info browser
.java	A Java source file
.jpg	JPEG compressed picture format
.l	`lex`(1) or `flex`(1) files
.lib	Common lisp library
.ln	Files for use with `lint`(1)
.lsp	Common lisp source
.m4	`M4`(1) source
.mac	Macro files for various programs

BUGS

Many of these bugs are harmless. Most of them cause local errors that can be fixed in the converted manuscript.

- Some macros and macro arguments are not recognized.
- Commands that are not separated from their argument by a space are not properly parsed (such as `.sp3i`).
- When some operators (notably `over`, `sub`, and `sup`) are renamed (via `define`) and then they are encountered in the text, `tr2tex` treats them as ordinary macros and does not apply their rules.
- `rpile`, `lpile`, and `cpile` are treated the same as `cpile`.
- `rcol` and `lcol` are treated the same as `ccol`.
- Math-mode `size`, `gsize`, `fat`, and `gfont` are ignored.
- `lineup` and `mark` are ignored. The rules are so different.
- Some `troff` commands are translated to commands that require delimiters that have to be explicitly put. Because they are sometimes not put in `troff`, they can create problems. Example: `.nf` is not closed by `.fi`.
- When local motions are converted to `nraise` or `nlower`, an `nhbox` is needed, which must be put manually after the conversion.
- `a sub i sub j` is converted to `a_i_j`, which TeX parses as `a_i{}_j` with a non-printing box. It is vague. `a sub {i subj}` is parsed correctly and converted to `a_{i_j}`.
- Line spacing is not changed within a paragraph in TeX, which is a bad practice anyway. TeX uses the last line spacing in effect in that paragraph.

TO DO

Access registers via the `.nr` command.

SEE ALSO

`texmatch`(9), `trmatch`(9)

AUTHOR

Kamal Al-Yahya, Stanford University

1 January 1987

Unicode

Unicode—The unified 16-bit super character set.

DESCRIPTION

The international standard ISO 10646 defines the Universal Character Set (UCS). UCS contains all the characters of all other character-set standards. It also guarantees round-trip compatibility; conversion tables can be built such that no information is lost when a string is converted from any other encoding to UCS and back.

UCS contains the characters required to represent almost all known languages. This includes apart from the many languages that use extensions of the Latin script also the following scripts and languages: Greek, Cyrillic, Hebrew, Arabic, Armenian, Gregorian, Japanese, Chinese, Hiragana, Katakana, Korean, Hangul, Devangari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayam, Thai, Lao, Bopomofo, and a number of others. Work is going on to include further scripts such as Tibetan, Khmer, Runic, Ethiopian, Hieroglyphics, various Indo-European languages, and many others. For most of these latter scripts, it was not yet clear how they can be encoded best when the standard was published in 1993. In addition to the characters required by these scripts, also a large number of graphical, typographical, mathematical, and scientific symbols such as those provided by TeX, PostScript, MS-DOS, Macintosh, Videotext, OCR, and many word processing systems have been included, as well as special codes that guarantee round-trip compatibility to all other existing character-set standards.

intro

`intro`—Introduction to administration and privileged commands.

DESCRIPTION

This chapter describes commands that either can be or are only used by the superuser, such as daemons and machine or hardware-related commands.

AUTHORS

Look at the header of the manual page for the authors and copyright conditions. Note that these can be different from page to page.

Linux, 24 July 1993

adduser, addgroup

`adduser`, `addgroup`—Add a user or group to the system.

SYNOPSIS

```
adduser [--system [--home directory] [--group]] [--quiet]
 [--force-badname] [--help] [--version] [--debug] username
adduser [--quiet] [--force-badname] [--help] [--version]
[--debug] username group
adduser [--group] [--quiet] [--force-badname] [--help]
[--version] [--debug] group
```

DESCRIPTION

`adduser` and `addgroup` add users and groups to the system according to information provided in the configuration file `/etc/adduser.conf`. `adduser` and `addgroup` automatically determine the UID or GID and place the entity in the password or group file as appropriate.

If necessary, `adduser` creates a home directory for the new user, copies “skeletal” user files to it from `/etc/skel`, and allows the system administrator to set an initial password and finger information for the user.

Because it needs to be able to write to such files as `/etc/passwd`, `adduser` can only be run as root.

Generally, there are two types of users and groups on a system: those users that log into the system and those “non-user” accounts and groups that exist for various system tasks and projects. Henceforth, user will refer to the login type and system user or group will refer to the type used for system maintenance and projects.

By default, each user in Debian GNU/Linux is given a corresponding group with the same name and ID, allowing people easily to give access to their home directories to others. This option can be turned off in the configuration file, in which case each user is, by default, added to a group called users.

Under Debian GNU/Linux, IDs less than or equal to 100 are allocated by the base system maintainer for various purposes. IDs from 101 to the value specified in the configuration file (`1000`, by default) are used for system users and groups. IDs greater than 1000 are reserved for users and their corresponding groups.

When invoked with a single name, `adduser` creates a user with that name. When given two names, `adduser` assumes that the first name represents an existing user and that the second name represents an existing group. In this case, the user is added to the group.

DESCRIPTION

`cfdisk` is a curses-based program for partitioning a hard disk drive. The device can be any one of the following:

```
/dev/hda [default]
/dev/hdb
/dev/sda
/dev/sdb
/dev/sdc
/dev/sdd
```

`cfdisk` first tries to read the geometry of the hard disk. If it fails, an error message is displayed and `cfdisk` exits. This should only happen when partitioning a SCSI drive on an adapter without a BIOS. To correct this problem, you can set the cylinders, heads, and sectors-per-track on the command line. Next, `cfdisk` tries to read the current partition table from the disk drive. If it is unable to figure out the partition table, an error is displayed and the program exits. This might also be caused by incorrect geometry information and can be overridden on the command line. Another way around this problem is with the `-z` option. This will ignore the partition table on the disk.

The main display is composed of four sections, from top to bottom: the header, the partitions, the command line, and a warning line. The header contains the program name and version number followed by the disk drive and its geometry. The partitions section always displays the current partition table. The command line is the place where commands and text are entered. The available commands are usually displayed in brackets. The warning line is usually empty except when there is important information to be displayed. The current partition is highlighted with reverse video (or an arrow if the `-a` option is given). All partition-specific commands apply to the current partition.

The format of the partition table in the partition's section is, from left to right: Name, Flags, Partition Type, Filesystem Type, and Size. The name is the partition device name. The flags can be `Boot`, which designates a bootable partition or NC, which stands for "Not Compatible with DOS or OS/2." DOS, OS/2, and possibly other operating systems require the first sector of the first partition on the disk and all logical partitions to begin on the second head. This wastes the second through the last sector of the first track of the first head (the first sector is taken by the partition table itself). `cfdisk` allows you to recover these "lost" sectors with the maximize command (`m`). Note that `fdisk`(8) and some early versions of DOS create all partitions with the number of sectors already maximized. For more information, see the maximize command later in this chapter. The partition type can be `Primary` or `Logical`. For unallocated space on the drive, the partition type can also be `Pri/Log` or empty (if the space is unusable). The filesystem type section displays the name of the filesystem used on the partition, if known. If it is unknown, then Unknown and the hex value of the filesystem type are displayed. A special case occurs when there are sections of the disk drive that cannot be used (because all the primary partitions are used). When this is detected, the filesystem type is displayed as `Unusable`. The size field displays the size of the partition in megabytes (by default). It can also display the size in sectors and cylinders (see the change units command later in this chapter). If an asterisks (`*`) appears after the size, this means that the partition is not aligned on cylinder boundaries.

DOS 6.*X* WARNING

The DOS 6.*x* `FORMAT` command looks for some information in the first sector of the data area of the partition and treats this information as more reliable than the information in the partition table. `DOS FORMAT` expects `DOS FDISK` to clear the first 512 bytes of the data area of a partition whenever a size change occurs. `DOS FORMAT` looks at this extra information even if the `/U` flag is given; we consider this a bug in `DOS FORMAT` and `DOS FDISK`.

The bottom line is that if you use `cfdisk` or `fdisk` to change the size of a DOS partition table entry and then you must also use `dd` to zero the first 512 bytes of that partition before using `DOS FORMAT` to format the partition. For example, if you were using `cfdisk` to make a DOS partition table entry for `/dev/hda1`, then (after exiting `fdisk` or `cfdisk` and rebooting Linux so that the partition table information is valid), you use the command `dd if=/dev/zero of=/dev/hda1 bs=512 count=1` to zero the first 512 bytes of the partition.

Be extremely careful if you use the `dd` command because a small typo can make all of the data on your disk useless.

For best results, you should always use an OS-specific partition table program. For example, you should make DOS partitions with the `DOS FDISK` program and Linux partitions with the Linux `fdisk` or Linux `cfdisk` program.

COMMANDS

`cfdisk` commands can be entered by pressing the desired key (pressing Enter after the command is not necessary). Here is a list of the available commands:

`b` Toggle bootable flag of the current partition. This allows you to select which primary partition is bootable on the drive.

`d` Delete the current partition. This will convert the current partition into free space and merge it with any free space immediately surrounding the current partition. A partition already marked as free space or marked as unusable cannot be deleted.

`g` Change the disk geometry (cylinders, heads, or sectors-per-track). Warning: This option should only be used by people who know what they are doing. A command-line option is also available to change the disk geometry. While at the change disk geometry command line, you can choose to change cylinders (`d`), heads (`h`), and sectors per track (`s`). The default value will be printed at the prompt, which you can accept by simply pressing the Enter key or you can exit without changes by pressing the Esc key. If you want to change the default value, simply enter the desired value and press Enter. The altered disk parameter values do not take effect until you return to the main menu (by pressing Enter or Esc at the change disk geometry command line. If you change the geometry such that the disk appears larger, the extra sectors are added at the end of the disk as free space. If the disk appears smaller, the partitions that are beyond the new last sector are deleted and the last partition on the drive (or the free space at the end of the drive) is made to end at the new last sector.

`h` Print the help screen.

`m` Maximize disk usage of the current partition. This command will recover the the unused space between the partition table and the beginning of the partition, at the cost of making the partition incompatible with DOS, OS/2, and possibly other operating systems. This option will toggle between maximal disk usage and DOS, OS/2, and so on compatible disk usage. The default when creating a partition is to create DOS, OS/2, and so on compatible partitions.

`n` Create new partitions from free space. If the partition type is `Primary` or `Logical`, a partition of that type will be created, but if the partition type is `Pri/Log`, you will be prompted for the type you want to create. Be aware that there are only four slots available for primary partitions and because there can be only one extended partition that contains all of the logical drives, all of the logical drives must be contiguous (with no intervening primary partition). `cfdisk` next prompts you for the size of the partition you want to create. The default size, equal to the entire free space of the current partition, is displayed in megabytes. You can either press the Enter key to accept the default size or enter a different size at the prompt. `cfdisk` accepts size entries in megabytes (`M`) (default), kilobytes (`K`), cylinders (`d`), and sectors (`S`) when you enter the number immediately followed by `M`, `K`, `C`, or `S`. If the partition fills the free space available, the partition is created and you are returned to the main command line. Otherwise, the partition can be created at the beginning or the end of the free space, and `cfdisk` will ask you to choose where to place the partition. After the partition is created, `cfdisk` automatically adjusts the other partition's partition types if all of the primary partitions are used.

