NAME

apt − command−line interface

SYNOPSIS

apt [**−h**] [**−o=***config_string*] [**−c=***config_file*] [**−t=***target_release*] [**−a=***architecture*] {list | search | show | update | install *pkg* [{=*pkg_version_number* | /*target_release*}]... | remove *pkg*... | upgrade | full−upgrade | edit−sources | {−v | −−version} | {−h | −−help}}

DESCRIPTION

apt provides a high−level commandline interface for the package management system. It is intended as an end user interface and enables some options better suited for interactive usage by default compared to more specialized APT tools like **apt-get**(8) and **apt-cache**(8).

Much like **apt** itself, its manpage is intended as an end user interface and as such only mentions the most used commands and options partly to not duplicate information in multiple places and partly to avoid overwhelming readers with a cornucopia of options and details.

update (**apt-get**(8))
: **update** is used to download package information from all configured sources. Other commands operate on this data to e.g. perform package upgrades or search in and display details about all packages available for installation.

upgrade (**apt-get**(8))
: **upgrade** is used to install available upgrades of all packages currently installed on the system from the sources configured via **sources.list**(5). New packages will be installed if required to satisfy dependencies, but existing packages will never be removed. If an upgrade for a package requires the remove of an installed package the upgrade for this package isn't performed.

full−upgrade (**apt-get**(8))
: full−upgrade performs the function of upgrade but will remove currently installed packages if this is needed to upgrade the system as a whole.

install, **remove**, **purge** (**apt-get**(8))
: Performs the requested action on one or more packages specified via **regex**(7), **glob**(7) or exact match. The requested action can be overridden for specific packages by append a plus (+) to the package name to install this package or a minus (−) to remove it.

  A specific version of a package can be selected for installation by following the package name with an equals (=) and the version of the package to select. Alternatively the version from a specific release can be selected by following the package name with a forward slash (/) and codename (stretch, buster, sid ...) or suite name (stable, testing, unstable). This will also select versions from this release for dependencies of this package if needed to satisfy the request.

  Removing a package removes all packaged data, but leaves usually small (modified) user configuration files behind, in case the remove was an accident. Just issuing an installation request for the accidentally removed package will restore its function as before in that case. On the other hand you can get rid of these leftovers by calling **purge** even on already removed packages. Note that this does not affect any data or configuration stored in your home directory.

autoremove (**apt-get**(8))
: autoremove is used to remove packages that were automatically installed to satisfy dependencies for other packages and are now no longer needed as dependencies changed or the package(s) needing them were removed in the meantime.

  You should check that the list does not include applications you have grown to like even though they were once installed just as a dependency of another package. You can mark such a package as manually installed by using **apt-mark**(8). Packages which you have installed explicitly via **install** are also never proposed for automatic removal.

search (**apt-cache**(8))

search can be used to search for the given **regex**(7) term(s) in the list of available packages and display matches. This can e.g. be useful if you are looking for packages having a specific feature. If you are looking for a package including a specific file try **apt-file**(1).

show (**apt-cache**(8))
Show information about the given package(s) including its dependencies, installation and download size, sources the package is available from, the description of the packages content and much more. It can e.g. be helpful to look at this information before allowing **apt**(8) to remove a package or while searching for new packages to install.

list (work−in−progress)
list is somewhat similar to **dpkg−query −−list** in that it can display a list of packages satisfying certain criteria. It supports **glob**(7) patterns for matching package names as well as options to list installed (**−−installed**), upgradeable (**−−upgradeable**) or all available (**−−all−versions**) versions.

edit−sources (work−in−progress)
edit−sources lets you edit your **sources.list**(5) files in your preferred texteditor while also providing basic sanity checks.

SCRIPT USAGE AND DIFFERENCES FROM OTHER APT TOOLS

The **apt**(8) commandline is designed as an end−user tool and it may change behavior between versions. While it tries not to break backward compatibility this is not guaranteed either if a change seems beneficial for interactive use.

All features of **apt**(8) are available in dedicated APT tools like **apt-get**(8) and **apt-cache**(8) as well. **apt**(8) just changes the default value of some options (see **apt.conf**(5) and specifically the Binary scope). So you should prefer using these commands (potentially with some additional options enabled) in your scripts as they keep backward compatibility as much as possible.

SEE ALSO

apt-get(8), **apt-cache**(8), **sources.list**(5), **apt.conf**(5), **apt-config**(8), The APT User's guide in /usr/share/doc/apt−doc/, **apt_preferences**(5), the APT Howto.

DIAGNOSTICS

apt returns zero on normal operation, decimal 100 on error.

BUGS

APT bug page[1]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug−reporting.txt or the **reportbug**(1) command.

AUTHOR

APT team

NOTES

1. APT bug page
   http://bugs.debian.org/src:apt

NAME

bash − GNU Bourne-Again SHell

SYNOPSIS

bash [options] [command_string | file]

COPYRIGHT

DESCRIPTION

Bash is an **sh**-compatible command language interpreter that executes commands read from the standard input or from a file. **Bash** also incorporates useful features from the *Korn* and *C* shells (**ksh** and **csh**).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). **Bash** can be configured to be POSIX-conformant by default.

OPTIONS

All of the single-character shell options documented in the description of the **set** builtin command can be used as options when the shell is invoked. In addition, **bash** interprets the following options when it is invoked:

−c If the **−c** option is present, then commands are read from the first non-option argument *command_string*. If there are arguments after the *command_string*, the first argument is assigned to **$0** and any remaining arguments are assigned to the positional parameters. The assignment to **$0** sets the name of the shell, which is used in warning and error messages.

−i If the **−i** option is present, the shell is *interactive*.

−l Make **bash** act as if it had been invoked as a login shell (see **INVOCATION** below).

−r If the **−r** option is present, the shell becomes *restricted* (see **RESTRICTED SHELL** below).

−s If the **−s** option is present, or if no arguments remain after option processing, then commands are read from the standard input. This option allows the positional parameters to be set when invoking an interactive shell.

−D A list of all double-quoted strings preceded by **$** is printed on the standard output. These are the strings that are subject to language translation when the current locale is not **C** or **POSIX**. This implies the **−n** option; no commands will be executed.

[−+]O [*shopt_option***]**

shopt_option is one of the shell options accepted by the **shopt** builtin (see **SHELL BUILTIN COMMANDS** below). If *shopt_option* is present, **−O** sets the value of that option; **+O** unsets it. If *shopt_option* is not supplied, the names and values of the shell options accepted by **shopt** are printed on the standard output. If the invocation option is **+O**, the output is displayed in a format that may be reused as input.

−− A **−−** signals the end of options and disables further option processing. Any arguments after the **−−** are treated as filenames and arguments. An argument of **−** is equivalent to **−−**.

Bash also interprets a number of multi-character options. These options must appear on the command line before the single-character options to be recognized.

−−debugger

Arrange for the debugger profile to be executed before the shell starts. Turns on extended debugging mode (see the description of the **extdebug** option to the **shopt** builtin below).

−−dump−po−strings

Equivalent to **−D**, but the output is in the GNU *gettext* **po** (portable object) file format.

−−dump−strings

Equivalent to **−D**.

−−help Display a usage message on standard output and exit successfully.

−−init−file *file*

−−rcfile *file*

Execute commands from *file* instead of the system wide initialization file */etc/bash.bashrc* and the standard personal initialization file *~/.bashrc* if the shell is interactive (see **INVOCATION** below).

−−login
Equivalent to **−l**.

−−noediting
Do not use the GNU **readline** library to read command lines when the shell is interactive.

−−noprofile
Do not read either the system-wide startup file */etc/profile* or any of the personal initialization files *~/.bash_profile*, *~/.bash_login*, or *~/.profile*. By default, **bash** reads these files when it is invoked as a login shell (see **INVOCATION** below).

−−norc Do not read and execute the system wide initialization file */etc/bash.bashrc* and the personal initialization file *~/.bashrc* if the shell is interactive. This option is on by default if the shell is invoked as **sh**.

−−posix
Change the behavior of **bash** where the default operation differs from the POSIX standard to match the standard (*posix mode*). See **SEE ALSO** below for a reference to a document that details how posix mode affects bash's behavior.