`p` Print the partition table to the screen or to a file. There are several different formats for the partition that you can choose from:

`r` Raw data format (exactly what would be written to disk).

`s` Partition table in sector order format.

`t` Partition table in raw format. The raw data format will print the sectors that would be written to disk if a `write` command is selected. First, the primary partition table is printed, followed by the partition tables associated with each logical partition. The data is printed in hex byte-by-byte with 16 bytes per line. The partition table in sector order format will print the partition table ordered by sector number. The fields, from left to right, are the number of the partition, the partition type, the first sector, the last sector, the offset from the first sector of the partition to the start of the data, the

This sequence will expect nothing and then send the string ATZ. The expected response to this is the string `OK`. When it receives `OK`, the string `ATDT5551212` dials the telephone. The expected string is `CONNECT`. If the string `CONNECT` is received, the remainder of the script is executed. However, if the modem finds a busy telephone, it sends the string `BUSY`. This causes the string to match the abort character sequence. The script then fails because it found a match to the abort string. If it received the string `NO CARRIER`, it aborts for the same reason. Either string may be received. Either string will terminate the chat script.

REPORT STRINGS

A report string is similar to the `ABORT` string. The difference is that the strings and all characters to the next control character such as a carriage return, are written to the report file.

The report strings may be used to isolate the transmission rate of the modem's connect string and return the value to the chat user. The analysis of the report string logic occurs in conjunction with the other string processing such as looking for the expect string. The use of the same string for a report and abort sequence is probably not very useful; however, it is possible.

The report strings do not change the completion code of the program.

These "report" strings may be specified in the script using the `REPORT` sequence. It's written in the script as in the following example:

```
REPORT CONNECT ABORT BUSY " ATDT5551212 CONNECT " ogin: account
```

This sequence expects nothing and then sends the string `ATDT5551212` to dial the telephone. The expected string is `CONNECT`. If the string `CONNECT` is received, the remainder of the script is executed. In addition, the program writes to the expect-file the string `CONNECT` plus any characters that follow it such as the connection rate.

TIME-OUT

The initial time-out value is `45` seconds. This may be changed using the `-t` parameter.

To change the time-out value for the next expect string, the following example may be used:

```
ATZ OK ATDT5551212 CONNECT TIMEOUT 10 ogin:-ogin: TIMEOUT 5 password:: hello2u2
```

This changes the time-out to `10` seconds when it expects the `login:` prompt. The time-out is then changed to `5` seconds when it looks for the password prompt.

The time-out, once changed, remains in effect until it is changed again.

SENDING EOT

The special reply string of `EOT` indicates that the chat program should send an `EOT` character to the remote. This is usually the End-of-file character sequence. A return character is not sent following the `EOT`. The `EOT` sequence may be embedded into the send string using the sequence `^D`.

GENERATING BREAK

The special reply string of `BREAK` causes a break condition to be sent. The break is a special signal on the transmitter. The normal processing on the receiver is to change the transmission rate. It may be used to cycle through the available transmission rates on the remote until you are able to receive a valid login prompt. The break sequence may be embedded into the send string using the `\K` sequence.

ESCAPE SEQUENCES

The expect and reply strings may contain escape sequences. All the sequences are legal in the reply string. Many are legal in the expect. Those that are not valid in the expect sequence are so indicated.

`''`	Expects or sends a null string. If you send a null string, it will still send the return character. This sequence may either be a pair of apostrophe or quote characters.
`\\b`	Represents a backspace character.

HISTORY

The command appeared in BSD 4.2.

BSD 4.2, 16 March 1991

crond

`crond`—`cron` daemon (Dillon's Cron).

SYNOPSIS

```
crond [-l#] [-d[#]] [-f] [-b] [-c directory]
```

OPTIONS

`crond` is a background daemon that parses individual `crontab` files and executes commands on behalf of the users in question.

`-l`*loglevel*	Set logging level; default is `8`.
`-d[`*debuglevel*`]`	Set debugging level; default is `0`. If no level is specified with the option, the default is `1`. This option also causes `crond` to run in the foreground.
`-f`	Run `crond` in the foreground.
`-b`	Run `crond` in the background (the default unless `-d` is specified).
`-c` *directory*	Specify directory containing `crontab` files.

DESCRIPTION

`crond` is responsible for scanning the `crontab` files and running their commands at the appropriate time. The `crontab` program communicates with `crond` through the `cron.update` file, which resides in the `crontabs` directory, usually `/var/spool/cron/crontabs`. This is accomplished by appending the filename of the modified or deleted `crontab` file to `cron.update`, which `crond` then picks up to resynchronize or remove its internal representation of the file.

`crond` has a number of built-in limitations to reduce the chance of it being ill-used. Potentially infinite loops during parsing are dealt with via a failsafe counter, and user `crontabs` are generally limited to 256 `crontab` entries. `crontab` lines may not be longer than 1024 characters, including the newline.

Whenever `crond` must run a job, it first creates a daemon-owned temporary file `O_EXCL` and `O_APPEND` to store any output, and then it `fork()`s and changes its user and group permissions to match that of the user the job is being run for. Then, it executes `/bin/sh -c` to run the job. The temporary file remains under the ownership of the daemon to prevent the user from tampering with it. Upon job completion, `crond` verifies the secureness of the mail file and, if it has been appended to, mails to the file to user. The `sendmail` program is run under the user's UID to prevent mail-related security holes. Unlike `crontab`, the `crond` program does not leave an open descriptor to the file for the duration of the job's execution because this might cause `crond` to run out of descriptors. When the `crontab` program allows a user to edit his `crontab`, it copies the `crontab` to a user-owned file before running the user's preferred editor. The `suid crontab` program keeps an open descriptor to the file, which it later uses to copy the file back, thereby ensuring the user has not tampered with the file type.

`crond` always synchronizes to the top of the minute, checking the current time against the list of possible jobs. The list is stored such that the scan goes very quickly, and `crond` can deal with several thousand entries without taking any noticeable amount of CPU.

AUTHOR

Matthew Dillon (`dillon@apollo.west.oic.com`)

1 May 1994

the state of the FIFO. The maximum number of characters in the FIFO when an interrupt occurred, the instantaneous count of characters in the FIFO, and how many characters are now in the FIFO are reported. This output might look like this:

```
/dev/cubC0: 830 ints, 9130 chars; fifo: 11 threshold, 11 max, 11 now
166.259866 interrupts/second, 1828.858521 characters/second
```

This output indicates that for this monitoring period, the interrupts were always being handled within one character time because `max` never rose above `threshold`. This is good, and you can probably run this way, provided that a large number of samples come out this way. You will lose characters if you overrun the FIFO because the Cyclades hardware does not seem to support the RTS RS-232 signal line for hardware flow control from the DCE to the DTE.

`cytune` will in query mode produce a summary report when ended with a `SIGINT` or when the threshold or time-out is changed.

There may be a responsiveness versus throughput tradeoff. The Cyclades card, at the higher speeds, is capable of putting a very high interrupt load on the system. This will reduce the amount of CPU time available for other tasks on your system. However, the time it takes to respond to a single character may be increased if you increase the threshold. This might be noticed by monitoring `ping`(8) times on a SLIP link controlled by a Cyclades card. If your SLIP link is generally used for interactive work such as `telnet`(1), you might want to leave the threshold low so that characters are responded to as quickly as possible. If your SLIP link is generally used for file transfer, WWW, and the like, setting the FIFO to a high value is likely to reduce the load on your system while not significantly affecting throughput. Alternatively, see the `-t` or `-T` options to adjust the time that the Cyclades waits before flushing its buffers. Units are 5ms.

If you are running a modem on a Cyclades port, it is likely that you want to maintain the threshold and time-out at a low value.

OPTIONS

`-s value`	Set the current threshold to `value` characters. Note that if the `tty` is not being held open by another process, the threshold will be reset on the next open. Only values between `1` and `12`, inclusive, are permitted.
`-t value`	Set the current flush time-out to `value` units. Note that if the `tty` is not being held open by another process, the threshold will be reset on the next open. Only values between `0` and `255`, inclusive, are permitted. Setting value to `0` forces the default, currently `0x20` (160ms) but soon to be `0x02` (10ms). Units are 5ms.
`-g`	Get the current threshold and time-out.
`-T value`	Set the default flush time-out to `value` units. When the `tty` is next opened, this value is used instead of the default. If value is `0`, then the value defaults to `0x20` (160ms), soon to be `0x02` (10ms).
`-G`	Get the default threshold and flush time-out values.
`-q`	Gather statistics about the `tty`. The results are only valid if the Cyclades driver has been compiled with `ENABLE MONITORING` defined. This is probably not the default.
`-i interval`	Statistics will be gathered every `interval` seconds.

BUGS

If you run two copies of `cytune` at the same time to report statistics about the same port, the `ints`, `chars`, and `max` values will be reset and not reported correctly. `cytune`(8) should prevent this but does not.

AUTHOR

Nick Simicich (`njs@scifi.emi.net`), with modifications by Rik Faith (`faith@cs.unc.edu`)

FILES

`/dev/ttyC[0-8]`

`/dev/cubC[0-8]`

SEE ALSO

`setserial`(8)

4 March 1995

debugfs

`debugfs`—`ext2` filesystem debugger.

SYNOPSIS

`debugfs [[-w ]device]`

DESCRIPTION

`debugfs` is a filesystem debugger. It can be used to examine and change the state of an `ext2` filesystem. `device` is the special file corresponding to the device containing the `ext2` filesystem (such as `/dev/hdXX`).

OPTIONS

`-w` Specify that the filesystem should be open in read-write mode. Without this option, the filesystem is open in read-only mode.

COMMANDS

`debugfs` is an interactive debugger. It understands a number of commands:

`cd file`

`chroot file`

`close` Close the currently open filesystem.

`clri file` Clear the contents of the inode corresponding to `file`.

`expand_dir, file` Expand a directory.

`find_free_block [goal]` Find the first free block, starting from `goal`, and allocate it.

`find_free_inode [dir [mode]]` Find a free inode and allocate it.

`freeb block` Mark the block as not allocated.

`freei file` Free the inode corresponding to `file`.

help

`iname inode` Print the filename corresponding to `inode` (currently not implemented).

`initialize device blocksize` Create an `ext2` file system on `device`.

`kill_file file` Remove a file and deallocate its blocks.

`ln source_file dest_file` Create a link.

`ls [pathname]` Emulate the `ls`(1) command.

`Modify_inode` file Modify the contents of the inode corresponding to `file`.

`mkdir file` Make a directory.

`open [-w] device` Open a filesystem.