−−restricted
The shell becomes restricted (see **RESTRICTED SHELL** below).

−−verbose
Equivalent to **−v**.

−−version
Show version information for this instance of **bash** on the standard output and exit successfully.

ARGUMENTS

If arguments remain after option processing, and neither the **−c** nor the **−s** option has been supplied, the first argument is assumed to be the name of a file containing shell commands. If **bash** is invoked in this fashion, **$0** is set to the name of the file, and the positional parameters are set to the remaining arguments. **Bash** reads and executes commands from this file, then exits. **Bash**'s exit status is the exit status of the last command executed in the script. If no commands are executed, the exit status is 0. An attempt is first made to open the file in the current directory, and, if no file is found, then the shell searches the directories in **PATH** for the script.

INVOCATION

A *login shell* is one whose first character of argument zero is a **−**, or one started with the **−−login** option.

An *interactive* shell is one started without non-option arguments (unless **−s** is specified) and without the **−c** option whose standard input and error are both connected to terminals (as determined by *isatty*(3)), or one started with the **−i** option. **PS1** is set and **$−** includes **i** if **bash** is interactive, allowing a shell script or a startup file to test this state.

The following paragraphs describe how **bash** executes its startup files. If any of the files exist but cannot be read, **bash** reports an error. Tildes are expanded in filenames as described below under **Tilde Expansion** in the **EXPANSION** section.

When **bash** is invoked as an interactive login shell, or as a non-interactive shell with the **−−login** option, it first reads and executes commands from the file */etc/profile*, if that file exists. After reading that file, it looks for *~/.bash_profile*, *~/.bash_login*, and *~/.profile*, in that order, and reads and executes commands from the first one that exists and is readable. The **−−noprofile** option may be used when the shell is started to inhibit this behavior.

When an interactive login shell exits, or a non-interactive login shell executes the **exit** builtin command, **bash** reads and executes commands from the file *~/.bash_logout*, if it exists.

When an interactive shell that is not a login shell is started, **bash** reads and executes commands from */etc/bash.bashrc* and *~/.bashrc*, if these files exist. This may be inhibited by using the **−−norc** option. The **−−rcfile** *file* option will force **bash** to read and execute commands from *file* instead of */etc/bash.bashrc* and *~/.bashrc*.

When **bash** is started non-interactively, to run a shell script, for example, it looks for the variable **BASH_ENV** in the environment, expands its value if it appears there, and uses the expanded value as the name of a file to read and execute. **Bash** behaves as if the following command were executed:

```
if [ -n "$BASH_ENV" ]; then . "$BASH_ENV"; fi
```

but the value of the **PATH** variable is not used to search for the filename.

If **bash** is invoked with the name **sh**, it tries to mimic the startup behavior of historical versions of **sh** as closely as possible, while conforming to the POSIX standard as well. When invoked as an interactive login shell, or a non-interactive shell with the **--login** option, it first attempts to read and execute commands from */etc/profile* and *~/.profile*, in that order. The **--noprofile** option may be used to inhibit this behavior. When invoked as an interactive shell with the name **sh**, **bash** looks for the variable **ENV**, expands its value if it is defined, and uses the expanded value as the name of a file to read and execute. Since a shell invoked as **sh** does not attempt to read and execute commands from any other startup files, the **--rcfile** option has no effect. A non-interactive shell invoked with the name **sh** does not attempt to read any other startup files. When invoked as **sh**, **bash** enters *posix* mode after the startup files are read.

When **bash** is started in *posix* mode, as with the **--posix** command line option, it follows the POSIX standard for startup files. In this mode, interactive shells expand the **ENV** variable and commands are read and executed from the file whose name is the expanded value. No other startup files are read.

Bash attempts to determine when it is being run with its standard input connected to a network connection, as when executed by the remote shell daemon, usually *rshd*, or the secure shell daemon *sshd*. If **bash** determines it is being run in this fashion, it reads and executes commands from *~/.bashrc* and *~/.bashrc*, if these files exist and are readable. It will not do this if invoked as **sh**. The **--norc** option may be used to inhibit this behavior, and the **--rcfile** option may be used to force another file to be read, but neither *rshd* nor *sshd* generally invoke the shell with those options or allow them to be specified.

If the shell is started with the effective user (group) id not equal to the real user (group) id, and the **-p** option is not supplied, no startup files are read, shell functions are not inherited from the environment, the **SHELLOPTS**, **BASHOPTS**, **CDPATH**, and **GLOBIGNORE** variables, if they appear in the environment, are ignored, and the effective user id is set to the real user id. If the **-p** option is supplied at invocation, the startup behavior is the same, but the effective user id is not reset.

DEFINITIONS

The following definitions are used throughout the rest of this document.

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**.

metacharacter

A character that, when unquoted, separates words. One of the following:
| & ; ( ) < > space tab newline

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** below) or the third word of a **case** or **for** command:

! case coproc do done elif else esac fi for function if in select then until while { } time [[ ]]

SHELL GRAMMAR

Simple Commands

A *simple command* is a sequence of optional variable assignments followed by **blank**-separated words and redirections, and terminated by a *control operator*. The first word specifies the command to be executed,

and is passed as argument zero. 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 one of the control operators **|** or **|&**. The format for a pipeline is:

[**time** [**−p**]] [!] *command* [[**|**⎪**|&**] *command2* ...]

The standard output of *command* is connected via a pipe to the standard input of *command2*. This connection is performed before any redirections specified by the command (see **REDIRECTION** below). If **|&** is used, *command*'s standard error, in addition to its standard output, is connected to *command2*'s standard input through the pipe; it is shorthand for **2>&1 |**. This implicit redirection of the standard error to the standard output is performed after any redirections specified by the command.

The return status of a pipeline is the exit status of the last command, unless the **pipefail** option is enabled. If **pipefail** is enabled, the pipeline's return status is the value of the last (rightmost) command to exit with a non-zero status, or zero if all commands exit successfully. If the reserved word **!** precedes a pipeline, the exit status of that pipeline is the logical negation of the exit status as described above. The shell waits for all commands in the pipeline to terminate before returning a value.

If the **time** reserved word precedes a pipeline, the elapsed as well as user and system time consumed by its execution are reported when the pipeline terminates. The **−p** option changes the output format to that specified by POSIX. When the shell is in *posix mode*, it does not recognize **time** as a reserved word if the next token begins with a '-'. The **TIMEFORMAT** variable may be set to a format string that specifies how the timing information should be displayed; see the description of **TIMEFORMAT** under **Shell Variables** below.

When the shell is in *posix mode*, **time** may be followed by a newline. In this case, the shell displays the total user and system time consumed by the shell and its children. The **TIMEFORMAT** variable may be used to specify the format of the time information.

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

Lists

A *list* is a sequence of one or more pipelines separated by one of the operators **;**, **&**, **&&**, or **||**, and optionally terminated by one of **;**, **&**, or **<newline>**.

Of these list operators, **&&** and **||** have equal precedence, followed by **;** and **&**, which have equal precedence.

A sequence of one or more newlines may appear in a *list* instead of a semicolon to delimit commands.

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.

AND and OR lists are sequences of one or more pipelines separated by the **&&** and **||** control operators, respectively. AND and OR lists are executed with left associativity. An AND list has the form

command1 **&&** *command2*

command2 is executed if, and only if, *command1* returns an exit status of zero.

An OR list has the form

command1 **||** *command2*

command2 is executed if and only if *command1* returns a non-zero exit status. The return status of AND and OR lists is the exit status of the last command executed in the list.

Compound Commands

A *compound command* is one of the following. In most cases a *list* in a command's description may be separated from the rest of the command by one or more newlines, and may be followed by a newline in

place of a semicolon.

(*list*) *list* is executed in a subshell environment (see **COMMAND EXECUTION ENVIRONMENT** below). Variable assignments and builtin commands that affect the shell's environment do not remain in effect after the command completes. The return status is the exit status of *list*.