`pwd`

`quit` Quit `debugfs`.

```
wait $ 30
if $errlvl != 0 goto error
loggedin:
# We are now logged in. Start the 'sliplogin' program,
# as this is not automatically done for me.
send sliplogin\n
wait SOME-STRING 15
# Set up the SLIP operating parameters.
get $mtu 1500
# Set Destination net/address as type 'default' (vice an address).
# This is used by the 'route' command to set the kernel routing table.
# Some machines seem to require this be done for SLIP to work properly.
default
# Say hello and fire up!
done:
print CONNECTED to $remote with address $rmtip
mode SLIP
goto exit
error:
print SLIP to $remote failed.
exit:
```

This script causes `dip` to dial up a host, log in, and get a SLIP interface channel going (in the same manner as with incoming connections). When all is set up, it simply goes into the background and waits for a hangup (or just a lethal signal), at which it hangs up and exits.

FILES

`/etc/passwd`

`/etc/diphosts`

AUTHORS

Fred N. van Kempen (`waltje@uwalt.nl.mugnet.org`), Paul Mossip (`mossip@vizlab.rutgers.edu`), Jeff Uphoff (`juphoff@aoc.nrao.edu`), Jim Seagrave (`jes@grendel.demon.co.uk`), Olaf Kirch (`okir@monad.sub.de`).

Version 3.3.7, 13 December 1993

dmesg

`dmesg`—Print or control the kernel ring buffer.

SYNOPSIS

```
dmesg [ -c ] [ -n level ]
```

DESCRIPTION

`dmesg` is used to examine or control the kernel ring buffer.

The program helps users to print their bootup messages. Instead of copying the messages by hand, the user need only

```
dmesg > boot.messages
```

and mail the `boot.messages` file to whoever can debug their problem.

There are also `-a` and `-r` options, which do nothing at all except allow you to say `fastrm -usa`, `fastrm -ussr`, or `fastrm -user`. These happen to often be convenient sets of options to use.

`fastrm` exits with a status of 0 if there were no problems or 1 if something went wrong. Attempting to remove a file that does not exist is not considered a problem. If the program exits with a nonzero status, it is probably a good idea to feed the list of files into an `xargs rm` pipeline.

fdformat

`fdformat`—Low-level formats a floppy disk.

SYNOPSIS

```
fdformat [ -n ] device
```

DESCRIPTION

`fdformat` does a low-level format on a floppy disk. `device` is usually one of the following (for floppy devices, the major is `2`, and the minor is shown for informational purposes only):

`/dev/fd0d360` (minor = 4)
`/dev/fd0h1200` (minor = 8)
`/dev/fd0D360` (minor = 12)
`/dev/fd0H360` (minor = 12)
`/dev/fd0D720` (minor = 16)
`/dev/fd0H720` (minor = 16)
`/dev/fd0h360` (minor = 20)
`/dev/fd0h720` (minor = 24)
`/dev/fd0H1440` (minor = 28)
`/dev/fd1d360` (minor = 5)
`/dev/fd1h1200` (minor = 9)
`/dev/fd1D360` (minor = 13)
`/dev/fd1H360` (minor = 13)
`/dev/fd1D720` (minor = 17)
`/dev/fd1H720` (minor = 17)
`/dev/fd1h360` (minor = 21)
`/dev/fd1h720` (minor = 25)
`/dev/fd1H1440` (minor = 29)

The generic floppy devices, `/dev/fd0` and `/dev/fd1`, will fail to work with `fdformat` when a non-standard format is being used or if the format has not been autodetected earlier. In this case, use `setfdprm`(8) to load the disk parameters.

OPTIONS

`-n` No verify. This option will disable the verification that is performed after the format.

SEE ALSO

`fd`(4), `setfdprm`(8), `mkfs`(8), `emkfs`(8)

For each line of input, `filechan` writes the initial fields, separated by whitespace and followed by a newline, to each of the files named in the filename fields. When writing to a file, `filechan` opens it in append mode and tries to lock it and change the ownership to the user and group who owns the directory where the file is being written.

By default, `filechan` writes its arguments into the directory `/news/spool/out.going`. The `-d` flag may be used to specify a directory the program should change to before starting.

If the `-p` flag is used, the program will write a line containing its process ID (in text) to the specified file.

If `filechan` is invoked with `-f 2` and given the following input:

```
news/software/b/132 <1643@munnari.oz.au>foo uunet
news/software/b/133 <102060@litchi.foo.com> uunet munnari
comp/sources/unix/2002 <999@news.foo.com>foo uunet munnari
```

Then the file `foo` will have these lines:

```
news/software/b/132 <1643@munnari.oz.au>
comp/sources/unix/2002 <999@news.foo.com>
```

The file `munnari` will have these lines:

```
news/software/b/133 <102060@litchi.foo.com>
comp/sources/unix/2002 <999@news.foo.com>
```

The file `uunet` will have these lines:

```
news/software/b/132 <1643@munnari.oz.au>
news/software/b/133 <102060@litchi.foo.com>
comp/sources/unix/2002 <999@news.foo.com>
```


Because the time window in which a file is open is very small, complicated flushing and locking protocols are not needed; a `mv`(1) followed by a `sleep`(1) for a couple of seconds is sufficient.

A map file may be specified by using the `-m` flag. Blank lines and lines starting with a number sign (`#`) are ignored. All other lines should have two hostnames separated by a colon. The first field is the name that may appear in the input stream; the second field names the file to be used when the name in the first field appears. For example, the following map file may be used to map the short names to the full domain names:

```
# This is a comment uunet:news.uu.net foo:foo.com munnari:munnari.oz.au
```

HISTORY

Written by Robert Elz (`kre@munnari.oz.au`); flags added by Rich $alz (`rsalz@uunet.uu.net`).

SEE ALSO

`buffchan`(8), `innd`(8), `newsfeeds`(5)

fsck

`fsck`—Check and repair a Linux filesystem.

SYNOPSIS

```
fsck [ -AVRTN ][-s ][-t fstype ][fs-options ] filesys [ ... ]
```

Available options:

`-d`	Debugging information is written to the syslog.
`-l`	Each FTP 1 session is logged in the syslog.
`-t`	The inactivity timeout period is set to timeout seconds. (The default is 15 minutes.)
`-T`	A client can also request a different timeout period; the maximum period allowed can be set to timeout seconds with the `-T` option. The default limit is 2 hours.

The FTP server currently supports the following FTP requests; case is not distinguished.

Request	*Description*
`ABOR`	Abort previous command
`ACCT`	Specify account (ignored)
`ALLO`	Allocate storage (vacuously)
`APPE`	Append to a file
`CDUP`	Change to parent of current working directory
`CWD`	Change working directory
`DELE`	Delete a file
`HELP`	Give help information
`LIST`	Give list files in a directory (`' ls -lA '`)
`MKD`	Make a directory
`MDTM`	Show last modification time of file
`MODE`	Specify data transfer mode
`NLST`	Give name list of files in directory
`NOOP`	Do nothing
`PASS`	Specify password
`PASV`	Prepare for server-to-server transfer
`PORT`	Specify data connection port
`PWD`	Print the current working directory
`QUIT`	Terminate session
`REST`	Restart incomplete transfer
`RETR`	Retrieve a file
`RMD`	Remove a directory
`RNFR`	Specify rename-from filename
`RNTO`	Specify rename-to filename
`SITE`	Nonstandard commands (see next section)
`SIZE`	Return size of file
`STAT`	Return status of server
`STOR`	Store a file
`STOU`	Store a file with a unique name
`STRU`	Specify data transfer structure
`SYST`	show operating system type of server system
`TYPE`	specify data transfer type
`USER`	specify username
`XCUP`	change to parent of current working directory (deprecated)

Request	*Description*
XCWD	change working directory (deprecated)
XMKD	make a directory (deprecated)
XPWD	print the current working directory (deprecated)
XRMD	remove a directory (deprecated)

The following non-standard or UNIX-specific commands are supported by the SITE request:

Request	*Description*	*Example*
UMASK	Change umask	SITE UMASK 002
IDLE	Set idle timer	SITE IDLE 60
CHMOD	Change mode of a file	SITE CHMOD 755
HELP	Give help information	SITE HELP

The remaining FTP requests specified in Internet RFC 959 are recognized but not implemented. MDTM and SIZE are not specified in RFC 959 but will appear in the next updated FTP RFC.

The FTP server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959. If a STAT command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

ftpd interprets file names according to the globbing conventions used by csh(1). This allows users to utilize the metacharacters *?[]{}~&.

ftpd authenticates users according to four rules.

The username must be in the password database and not have a null password. In this case, a password must be provided by the client before any file operations may be performed.

The username must not appear in the file (see ftpusers(5)).

The user must have a standard shell returned by getusershell(3).

If the username is anonymous or FTP, an anonymous FTP account must be present in the password file (user FTP). In this case, the user is allowed to log in by specifying any password. (By convention, this is given as the client host's name.)

In the last case, ftpd takes special measures to restrict the client's access privileges. The server performs a chroot(2) command to the home directory of the FTP user. So that system security is not breached, it is recommended that the FTP subtree be constructed with care; the following rules are recommended:

~ftp	Make the home directory owned by root and unwritable by anyone.
~ftp/bin	Make this directory owned by root and unwritable by anyone. The program ls(1) must be present to support the list command. This program should have mode 111.
~ftp/etc	Make this directory owned by root and unwritable by anyone. The files passwd(5) and group(5) must be present for the ls command to be able to produce owner names rather than numbers. The password field in passwd is not used and should not contain real encrypted passwords. These files should be mode 444 and owned by root. Don't use the system's /etc/passwd file as the password file or the system's /etc/group file as the group file in the ~ftp/etc directory.
Pa ~ftp/pub	Make this directory mode 755 and owned by root. Create a subdirectory in ~ftp/pub with the appropriate mode (777 or 733) if you want to allow normal users to upload files.

SEE ALSO

ftp(1), getusershell(3), ftpusers(5), syslogd(8)

BUGS

The anonymous account is inherently dangerous and should avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user ID of the logged-in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized but are possibly incomplete.

HISTORY

The command appeared in BSD 4.2.

BSD 4.2, 16 March 1991

ifconfig

`ifconfig`—Configure a network interface.

SYNOPSIS

```
ifconfig [interface]
ifconfig interface [aftype] options ¦ address ...
```

DESCRIPTION

`ifconfig` is used to set up (and maintain thereafter) the kernel-resident network interfaces. It is used at boot time to configure most of them to a running state. After that, it is usually only needed when debugging or when system tuning is needed.

If no arguments are given, `ifconfig` just displays the status of the currently defined interfaces. If the single interface argument is given, it displays the status of the given interface only. Otherwise, it assumes that things have to be set up.

ADDRESS FAMILIES

If the first argument after the interface name is recognized as the name of a supported address family, that address family is used for decoding and displaying all protocol addresses. Currently supported address families include `inet` (TCP/IP, default) and `ax25` (AMPR Packet Radio.)