{ *list*; } *list* is simply executed in the current shell environment. *list* must be terminated with a newline or semicolon. This is known as a *group command*. The return status is the exit status of *list*. Note that unlike the metacharacters **(** and **)**, **{** and **}** are *reserved words* and must occur where a reserved word is permitted to be recognized. Since they do not cause a word break, they must be separated from *list* by whitespace or another shell metacharacter.

((*expression*))
The *expression* is evaluated according to the rules described below under **ARITHMETIC EVALUATION**. If the value of the expression is non-zero, the return status is 0; otherwise the return status is 1. This is exactly equivalent to **let "***expression***"**.

[[*expression* **]]**
Return a status of 0 or 1 depending on the evaluation of the conditional expression *expression*. Expressions are composed of the primaries described below under **CONDITIONAL EXPRESSIONS**. Word splitting and pathname expansion are not performed on the words between the **[[** and **]]**; tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution, and quote removal are performed. Conditional operators such as **−f** must be unquoted to be recognized as primaries.

When used with **[[**, the < and > operators sort lexicographically using the current locale.

See the description of the *test* builtin command (section SHELL BUILTIN COMMANDS below) for the handling of parameters (i.e. missing parameters).

When the == and **!=** operators are used, the string to the right of the operator is considered a pattern and matched according to the rules described below under **Pattern Matching**, as if the **extglob** shell option were enabled. The = operator is equivalent to ==. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. The return value is 0 if the string matches (==) or does not match (**!=**) the pattern, and 1 otherwise. Any part of the pattern may be quoted to force the quoted portion to be matched as a string.

An additional binary operator, **=~**, is available, with the same precedence as == and **!=**. When it is used, the string to the right of the operator is considered an extended regular expression and matched accordingly (as in *regex*(3)). The return value is 0 if the string matches the pattern, and 1 otherwise. If the regular expression is syntactically incorrect, the conditional expression's return value is 2. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. Any part of the pattern may be quoted to force the quoted portion to be matched as a string. Bracket expressions in regular expressions must be treated carefully, since normal quoting characters lose their meanings between brackets. If the pattern is stored in a shell variable, quoting the variable expansion forces the entire pattern to be matched as a string. Substrings matched by parenthesized subexpressions within the regular expression are saved in the array variable **BASH_REMATCH**. The element of **BASH_REMATCH** with index 0 is the portion of the string matching the entire regular expression. The element of **BASH_REMATCH** with index *n* is the portion of the string matching the *n*th parenthesized subexpression.

Expressions may be combined using the following operators, listed in decreasing order of precedence:

(*expression* **)**
Returns the value of *expression*. This may be used to override the normal precedence of operators.

! *expression*
True if *expression* is false.

expression1 **&&** *expression2*
True if both *expression1* and *expression2* are true.

expression1 **||** *expression2*
True if either *expression1* or *expression2* is true.

The **&&** and **||** operators do not evaluate *expression2* if the value of *expression1* is sufficient to determine the return value of the entire conditional expression.

for *name* [[**in** [*word* ...]] ;] **do** *list* ; **done**
The list of words following **in** is expanded, generating a list of items. The variable *name* is set to each element of this list in turn, and *list* is executed each time. If the **in** *word* is omitted, the **for** command executes *list* once for each positional parameter that is set (see **PARAMETERS** below). The return status is the exit status of the last command that executes. If the expansion of the items following **in** results in an empty list, no commands are executed, and the return status is 0.

for ((*expr1* ; *expr2* ; *expr3*)) ; **do** *list* ; **done**
First, the arithmetic expression *expr1* is evaluated according to the rules described below under **ARITHMETIC EVALUATION**. The arithmetic expression *expr2* is then evaluated repeatedly until it evaluates to zero. Each time *expr2* evaluates to a non-zero value, *list* is executed and the arithmetic expression *expr3* is evaluated. If any expression is omitted, it behaves as if it evaluates to 1. The return value is the exit status of the last command in *list* that is executed, or false if any of the expressions is invalid.

select *name* [**in** *word*] ; **do** *list* ; **done**
The list of words following **in** is expanded, generating a list of items. The set of expanded words is printed on the standard error, each preceded by a number. If the **in** *word* is omitted, the positional parameters are printed (see **PARAMETERS** below). The **PS3** prompt is then displayed and a line read from the standard input. If the line consists of a number corresponding to one of the displayed words, then the value of *name* is set to that word. If the line is empty, the words and prompt are displayed again. If EOF is read, the command completes. Any other value read causes *name* to be set to null. The line read is saved in the variable **REPLY**. The *list* is executed after each selection until a **break** command is executed. The exit status of **select** is the exit status of the last command executed in *list*, or zero if no commands were executed.

case *word* **in** [[(] *pattern* [| *pattern*] ...) *list* ;;] ... **esac**
A **case** command first expands *word*, and tries to match it against each *pattern* in turn, using the same matching rules as for pathname expansion (see **Pathname Expansion** below). The *word* is expanded using tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution and quote removal. Each *pattern* examined is expanded using tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, and process substitution. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. When a match is found, the corresponding *list* is executed. If the **;;** operator is used, no subsequent matches are attempted after the first pattern match. Using **;&** in place of **;;** causes execution to continue with the *list* associated with the next set of patterns. Using **;;&** in place of **;;** causes the shell to test the next pattern list in the statement, if any, and execute any associated *list* on a successful match. The exit status is zero if no pattern matches. Otherwise, it is the exit status of the last command executed in *list*.

if *list*; **then** *list*; [**elif** *list*; **then** *list*;] ... [**else** *list*;] **fi**
The **if** *list* is executed. If its exit status is zero, the **then** *list* is executed. Otherwise, each **elif** *list* is executed in turn, and if its exit status is zero, the corresponding **then** *list* is executed and the command completes. Otherwise, the **else** *list* is executed, if present. The exit status is the exit status of the last command executed, or zero if no condition tested true.

while *list-1*; **do** *list-2*; **done**
until *list-1*; **do** *list-2*; **done**
The **while** command continuously executes the list *list-2* as long as the last command in the list *list-1* returns an exit status of zero. The **until** command is identical to the **while** command, except that the test is negated: *list-2* is executed as long as the last command in *list-1* returns a non-zero exit status. The exit status of the **while** and **until** commands is the exit status of the last command executed in *list-2*, or zero if none was executed.

Coprocesses

A *coprocess* is a shell command preceded by the **coproc** reserved word. A coprocess is executed asynchronously in a subshell, as if the command had been terminated with the **&** control operator, with a two-way pipe established between the executing shell and the coprocess.

The format for a coprocess is:

coproc [*NAME*] *command* [*redirections*]

This creates a coprocess named *NAME*. If *NAME* is not supplied, the default name is **COPROC**. *NAME* must not be supplied if *command* is a *simple command* (see above); otherwise, it is interpreted as the first word of the simple command. When the coprocess is executed, the shell creates an array variable (see **Arrays** below) named *NAME* in the context of the executing shell. The standard output of *command* is connected via a pipe to a file descriptor in the executing shell, and that file descriptor is assigned to *NAME*[0]. The standard input of *command* is connected via a pipe to a file descriptor in the executing shell, and that file descriptor is assigned to *NAME*[1]. This pipe is established before any redirections specified by the command (see **REDIRECTION** below). The file descriptors can be utilized as arguments to shell commands and redirections using standard word expansions. The file descriptors are not available in subshells. The process ID of the shell spawned to execute the coprocess is available as the value of the variable *NAME*_PID. The **wait** builtin command may be used to wait for the coprocess to terminate.

Since the coprocess is created as an asynchronous command, the **coproc** command always returns success. The return status of a coprocess is the exit status of *command*.

Shell Function Definitions

A shell function is an object that is called like a simple command and executes a compound command with a new set of positional parameters. Shell functions are declared as follows:

name () *compound−command* [*redirection*]
function *name* [()] *compound−command* [*redirection*]

This defines a function named *name*. The reserved word **function** is optional. If the **function** reserved word is supplied, the parentheses are optional. The *body* of the function is the compound command *compound−command* (see **Compound Commands** above). That command is usually a *list* of commands between { and }, but may be any command listed under **Compound Commands** above, with one exception: If the **function** reserved word is used, but the parentheses are not supplied, the braces are required. *compound−command* is executed whenever *name* is specified as the name of a simple command. When in *posix mode*, *name* may not be the name of one of the POSIX *special builtins*. Any redirections (see **REDIRECTION** below) specified when a function is defined are performed when the function is executed. The exit status of a function definition is zero unless a syntax error occurs or a readonly function with the same name already exists. When executed, the exit status of a function is the exit status of the last command executed in the body. (See **FUNCTIONS** below.)