OPTIONS

`interface`	The name of the NET interface. This usually is a name such as `wd0`, `sl3`, or something like that: a device driver name followed by a unit number.
`up`	This flag causes the interface to be activated. It is implicitly specified if the interface is given a new address (see below).
`down`	This flag causes the driver for this interface to be shut down and is useful when things start going wrong.
`[-]arp`	Enable or disable the use of the ARP protocol on this interface. If the minus (–) sign is present, the flag is turned OFF.
`[-]trailers`	Enable or disable the use of trailers on Ethernet frames. This is not used in the current implementation of NET.
`[-]allmulti`	Enable or disable the `promiscuous` mode of the interface. This means that all incoming frames get sent to the network layer of the system kernel, allowing for networking monitoring.
`metric N`	This parameter sets the interface metric. It is not used at present, but we implement it for the future.
`mtu N`	This parameter sets the Maximum Transfer Unit (MTU) of an interface. For Ethernet, this is a number in the range of 1000-2000 (default is `1500`). For SLIP, use something between 200 and 4096. Note that the current implementation does not handle IP fragmentation yet, so you'd better make the MTU large enough!
`dstaddr addr`	Set the "other end's" IP address in case of a point-to-point link, such as PPP. This keyword is obsoleted by the new `pointopoint` keyword.

DIAGNOSTICS

If `/sbin/init` finds that it is continuously respawning an entry more than ten times in two minutes, it will assume that there is an error in the command string, generate an error message on the system console, and refuse to respawn this entry until either five minutes has elapsed or it receives a signal. This prevents it from eating up system resources when someone makes a typographical error in the `/etc/inittab` file or the program for the entry is removed.

AUTHOR

Miquel van Smoorenburg (`miquels@drinkel.nl.mugnet.org`); initial manual page by Michael Haardt (`u31b3hs@pool.informatik.rwth-aachen.de`).

SEE ALSO

`getty`(1), `login`(1), `sh`(1), `who`(1), `shutdown`(1), `kill`(2), `inittab`(5), `utmp`(5)

19 January 1994

innd, inndstart

`innd`, `inndstart`—InterNetNews daemon.

SYNOPSIS

```
innd [ -a lf.c.day ][ -l ][ -f ][-i count ][-o count ][-l size]
[-m mode ][ -n flags ][ -p port ][-r ][-s ][-S host ][-t timeout ][-u ][-x ]
inndstart [ flags ]
```

DESCRIPTION

`innd`, the InterNetNews daemon, handles all incoming NNTP feeds. It reads the `active`(5), `newsfeeds`(5), and `hosts.nntp`(5) files into memory. It then opens the NNTP port to receive articles from remote sites, a UNIX-domain stream socket to receive articles from local processes such as `nnrpd`(8) and `rnews`(1), and a UNIX-domain datagram socket for use by `ctlinnd`(8) to direct the server to perform certain actions. It also opens the `history`(5) database and two log files to replace its standard output and standard error. If the `-p` flag is used, then the NNTP port is assumed to be open on the specified descriptor. (If this flag is used, then `innd` assumes it is running with the proper permissions and it does not call `chown`(2) on any files or directories it creates.)

Once the files and sockets have been opened, `innd` waits for connections and data to be ready on its ports by using `select`(2) and non-blocking I/O. If no data is available, then it flushes its in-core data structures. The default number of seconds to time out before flushing is `300`. This timeout may be changed by using the `-t` flag.

To limit the number of incoming NNTP connections, use the `-i` flag. A value of `0` suppresses this check.

To limit the number of files that are kept open for outgoing file feeds, use the `-o` flag. The default is the number of available descriptors minus some reserved for internal use.

To limit the size of an article, use the `-l` flag. If this flag is used, then any article bigger than `size` bytes is rejected. The default is no checking, which can also be obtained by using a value of `0`.

`innd` rejects articles that are too old. Although this behavior can be controlled by the history database, occasionally a site dumps a batch of very old news back onto the network. Use the `-c` flag to specify a cutoff. For example, `-c21` rejects any articles that were posted more than 21 days ago. A value of `0` suppresses this check.

`innd` usually puts itself into the background, sets its standard output and error to log files, and disassociates itself from the terminal. Using the `-d` flag instructs the server to not do this, whereas using the `-f` flag just leaves the server running the foreground. The logs are usually buffered; use the `-u` flag to have them unbuffered.

To start the server in a paused or throttled state (see `ctlinnd`(8)) use the `-m` flag to set the initial running mode. The argument should start with a single letter `g`, `p`, or `t` to emulate the `go`, `pause`, or `throttle` commands.

BUGS

All generic options must precede and not be combined with filesystem-specific options. Some filesystem-specific programs do not support the `-v` (verbose) option nor return meaningful exit codes. Also, some filesystem-specific programs do not automatically detect the device size and require the blocks parameter to be specified.

AUTHORS

David Engel (`david@ods.com`), Fred N. van Kempen (`waltje@uwalt.nl.mugnet.org`), and Ron Sommeling (`sommel@sci.kun.nl`). The manual page was shamelessly adapted from Remy Card's version for the `ext2` filesystem.

SEE ALSO

`fsck`(8), `mkfs.minix`(8), `mkfs.ext`(8), `mkfs.ext2`(8), `mkfs.xiafs`(8)

Version 1.9, Jun 19[illegible]

mkfs

`mkfs`—Make a Linux MINIX filesystem.

SYNOPSIS

```
mkfs [ -c ] [ -nnamelength ] [ -i inodecount ] device size-in-blocks
mkfs [ -l filename ] device size-in-blocks
```

DESCRIPTION

`mkfs` creates a Linux MINIX filesystem on a device (usually a disk partition).

The device is usually of the following form:

```
/dev/hda[1-8]
/dev/hdb[1-8]
/dev/sda[1-8]
/dev/sdb[1-8]
```

The `size-in-blocks` parameter is the desired size of the filesystem in blocks. This information can be determined from the `fdisk`(8) program. Only block counts strictly greater than 10 and strictly less than 65,536 are allowed.

OPTIONS

`-c`	Check the device for bad blocks before creating the filesystem. If any are found, the count is printed.
`-nnamelength`	Specify the maximum length of filenames. No space is allowed between the `-n` and the `namelength`. Currently, the only allowable values are `14` and `30`. `30` is the default.
`-i inodecount`	Specify the number of inodes for the filesystem.
`-l filename`	Read the bad blocks list from `filename`. The file has one bad block number per line. The count of bad blocks read is printed.

EXIT CODES

The exit code returned by `mkfs.minix` is one of the following:

`0`	No errors
`8`	Operational error
`16`	Usage or syntax error

SEE ALSO

`fsck`(8), `mkefs`(8), `efsck`(8), `reboot`(8)

AUTHOR

Linus Torvalds (torvalds@cs.helsinki.fi). Error code values by Rik Faith (faith@cs.unc.edu). Inode request feature by Scott Heavner (sdh@po.cwru.edu). Support for the filesystem valid flag by Dr. Wettstein (greg%wind.uucp@plains.nodak.edu).

Check to prevent mkfs of mounted filesystem and boot sector clearing by Daniel Quinlan (quinlan@yggdrasil.com).

Linux 0.99, 10 January 1994

mklost+found

mklost+found—Create a lost+found directory on a mounted Linux second extended filesystem.

SYNOPSIS

```
mklost+found
```

DESCRIPTION

mklost+found is used to create a lost+found directory in the current working directory on a Linux second extended filesystem. mklost+found pre-allocates disk blocks to the directory to make it usable by e2fsck.

OPTIONS

There are none.

AUTHOR

mklost+found was written by Remy Card (card@masi.ibp.fr), the developer and maintainer of the ext2 fs.

BUGS

There are none. :-)

AVAILABILITY

mklost+found is available for anonymous FTP from ftp.ibp.fr and tsx-11.mit.edu in /pub/linux/packages/ext2fs.

SEE ALSO

e2fsck(8), mke2fs(8)

Version 0.5b, November 1994

mkswap

mkswap—Set up a Linux swap area.

SYNOPSIS

```
mkswap [ -c ] device [size-in-blocks]
```

DESCRIPTION

mkswap sets up a Linux swap area on a device or in a file.

The device is usually of the following form:

```
/dev/hda[1-8]
/dev/hdb[1-8]
/dev/sda[1-8]
/dev/sdb[1-8]
```

DESCRIPTION

The `mount` command calls the `mount`(2) system call to prepare and graft a special device onto the filesystem tree at the point node. If either `special` or `node` are not provided, the appropriate information is taken from the `fstab`(5) file. The special keyword `none` can be used instead of a path or node specification. This is useful when mounting the `proc` filesystem.

The system maintains a list of currently mounted filesystems. If no arguments are given to `mount`, this list is printed.

Options available for the `mount` command:

`-f`	Causes everything to be done except for the actual system call; if it's not obvious, this "fakes" mounting the filesystem. This option is useful in conjunction with the `-v` flag to determine what the `mount` command is trying to do.
`-o`	Options are specified with a `-o` flag followed by a comma-separated string of options. Note that many of these options are only useful when they appear in the `/etc/fstab` file. The following options apply to any filesystem that is being mounted:
`async`	All I/O to the filesystem should be done asynchronously.
`auto`	Can be mounted with the `-a` option.
`defaults`	Use default options: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, and `async`.
`dev`	Interpret character or block special devices on the filesystem.
`exec`	Permit execution of binaries.
`noauto`	Can only be mounted explicitly (that is, the `-a` option does not cause the filesystem to be mounted).
`nodev`	Do not interpret character or block special devices on the filesystem. This option is useful for a server that has filesystems containing special devices for architectures other than its own.
`noexec`	Do not allow execution of any binaries on the mounted filesystem. This option is useful for a server that has filesystems containing binaries for architectures other than its own.
`nosuid`	Do not allow set-user-identifier or set-group-identifier bits to take effect.
`nouser`	Forbid an ordinary (that is, non-root) user to mount the filesystem.
`remount`	Attempt to remount an already-mounted filesystem. This is commonly used to change the mount flags for a filesystem, especially to make a read-only filesystem writable.
`ro`	Mount the filesystem read-only.
`rw`	Mount the filesystem read-write.
`suid`	Allow set-user-identifier or set-group-identifier bits to take effect.
`sync`	All I/O to the filesystem should be done synchronously.
`user`	Allow an ordinary user to mount the filesystem. Ordinary users always have the following options activated: `noexec`, `nosuid`, and `nodev` (unless overridden by the super-user by using, for example, the following option line: `user,exec,dev,suid`.

The following options apply only to certain filesystems:

`case=value`	For the `hpfs` filesystem, specify case as `lower` or `asis`.
`check=value`	Tells the `ext2` filesystem kernel code to do some more checks while the filesystem is mounted. Currently (0.99.15), the following values can be specified with this option:
`none`	No extra check is performed by the kernel code.
`normal`	The inodes and blocks bitmaps are checked when the filesystem is mounted (this is the default).
`strict`	In addition to the normal checks, block deallocation checks that the block to free is in the data zone.
`check=value`	For the `msdos` filesystem, three different levels of specificity can be chosen:

DESCRIPTION

This command allows the nameserver administrator to send various signals to the nameserver or to restart it. Zero or more directives may be given from the following list:

`status`	Displays the current status of named as shown by `ps`(1).
`dumpdb`	Causes `named` to dump its database and cache to `/var/tmp/named_dump.db` (uses the `INT` signal.)
`reload`	Causes `named` to check the serial numbers of all primary and secondary zones and to reload those that have changed (uses the `HUP` signal.)
`stats`	Causes `named` to dump its statistics to `/var/tmp/named.stats` (uses the `IOT` or `ABRT` signal.)
`trace`	Causes `named` to increment its "tracing level" by one. Whenever the tracing level is nonzero, trace information is written to `/var/tmp/named.run`. Higher tracing levels result in more detailed information (uses the `USR1` signal).
`notrace`	Causes `named` to set its "tracing level" to zero, closing `/var/tmp/named.run` if it is open (uses the `USR2` signal).
`querylog`	Causes `named` to toggle the "query logging" feature, which results in a `syslog`(3) of each incoming query (uses the `WINCH` signal). Note that query logging consumes quite a lot of log file space. This directive may also be given as `qrylog`.
`start`	Causes `named` to be started as long as it isn't already running.
`stop`	Causes `named` to be stopped if it is running.
`restart`	Causes `named` to be killed and restarted.

BUGS

Arguments to `named` are not preserved by `restart` or known by `start`. Some mechanism for controlling the parameters and environment should exist.

Implemented as a `sh`(1) script.

AUTHOR

Paul Vixie (Internet Software Consortium)

SEE ALSO

`named`(8), `named.reload`(8), `named.restart`(8)

27 November 1994

netstat

`netstat`—Display active network connections

SYNOPSIS

```
netstat [[-a ¦ [-t ¦ -u ¦ -w]] [-n ¦ -o] ¦ -x] [-c]
netstat -r [-c] [-n]
netstat -v
```

DESCRIPTION

`netstat` displays the status of network connections on either TCP, UDP, or RAW sockets to the system. By default, `netstat` only displays status on those TCP sockets that are not in the `LISTEN` state (that is, connections to active processes). To obtain information about the kernel routing table, `netstat` may be invoked with the option `-r`. A listing of internal UNIX connections can be obtained by invoking netstat with the option `-x`.

`LISTENING`	The socket is listening for a connection request.
`UNCONNECTED`	The socket is not connected to another one.
`CONNECTING`	The socket is about to establish a connection.
`CONNECTED`	The socket is connected.
`DISCONNECTING`	The socket is disconnecting.
`UNKNOWN`	This state should never happen.
`Path`	This displays the pathname that the corresponding processes attached to the socket.

The network routing table (invoked with `netstat -r`) shows up the following information:

Destination net/address	The destination address of a resolved host or hand-entered network is displayed. Unless the option `-n` is given, the hosts or nets are resolved. An entry named `default` shows up the default route for the kernel.
Gateway address	If there is no asterisk (*) displayed, any data is routed to the dedicated gateway.
Flags	Possible routing flags are
	`U` This route is usable.
	`G` Destination is a gateway.
	`H` Destination is a host entry.
	`N` Destination is a Net entry.
	`R` Route will be reinstated after timeout.
	`D` This one is created dynamically (by redirection).
	`M` This one is modified dynamically (by redirection).
RefCnt	Reference count for this route.
Use	How many times this route was used yet.
Iface	This is the name of the interface where this route belongs.

OPTIONS

`-a`	Display information about all Internet sockets, such as TCP, UDP, and RAW, including those sockets that are listening only.
`-c`	Generate a continuous listing of network status: network status is displayed every second until the program is interrupted.
`-n`	Causes `netstat` not to resolve hostnames and service names when displaying remote and local address and port information.
`-o`	Display timer states, expiration times, and backoff state.
`-r`	Display kernel routing table.
`-t`	Display information about TCP sockets only, including those that are listening.
`-u`	Display information about UDP sockets only.
`-v`	Print version information.
`-w`	Display information about raw sockets.
`-x`	Display information about UNIX domain sockets.

FILES

`/proc/net/tcp`	TCP socket information
`/proc/net/udp`	UDP socket information
`/proc/net/raw`	RAW socket information
`/proc/net/unix`	UNIX domain socket information

`nostat`	This keyword disables the status report generated by innstat (see `newslog`(8)). Without this keyword, the status report is the first function performed, just prior to obtaining the `news.daily` lock.
`noexpire`	By default, `expire` is invoked to remove old news articles. Using this keyword disables this function.
`noexplog`	`expire` usually appends information to `/var/log/news/expire.log` (see `newslog`(5)). Using this keyword causes the `expire` output to be handled as part of `news.daily`'s output. It has no effect if the `noexpire` keyword is used.
`flags='expire\args'`	By default, `expire` is invoked with the an argument of `-v1`. Using this keyword changes the arguments to those specified. Be careful to use quotes if multiple arguments are needed. This keyword has no effect if the `noexpire` keyword is used.
`nologs`	After expiration, `scanlogs`(8) is invoked to process the log files. Using this keyword disables all log processing functions.
`norotate`	By default, log processing includes rotating and cleaning out log files. Using this keyword disables the rotating and cleaning aspect of the log processing. The log files are only scanned for information and no log files are altered. This keyword has no effect if the `nologs` keyword is used.
`norenumber`	This keyword disables the `ctlinnd`(8) renumber operation. Usually, the low watermark for all newsgroups (see `active`(5)) is reset.
`norm`	By default, any socket `ctlinnd` socket that has not been modified for two days is removed. Using this keyword disables this function.
`nomail`	`news.daily` usually sends a mail message containing the results to the administrator. Using this keyword causes this message to be sent to `stdout` and `stderr` instead. Usually, all utilities invoked by the script have their `stdout` and `stderr` redirected into a file. If the file is empty, no message is sent.
`expireover`	The `expireover` program is called after expiration to purge the overview databases.
`expireoverflags='expireovernargs'`	If the `expireover` keyword is used, this keyword may be used to specify the flags to be passed to `expireover`. If the `delayrm` keyword is used, then the default value is `-z` and the list of deleted files; otherwise, the default value is `-s`.
`/full/path`	The program specified by the given path is executed just before any expiration is done. A typical use is to specify an alternate expiration program and use the `noexpire` keyword. Multiple programs may be specified; they are invoked in order.

The `norotate` keyword is passed on to `scanlogs` if it is invoked. `expirerm` is a script that removes a list of files. The specified file lists the files. It is sorted and then fed into a pipeline responsible for doing the removal, usually `fastrm`(8). If there seemed to be a problem removing the files, then mail is sent to the news administrator. If there were no problems, then file is renamed to `/var/log/news/expire.list` where it is kept (for safety) until the next day's expiration.

`innwatch` is a script that can be started at news boot time. It periodically—every `sleeptime` seconds— examines the load average and the number of free blocks and inodes on the spool partition, as described by its control file, `innwatch.ctl`(5). If the load gets too high or the disk gets too full, it throttles the server. When the condition restores, it unblocks the server. In addition, on each pass through the loop, it checks the specified log file to see if it has been modified and sends mail to the news administrator if so. It is usually a good idea to set this to the `syslog`(3) file that receives critical news messages. Upon receipt of an interrupt signal, `innwatch` reports its status in the file `/news/lib/innwatch.status`.

`innccheck` is a `perl`(1) script that verifies the syntax and permissions of all InterNetNews configuration files. If no files are specified, it checks all files. A filename may be followed by an equal sign and a path to indicate the pathname to use for the file. For example, `newsfeeds=/tmp/nf` checks the syntax of a new `newsfeeds`(5) without requiring it to be installed. If the `-v` flag is used, it prints status information as it checks each file. If the `-pedantic` flag is used, it checks the files for omissions that are not strictly errors but might indicate a configuration error.

If any file is specified, only the permissions on those files are checked. The `-noperms` flag suppresses this check. If the `-perms` flag is used, the script verifies the ownership and permissions of all files. The `-fix` flag can also be used so that the output can be executed as a shell script.

HISTORY

Written by Rich $alz (rsalz@uunet.uu.net) for InterNetNews. Overview support added by Rob Robertston (rob@violet.berkeley.edu) and Rich in January 1993.

SEE ALSO

ctlinnd(8), innd(8), inn.conf(5), nnrp.access(5), signal(2), wildmat(3)

nntpsend

nntpsend—Send Usenet articles to remote site.

SYNOPSIS

```
nntpsend [ -d ][-p ][-r ][-S ][-s size ][-t timeout ]
[-T timelimit ][sitename fqdn ] ...
```

DESCRIPTION

nntpsend is a front end that invokes innxmit(1) to send Usenet articles to a remote NNTP site.

The sites to be fed may be specified by giving sitename fqdn pairs on the command line. If no such pairs are given, nntpsend defaults to the information given in the nntpsend.ctl(5) configuration file.

The sitename should be the name of the site as specified in the newsfeeds(5) file. The fqdn should be the hostname or IP address of the remote site. An innxmit is launched for sites with queued news. All innxmit processes are spawned in the background and the script waits for them all to finish before returning. Output is sent to the file /var/log/news/nntpsend.log. To avoid overwhelming the local system, nntpsend waits five seconds before spawning each child. The flag -a is always given as a flag to innxmit.

nntpsend expects that the batchfile for a site is named /news/spool/out.going/sitename. To prevent batchfile corruption, shlock(1) is used to "lock" these files.

The -p, -r, -S, -t, and -T flags are passed on to the child innxmit program. Note that if the -p flag is used then no connection is made and no articles are fed to the remote site. It is useful to have cron(8) invoke nntpsend with this flag in case a site cannot be reached for an extended period of time.

If the -s flag is used, then shrinkfile(1) is invoked to perform a tail truncation on the batchfile and the flag is passed to it.

When sitename fqdn pairs are given on the command line, any flags given on the command completely describe how innxmit and shrinkfile operate. When no such pairs are given on the command line, then the information found in nntpsend.ctl becomes the default flags for that site. Any flags given on the command line override the default flags for the site.

For example, with the following control file:

```
nsavax:erehwon.nsavax.gov::-S -t60
group70:group70.org::
walldrug:walldrug.com:1m:-T1800 -t300
```

The command

```
nntpsend
```

will result in the following:

```
Sitename   Truncation   Innxmit flags
nsavax     (none)       -a -S -t60
group70    (none)       -a -t180
walldrug   1m           -a -T1800 -t300
```

	HESIOD	The MIT Athena Hesiod class.
	ANY	Wildcard (any of the above).
	The class specifies the protocol group of the information. (Default = IN, abbreviation = cl.)	
[no]debug	Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer. (Default = nodebug, abbreviation = [no]deb.)	
[no]d2	Turn exhaustive debugging mode on. Essentially, all fields of every packet are printed. (Default = nod2.)	
domain=name	Change the default domain name to name. The default domain name is appended to a lookup request depending on the state of the defname and search options. The domain search list contains the parents of the default domain if it has at least two components in its name. For example, if the default domain is CC.Berkeley.EDU, the search list is CC.Berkeley.EDU and Berkeley.EDU. Use the set srchlist command to specify a different list. Use the set all command to display the list. (Default = value from hostname, /etc/resolv.conf, or LOCALDO-MAIN, abbreviation = do.)	
srchlist=name1/name2/...	Change the default domain name to name1 and the domain search list to name1, name2, and so on. A maximum of six names separated by slashes (/) can be specified. For example, set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU sets the domain to lcs.MIT.EDU and the search list to the three names. This command overrides the default domain name and search list of the set domain command. Use the set all command to display the list. (Default = value based on hostname, /etc/resolv.conf, or LOCAL-DOMAIN, abbreviation = srch.)	
[no]defname	If set, append the default domain name to a single-component lookup request (that is, one that does not contain a period). (Default = defname, abbreviation = [no]def.)	
[no]search	If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received. (Default = search, abbreviation = [no]sea.)	
port=value	Change the default TCP/UDP nameserver port to value. (Default = 53, abbreviation = po.)	
querytype=value, type=value	Change the type of information query to one of the following:	
	A	The host's Internet address.
	CNAME	The canonical name for an alias.
	HINFO	The host CPU and operating system type.
	MINFO	The mailbox or mail list information.
	MX	The mail exchanger.
	NS	The nameserver for the named zone.
	PTR	The host name if the query is an Internet address; otherwise, the pointer to other information.
	SOA	The domain's "start-of-authority" information.
	TXT	The text information.
	UINFO	The user information.
	WKS	The supported well-known services.
	Other types (ANY, AXFR, MB, MD, MF, NULL) are described in the RFC-1035 document. (Default = A, abbreviations = q, ty.)	
[no]recurse	Tell the nameserver to query other servers if it does not have the information. (Default = recurse, abbreviation = [no]rec.)	
retry=number	Set the number of retries to number. When a reply to a request is not received within a certain amount of time (changed with set timeout), the timeout period is doubled and the request is resent. The retry value controls how many times a request is resent before giving up. (Default = 4, abbreviation = ret.)	

`portmap` must be started before any RPC servers are invoked.

Usually, `portmap` forks and dissociates itself from the terminal like any other daemon. Portmap then logs errors using `syslog`(3).

Option available:

`-d` (debug) prevents `portmap` from running as a daemon and causes errors and debugging information to be printed to the standard error output.

SEE ALSO

`inetd.conf`(5), `rpcinfo`(8), `inetd`(8)

BUGS

If `portmap` crashes, all servers must be restarted.

HISTORY

The `portmap` command appeared in BSD 4.3.

BSD 4.3, 16 March 1991

powerd

`powerd`—Monitor a serial line connected to a UPS

SYNOPSIS

```
/etc/powerd serial-device
```

DESCRIPTION

`powerd` is a daemon process that sits in the background and monitors the state of the DCD line of the serial device. It is meant that this line is connected to a UPS (Uninterruptible Power Supply) so that it knows about the state of the UPS. As soon as `powerd` senses that the power is failing (it sees that DCD goes low) it notifies `init`(8) and `init` executes the `powerwait` and `powerfail` entries. If `powerd` senses that the power has been restored, it notifies `init` again and `init` executes the `powerokwait` entries.

ARGUMENTS

`serial-device`	Some serial port that is not being used by some other device and does not share an interrupt with any other serial port.

DIAGNOSTICS

`powerd` regularly checks the DSR line to see if it's high. DSR should be directly connected to DTR and `powerd` keeps that line high, so if DSR is low, something is wrong with the connection. `powerd` notifies you about this fact every two minutes. When it sees that the connection is restored, it will say so.

IMPLEMENTATION

It's pretty simple to connect your UPS to the Linux machine. The steps are easy:

1. Make sure you have an UPS with a simple relais output: it should close its connections (make) if the power is gone, and it should open its connections (break) if the power is good.
2. Buy a serial plug. Connect the DTR line to the DSR line directly. Connect the DTR line and the DCD line with a 10 kilo ohm resistor. Connect the relais-output of the UPS to GROUND and the DCD line. If you don't know what pins DSR, DTR, DCD and GROUND are, you can always ask at the store where you bought the plug.
3. You're all set.

`bsdcomp nr,nt`	Request that the peer compress packets that it sends, using the BSD-Compress scheme, with a maximum code size of `nr` bits and agree to compress packets sent to the peer with a maximum code size of `nt` bits. If `nt` is not specified, it defaults to the value given for `nr`. Values in the range `9` to `15` may be used for `nr` and `nt`; larger values give better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of `0` for `nr` or `nt` disables compression in the corresponding direction.
`-bsdcomp`	Disables compression; `pppd` does not request or agree to compress packets using the BSD-Compress scheme.
`+chap`	Require the peer to authenticate itself using CHAP (Cryptographic Handshake Authentication Protocol) authentication.
`-chap`	Don't agree to authenticate using CHAP.
`chap-interval n`	If this option is given, `pppd` rechallenges the peer every `n` seconds.
`chap-max-challenge n`	Set the maximum number of CHAP challenge transmissions to *u* (default is `10`).
`chap-restart n`	Set the CHAP restart interval (retransmission timeout for challenges) to `n` seconds (default is `3`).
`-crtscts`	Disable hardware flow control (RTS/CTS) on the serial port. If neither the `crtscts` nor the `-crtscts` option is given, the hardware flow control setting for the serial port is left unchanged.
`-d`	Increase debugging level (same as the debug option).
`debug`	Increase debugging level (same as `-d`). If this option is given, `pppd` logs the contents of all control packets sent or received in a readable form. The packets are logged through `syslog` with facility daemon and level debug. This information can be directed to a file by setting up `/etc/syslog.conf` appropriately (see `syslog.conf`(5)).
`-defaultroute`	Disable the `defaultroute` option. The system administrator who wants to prevent users from creating default routes with `pppd` can do so by placing this option in the `/etc/ppp/options` file.
`-detach`	Don't fork to become a background process (otherwise, `pppd` will do so if a serial device other than its controlling terminal is specified).
`dns-addr n`	This option sets the IP address or addresses for the Domain Name Server. It is used by Microsoft Windows clients. The primary DNS address is specified by the first instance of the `dns-addr` option. The secondary is specified by the second instance.
`domain d`	Append the domain name `d` to the local hostname for authentication purposes. For example, if `gethost-name()` returns the name `porsche`, but the fully qualified domain name is `porsche.Quotron.COM`, you use the `domain` option to set the domain name to `Quotron.COM`.
`-ip`	Disable IP address negotiation. If this option is used, the remote IP address must be specified with an option on the command line or in an options file.
`+ip-protocol`	Enable the IPCP and IP protocols. This is the default condition. This option is only needed if the default setting is `-ip-protocol`.
`-ip-protocol`	Disable the IPCP and IP protocols. This should only be used if you know you are using a client that only understands IPX and you don't want to confuse the client with the IPCP protocol.
`+ipx-protocol`	Enable the IPXCP and IPX protocols. This is the default condition if your kernel supports IPX. This option is only needed if the default setting is `-ipx-protocol`. If your kernel does not support IPX, this option has no effect.

```
mv history.n history
mv history.n.pag history.pag
mv history.n.dir history.dir
else
echo 'Problem rebuilding history; old file not replaced'
fi
ctlinnd go "Rebuilding history database"
```

Note that this keeps no record of expired articles.

HISTORY

Written by Rich $alz (`rsalz@uunet.uu.net`) for InterNetNews.

SEE ALSO

`dbz`(3), `history`(5), `innd`(8)

quotacheck

`quotacheck`—Scan a filesystem for disk usages

SYNOPSIS

```
quotacheck [-g] [-u] [-v] -a
quotacheck [-g] [-u] [-v] filesys ...
```


DESCRIPTION

`quotacheck` performs a filesystem scan for usage of files and directories, used by either user or group. The output is the quota file for the corresponding filesystem. By default, the names for these files are

A user scan	`quota.user`
A group scan	`quota.group`

The resulting file consists of a `struct dqblk` for each possible ID up to the highest existing UID or GID and contains the values for the disk file and block usage and possibly excess time for these values. (For definitions of `struct dqblk`, see `linux/quota.h`.)

`quotacheck` should be run each time the system boots and mounts non-valid filesystems. This is most likely to happen after a system crash.

The speed of the scan decreases with the amount of directories increasing. The time needed doubles when disk usage is doubled as well. A 100MB partition used for 94 percent is scanned in one minute; the same partition used for 50 percent is done in 25 seconds.

OPTIONS

`-v`	This way, the program will give some useful information about what it is doing, plus some fancy stuff.
`-d`	This means debug. It will result in a lot of information that can be used in debugging the program. The output is very verbose and the scan will not be fast.
`-u`	This flag tells the program to scan the disk and to count the files and directories used by a certain UID. This is the default action.
`-g`	This flag forces the program to count the files and directories used by a certain GID.

NOTE

`checkquota` should only be run as superuser. Non-privileged users are presumably not allowed to read all the directories on the given filesystem.

HISTORY

The `renice` command appeared in BSD 4.0.

BSD 4, 9 June 1993

repquota

`repquota`—Summarize quotas for a filesystem.

SYNOPSIS

```
/usr/etc/repquota [ -vug ] filesystem...
/usr/etc/repquota [ -avug ]
```

DESCRIPTION

`repquota` prints a summary of the disk usage and quotas for the specified filesystems. For each user, the current number of files and amount of space (in kilobytes) is printed, along with any quotas created with `edquota`(8).

OPTIONS

`-a`	Report on all filesystems indicated in `/etc/fstab` to be read-write with quotas.
`-v`	Report all quotas even if there is no usage.
`-g`	Report quotas for groups.
`-u`	Report quotas for users. This is the default.

Only the superuser can view quotas that are not their own.

FILES

`quotas`	Quota file at the filesystem root
`/etc/fstab`	Default filesystems

SEE ALSO

`quota`(1), `quotactl`(2), `edquota`(8), `quotacheck`(8), `quotaon`(8)

8 June 1993

rexecd

`rexecd`—Remote execution server.

SYNOPSIS

```
rexecd
```

DESCRIPTION

`rexecd` is the server for the `rexec`(3) routine. The server provides remote execution facilities with authentication based on usernames and passwords.

`rexecd` listens for service requests at the port indicated in the `exec` service specification; see `services`(5). When a service request is received, the following protocol is initiated:

1. The server reads characters from the socket up to a null `\0` byte. The resultant string is interpreted as an ASCII number, base 10.

When a request packet is received, `routed` formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric relative to the sender.

Response packets received by `routed` are used to update the routing tables if one of the following conditions is satisfied:

No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (the hop count is not infinite).

The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, `routed` records the change in its internal tables and updates the kernel routing table. The change is reflected in the next response packet sent.

In addition to processing incoming packets, `routed` also periodically checks the routing table entries. If an entry has not been updated for three minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to ensure the invalidation is propagated throughout the local Internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The receipt of routing responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

Options supported by `routed`:

`-d`	Enable additional debugging information to be logged, such as bad packets received.
`-g`	This flag is used on internetwork routers to offer a route to the "default" destination. This is typically used on a gateway to the Internet or on a gateway that uses another routing protocol whose routes are not reported to other local routers.
`-s`	Supplying this option forces `routed` to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present or if a point-to-point link is in use.
`-q`	This is the opposite of the `-s` option.
`-t`	If the `-t` option is specified, all packets sent or received are printed on the standard output. In addition, `routed` will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process.

Any other argument supplied is interpreted as the name of file in which `routed`'s actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received that are related to the changed route.

In addition to the facilities described previously, `routed` supports the notion of "distant" passive and active gateways. When `routed` is started, it reads the file to find gateways that might not be located using only information from the `SIOGIFCONF`ioctl (2). Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, whereas gateways marked active should be willing to exchange routing information (that is, they should have a `routed` process running on the machine). Routes through passive gateways are installed in the kernel's routing tables once upon startup. Such routes are not included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway, and if no routing information is received for a period of the time, the associated route is deleted. Gateways marked external are also passive but are not placed in the kernel routing table

`auto_irq` — During autoconfiguration, try to determine the IRQ. This feature is not guaranteed to always produce the correct result; some hardware configurations will fool the Linux kernel. It is generally safer not to use the `auto_irq` feature but rather to specify the IRQ to be used explicitly, using the `irq` parameter.

`^auto_irq` — During autoconfiguration, do not try to determine the IRQ.

`skip_test` — During autoconfiguration, skip the UART test. Some internal modems do not have National Semiconductor compatible UARTs but have cheap imitations instead. Some of these cheesy imitation UARTs do not fully support the loopback detection mode, which is used by the kernel to make sure there really is a UART at a particular address before attempting to configure it. For certain internal modems, you will need to specify this parameter so Linux can initialize the UART correctly.

`^skip_test` — During autoconfiguration, do not skip the UART test.

`baud_base baud_base` — This option sets the base baud rate, which is the clock frequency divided by 16. Usually, this value is `115200`, which is also the fastest baud rate, which the UART can support.

`spd_hi` — Use 57.6KB when the application requests 38.4KB. This parameter can be specified by a non-privileged user.

`spd_vhi` — Use 115KB when the application requests 38.4KB. This parameter can be specified by a non-privileged user.

`spd_cust` — Use the custom divisor to set the speed when the application requests 38.4KB. In this case the baud rate is the `base_baud` divided by the `divisor`. This parameter can be specified by a non-privileged user.

`spd_normal` — Use 38.4KB when the application requests 38.4KB. This parameter can be specified by a non-privileged user.

`divisor divisor` — This option sets the custom divisor. This divisor will be used when the `spd_cust` option is selected and the serial port is set to 38.4KB by the application. This parameter can be specified by a non-privileged user.

`sak` — Set the break key at the Secure Attention Key.

`^sak` — Disable the Secure Attention Key.

`fourport` — Configure the port as an AST Fourport card.

`^fourport` — Disable AST Fourport configuration.

`close_delay delay` — Specify the amount of time, in hundredths of a second, that DTR should remain low on a serial line after the callout device is closed before the blocked dial-in device raises DTR again. The default value of this option is `50`, or a half-second delay.

`Session_lockout` — Lock out callout port (`/dev/cuaXX`) accesses across different sessions. That is, once a process has opened a port, do not allow a process with a different session ID to open that port until the first process has closed it.

`^session_lockout` — Do not lock out callout port accesses across different sessions.

`pgrp_lockout` — Lock out callout port (`/dev/cuaXX`) accesses across different process groups. That is, once a process has opened a port, do not allow a process in a different process group to open that port until the first process has closed it.

`^pgrp_lockout` — Do not lock out callout port accesses across different process groups.

`hup_notify` — Notify a process blocked on opening a dial-in line when a process has finished using a callout line (either by closing it or by the serial line being hung up) by returning `EAGAIN` to the open.

The application of this parameter is for `gettys` that are blocked on a serial port's dial-in line. This allows the `getty` to reset the modem (which may have had its configuration modified by the application using the callout device) before blocking on the open again.

`^hup_notify` — Do not notify a process blocked on opening a dial-in line when the callout device is hung up.

BUGS

This program is called `simpleinit` to distinguish it from the System V compatible versions of `init` that are starting to appear in the Linux community. `simpleinit` should be linked to, or made identical with, `init` for correct functionality.

AUTHOR

Peter Orbaek (`poe@daimi.aau.dk`), version 1.20, with patches for single-user mode by Werner Almesberger.

Linux 0.99, 20 November 1993

slattach

`slattach`—Attach a network interface to a serial line.

SYNOPSIS

```
slattach [-v] [-p proto] [-s speed] [tty]
```

DESCRIPTION

`slattach` is a little program that can be used to put a normal terminal ("serial") line into one of several "network" modes, thus allowing you to use it for point-to-point links to other computers.

OPTIONS

`[-v]` Enable debugging output. Useful when determining why a given setup doesn't work.

`[-p proto]` Set a specific kind of protocol to use on the line. The default is set to `cslip`, compressed SLIP. Other possible values are `slip` (normal SLIP), `ppp` (Point-to-Point Protocol), and `kiss` (AX.25 TNC protocol).

`[-s speed]` Set a specific line speed other than the default.

If no arguments are given, the current terminal line (usually the login device) is used. Otherwise, an attempt is made to claim the indicated terminal port, lock it, and open it.

FILES

`/dev/cua*`

BUGS

None so far.

AUTHOR

Fred N. van Kempen (`waltje@uwalt.nl.mugnet.org`)

20 September 1993

sliplogin

`sliplogin`—Attach a serial line network interface.

SYNOPSIS

```
sliplogin [loginname]
```

DESCRIPTION

`sliplogin` is used to turn the terminal line on standard input into a serial line IP SLIP link to a remote host. To do this, the program searches the file for an entry matching `loginname` (which defaults to the current login name if omitted). If a

This may take some acclimatization for those individuals used to the pure BSD behavior, but testers have indicated that this syntax is somewhat more flexible than the BSD behavior. Note that these changes should not affect standard `syslog.conf` files. You must specifically modify the configuration files to obtain the enhanced behavior.

SUPPORT FOR REMOTE LOGGING

These modifications provide network support to the `syslogd` facility. Network support means that messages can be forwarded from one node running `syslogd` to another node running `syslogd` where they will be actually logged to a disk file.

The strategy is to have `syslogd` listen on a UNIX domain socket for locally generated log messages. This behavior will allow `syslogd` to interoperate with the `syslog` found in the standard C library. At the same time, `syslogd` listens on the standard `syslog` port for messages forwarded from other hosts. To have this work correctly, the services files (typically found in `/usr/etc/inet`) must have the following entry:

```
syslog 514/udp
```

To cause messages to be forwarded to another host, replace the normal file line in the `syslog.conf` file with the name of the host to which the messages is to be sent prepended with an `@`.

For example, to forward all messages to a remote host, use the following `syslog.conf` entry:

```
# Sample syslogd configuration file to
# messages to a remote host forward all.
.* @hostname
```

To forward all kernel messages to a remote host, the configuration file is

```
# Sample configuration file to forward all kernel
# messages to a remote host.
kern.* @hostname
```


OUTPUT TO NAMED PIPES (FIFOS)

This version of `syslogd` has support for logging output to named pipes (FIFOs). A FIFO or named pipe can be used as a destination for log messages by prepending a | to the name of the file. This is handy for debugging. Note that the FIFO must be created with the `mkfifo` command before `syslogd` is started.

The following configuration file routes debug messages from the kernel to a FIFO:

```
# Sample configuration to route kernel debugging
# messages ONLY to /usr/adm/debug which is a
# named pipe.
kern.=debug ¦/usr/adm/debug
```

INSTALLATION CONCERNS

There is probably one important consideration when installing this version of `syslogd`. This version of `syslogd` is dependent on proper formatting of messages by the `syslog` function. The functioning of the `syslog` function in the shared libraries changed somewhere in the region of `libc.so.4.[2-4].n`. The specific change was to null-terminate the message before transmitting it to the `/dev/log` socket. Proper functioning of this version of `syslogd` is dependent on null-termination of the message.

This problem will typically manifest itself if old statically linked binaries are being used on the system. Binaries using old versions of the `syslog` function will cause empty lines to be logged, followed by the message with the first character in the message removed. Relinking these binaries to newer versions of the shared libraries will correct this problem.

SECURITY THREATS

There is the potential for the `syslogd` daemon to be used as a conduit for a denial of service attack. Thanks go to John Morrison (`jmorriso@rflab.ee.ubc.ca`) for alerting me to this potential. A rogue programmer could very easily flood the `syslogd` daemon with `syslog` messages resulting in the log files consuming all the remaining space on the filesystem.

to which it is connected, except as modified by the options. It requests synchronization service from the first master server located. If permitted by the `-M` flag, it provides synchronization service on any attached networks on which no current master server is detected. Such a server propagates the time computed by the top-level master. The `-n` flag, followed by the name of a network that the host is connected to (see `networks`(5)), overrides the default choice of the network addresses made by the program. Each time the `-n` flag appears, that network name is added to a list of valid networks. All other networks are ignored. The `-i` flag, followed by the name of a network to which the host is connected (see `networks`(5)), overrides the default choice of the network addresses made by the program. Each time the `-i` flag appears, that network name is added to a list of networks to ignore. All other networks are used by the time daemon. The `-n` and `-i` flags are meaningless if used together.

`timed` checks for a master time server on each network to which it is connected, except as modified by the `-n` and `-i` options. If it finds masters on more than one network, it chooses one network on which to be a "slave" and then periodically checks the other networks to see if the masters there have disappeared.

One way to synchronize a group of machines is to use an NTP daemon to synchronize the clock of one machine to a distant standard or a radio receiver and -F hostname to tell its `timed` daemon to trust only itself.

Messages printed by the kernel on the system console occur with interrupts disabled. This means that the clock stops while they are printing. A machine with many disk or network hardware problems and consequent messages can not keep good time by itself. Each message typically causes the clock to lose a dozen milliseconds. A time daemon can correct the result.

Messages in the system log about machines that failed to respond usually indicate machines that crashed or were turned off. Complaints about machines that failed to respond to initial time settings are often associated with "multi-homed" machines that looked for time masters on more than one network and eventually chose to become a slave on the other network.

WARNING

If two or more time daemons, whether timed, NTP, try to adjust the same clock, temporal chaos will result. If both this and another time daemon are run on the same machine, ensure that the `-F` flag is used, so that `timed` never attempts to adjust the local clock.

The protocol is based on UDP/IP broadcasts. All machines within the range of a broadcast that are using the TSP protocol must cooperate. There cannot be more than a single administrative domain using the `-F` flag among all machines reached by a broadcast packet. Failure to follow this rule is usually indicated by complaints concerning "untrusted" machines in the system log.

FILES

`/var/log/timed.log` tracing file for timed

`/var/log/timed.masterlog` log file for master timed

SEE ALSO

`date`(1), `adjtime`(2), `gettimeofday`(2), `icmp`(4), `timedc`(8), "TSP: The Time Synchronization Protocol for UNIX 4.3 BSD," R. Gusella, S. Zatti.

HISTORY

The `timed` daemon appeared in BSD 4.3.

BSD 4.3, 11 May 1993

timedc

`timedc`—Timed control program.

SYNOPSIS

```
timedc [command] [argument ...]
```

```
9  ***
10  * * *
11  * * *
12  * * *
13  rip.Berkeley.EDU (128.32.131.22)  59 ms !  39 ms !  39 ms
!
```

Notice that there are 12 “gateways” (13 is the final destination) and exactly the last half of them are “missing.” What’s really happening is that rip (a Sun-3 running Sun OS3.5) is using the `ttl` from our arriving datagram as the `ttl` in its ICMP reply. The reply will time out on the return path (with no notice sent to anyone because ICMPs aren’t sent for ICMPs) until we probe with a `ttl` that’s at least twice the path length. That is, rip is really only seven hops away. A reply that returns with a `ttl` of `1` is a clue this problem exists. `traceroute` prints a `!` after the time if the `ttl` is less than or equal to 1. Because vendors ship a lot of obsolete (DEC Ultrix, Sun 3.*x*) or non-standard HPUX software, expect to see this problem frequently or take care picking the target host of your probes. Other possible annotations after the time are `!H`, `!N`, `!P` (got a host, network, or protocol unreachable), `!S`, or `!F` (source route failed or fragmentation needed—neither of these should ever occur and the associated gateway is busted if you see one). If almost all the probes result in some kind of unreachable, `traceroute` will give up and exit.

This program is intended for use in network testing, measurement, and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use `traceroute` during normal operations or from automated scripts.

AUTHOR

Implemented by Van Jacobson from a suggestion by Steve Deering. Debugged by a cast of thousands with particularly cogent suggestions or fixes from C. Philip Wood, Tim Seaver, and Ken Adelman.

SEE ALSO

`netstat`(1), `ping`(8)

BSD 4.3, 6 June 1993

tune2fs

`tune2fs`—Adjust tunable filesystem parameters on second extended filesystems.

SYNOPSIS

```
tune2fs [ -l ][-c max-mount-counts ][-e errors-behavior ]
[-i interval-between-checks ][ -m reserved-blocks-percentage ]
[-r reserved-blocks-count ][-u user ][-g group ]device
```

DESCRIPTION

`tune2fs` adjusts tunable filesystem parameters on a Linux second extended filesystem.

Never use `tune2fs` on a read/write mounted filesystem to change parameters!

OPTIONS

`-c max-mount-counts`
: Adjust the maximal mounts count between two filesystem checks.

`-e errors-behavior`
: Change the behavior of the kernel code when errors are detected. `errors-behavior` can be one of the following:

  `continue` Continue normal execution.

  `remount-ro` Remount the filesystem read-only.

  `panic` Causes a kernel panic.

you don't care how fast your printer goes or are printing text on a slow printer with a buffer, then `500` (5 seconds) should be fine and will give you very low system load. This value generally should be lower for printing graphics than text, by a factor of approximately 10, for best performance.

`-c <CHARS>` The number of times to try to output a character to the printer before sleeping for `-t <TIME>`. It is the number of times around a loop that tries to send a character to the printer. `120` appears to be a good value for most printers. `250` is the default because there are some printers that require a wait this long, but feel free to change this. If you have a very fast printer like an HP Laserjet 4, a value of `10` might make more sense. If you have a really old printer, you can increase this.

Setting `-t <TIME>` to `0` is equivalent to setting `-c <CHARS>` to infinity.

`-w <WAIT>` The busy loop counter for the strobe signal. Although most printers appear to be able to deal with an extremely short strobe, some printers demand a longer one. Increasing this from the default `0` might make it possible to print with those printers. This can also make it possible to use longer cables.

`-a [on¦off]` This is whether to abort on printer error; the default is not to. If you are sitting at your computer, you probably want to be able to see an error and fix it and have the printer go on printing. On the other hand, if you aren't, you might want the ability that your printer spooler find out that the printer isn't ready, quit trying, and send you mail about it. The choice is yours.

`-o [on¦off]` This option is much like `-a`. It makes any `open()` of this device check to see that the device is online and not reporting any out-of-paper or other errors. This is the correct setting for most versions of `lpd`.

`-C [on¦off]` This option adds extra ("careful") error checking. When this option is on, the printer driver will ensure that the printer is online and not reporting any out-of-paper or other errors before sending data. This is particularly useful for printers that usually appear to accept data when turned off.

`-s` This option returns the current printer status, both as a decimal number from 0 to 255 and as a list of active flags. When this option is specified, `-q off`, turning off the display of the current IRQ, is implied.

`-o`, `-C`, and `-s` all require a Linux kernel version of 1.1.76 or later.

-r This option resets the port. It requires a Linux kernel version of 1.1.80 or later.

`-q [on¦off]` This option sets printing the display of the current IRQ setting.

Cohesive Systems, 26 August 1992

update_state

`update_state`—Update system state.

SYNOPSIS

```
update_state
```

DESCRIPTION

`update_state` updates a bunch of system states. It takes a long time to execute and would be suitable for execution in a `cron` job.

Currently, `update_state` performs the following functions: updates the locate database (in `/usr/lib/locate`), updates the `whatis` database(in `/usr/man`, `/usr/local/man`, `/usr/X386/man`, and `/usr/interviews/man`), and updates the TeX `ls-R` cache file (in `/usr/lib/texmf`).

```
Leap YEAR MONTH DAY HH:MM:SS CORR R/S
```

For example:

```
Leap 1974 Dec 31 23:59:60 + S
```

The `YEAR`, `MONTH`, `DAY`, and `HH:MM:SS` fields tell when the leap second happened. The `CORR` field should be `+` if a second was added or `-` if a second was skipped. The `R/S` field should be (an abbreviation of) `Stationary` if the leap second time given by the other fields should be interpreted as GMT or (an abbreviation of) `Rolling` if the leap second time given by the other fields should be interpreted as local wall clock time.

NOTE

For areas with more than two types of local time, you may need to use local standard time in the `AT` field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

FILE

`/usr/local/etc/zoneinfo` standard directory used for created files.

SEE ALSO

`newctime`(3), `tzfile`(5), `zdump`(8)

add_timer, del_timer, init_timer

add_timer, del_timer, init_timer—Manage event timers.

SYNOPSIS

```
#include <asm/param.h>
#include <linux/timer.h>
extern void add_timer(struct timer_list * timer);
extern int del_timer(struct timer_list * timer);
extern inline void init_timer(struct timer_list * timer);
```

DESCRIPTION

add_timer schedules an event, adding it to a linked list of events maintained by the kernel. del_timer deletes a scheduled event. timer points to a

```
struct timer_list {
struct timer_list *next;
struct timer_list *prev;
unsigned long expires;
unsigned long data;
void (*function)(unsigned long);
};
```

init_timer sets next and prev to NULL. This is required for the argument of add_timer. expires is the desired duration of the timer in jiffies (where there are HZ (typically 100) jiffies per second. When the timer expires, function is called with data as its argument. It is the responsibility of function to delete the event. If the same function is managing several timers, the argument can be used to distinguish which one expired.

RETURN VALUE

del_timer returns zero on error—if next or prev are not NULL, but the timer was not found. del_timer also sets expires to the time remaining before the timer expires and sets next and prev to NULL. Thus, calling del_timer followed immediately by add_timer is a no-op provided a kernel tick does not occur between the two calls.

AUTHOR

Linus Torvalds

Linux 1.2.8, 31 May 1995

adjust_clock

adjust_clock—Adjusts startup time counter to tick in GMT.

SYNOPSIS

```
linux/kernel/sys.c
void adjust_clock();
```

DESCRIPTION

This routine adjusts the startup time by adding the time zone information to it. The goal is to get the startup time ticking in GMT time.

NOTES

This routine is called from settimeofday(2) when the time-zone information is first set.

```
struct file_operations *f_op;
};
```

From `linux/fs/file_table.c`

```
struct file *first_file;
int nr_files = 0;
```

DESCRIPTION

The file table is fundamentally important to any UNIX system. It is where all open files (Linux includes closed files as well) are stored and managed by the kernel. For Linux, you can hardly do anything without referencing it in some way.

Linux stores its file table as a double circular linked list. The root pointer to the "head" of this list is `first_file`. Also, a count of how many entries are in the file table is maintained, called `nr_files`. Under this scheme, the file table for Linux could be as large as memory could hold. Unfortunately, this would be unmanageable in most cases. Your computer would be in the kernel most of the time when processes are more important. To keep this from happening, `nr_files` is tested against `NR_FILE` to limit the number of file table entries.

UNDERSTANDING THE STRUCTURE OF THE FILE TABLE

The file table is organized as a double circular linked list. Imagine a circle of people with everyone facing the same direction. Each person is facing so that one arm is in the circle and the other arm is outside the circle. Now if each person put his or her right hand on the shoulder of the person in front of him or her and in each person touched the person behind him or her with his or her left hand. You have formed two circles of arms, one inside and the other outside. The right arms represent pointers to the next entry (or person). The left arms represent pointers to the previous entry (or person).

THE FILE STRUCTURE, A FILE TABLE ENTRY

At first glance, a table entry looks quite simple. An entry contains how a file was opened, what `tty` device, a reference count, pointers to other entries, pointer to v-node (the `vfs` i-node) filesystem-specific i-node information, and so on.

`f_mode`	After ANDing with `O_ACCMODE`, this is what bits 0 and 1 mean:
00	No permissions needed
01	Read-permission
10	Write-permission
11	Read-write
`f_rdev`	It is used only with `tty` lines. It contains the major and minor numbers of the `tty` device.
`f_pos`	The current position in a file, if meaningful.
`f_flags`	Storage for the flags from `open()` and `fcntl()`
`f_count`	Reference counter
`f_reada`	This is a Boolean variable where `True` means that an actual read is needed.
`f_next`, `f_prev`	Pointers to other entries
`f_inode`	Pointer to v-node and filesystem-specific i-node information
`f_op`	Pointer to a file's operations

AUTHOR

Linus Torvalds

SEE ALSO

`insert_file_free`(9), `remove_file_free`(9), `put_last_free`(9) `grow_files`(9), `file_table_init`(9), `get_empty_filp`(9)

Linux 0.99.10, 11 July 1993

G

Q

R

S

W