COMMENTS

In a non-interactive shell, or an interactive shell in which the **interactive_comments** option to the **shopt** builtin is enabled (see **SHELL BUILTIN COMMANDS** below), a word beginning with **#** causes that word and all remaining characters on that line to be ignored. An interactive shell without the **interactive_comments** option enabled does not allow comments. The **interactive_comments** option is on by default in interactive shells.

QUOTING

Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to disable special treatment for special characters, to prevent reserved words from being recognized as such, and to prevent parameter expansion.

Each of the *metacharacters* listed above under **DEFINITIONS** has special meaning to the shell and must be quoted if it is to represent itself.

When the command history expansion facilities are being used (see **HISTORY EXPANSION** below), the *history expansion* character, usually **!**, must be quoted to prevent history expansion.

There are three quoting mechanisms: the *escape character*, single quotes, and double quotes.

A non-quoted backslash (\) is the *escape character*. It preserves the literal value of the next character that follows, with the exception of <newline>. If a \<newline> pair appears, and the backslash is not itself quoted, the \<newline> is treated as a line continuation (that is, it is removed from the input stream and effectively ignored).

Enclosing characters in single quotes preserves the literal value of each character within the quotes. A single quote may not occur between single quotes, even when preceded by a backslash.

Enclosing characters in double quotes preserves the literal value of all characters within the quotes, with the exception of **$**, `, \, and, when history expansion is enabled, **!**. When the shell is in *posix mode*, the **!** has no special meaning within double quotes, even when history expansion is enabled. The characters **$** and ` retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following characters: **$**, `, **"**, \, or **<newline>**. A double quote may be quoted within double quotes by preceding it with a backslash. If enabled, history expansion will be performed unless an **!** appearing in double quotes is escaped using a backslash. The backslash preceding the **!** is not removed.

The special parameters ***** and **@** have special meaning when in double quotes (see **PARAMETERS** below).

Words of the form **$**'*string*' are treated specially. The word expands to *string*, with backslash-escaped characters replaced as specified by the ANSI C standard. Backslash escape sequences, if present, are decoded as follows:

\a alert (bell)
\b backspace
\e
\E an escape character
\f form feed
\n new line
\r carriage return
\t horizontal tab
\v vertical tab
\\ backslash
\' single quote
\" double quote
\? question mark
nnn the eight-bit character whose value is the octal value *nnn* (one to three digits)
\x*HH* the eight-bit character whose value is the hexadecimal value *HH* (one or two hex digits)
\u*HHHH*
the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHH* (one to four hex digits)
\U*HHHHHHHH*
the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHHHHHH* (one to eight hex digits)
\c*x* a control-*x* character

The expanded result is single-quoted, as if the dollar sign had not been present.

A double-quoted string preceded by a dollar sign (**$**"*string*") will cause the string to be translated according to the current locale. If the current locale is **C** or **POSIX**, the dollar sign is ignored. If the string is translated and replaced, the replacement is double-quoted.

PARAMETERS

A *parameter* is an entity that stores values. It can be a *name*, a number, or one of the special characters listed below under **Special Parameters**. A *variable* is a parameter denoted by a *name*. A variable has a *value* and zero or more *attributes*. Attributes are assigned using the **declare** builtin command (see **declare** below in **SHELL BUILTIN COMMANDS**).

A parameter is set if it has been assigned a value. The null string is a valid value. Once a variable is set, it may be unset only by using the **unset** builtin command (see **SHELL BUILTIN COMMANDS** below).

A *variable* may be assigned to by a statement of the form

name=[*value*]

If *value* is not given, the variable is assigned the null string. All *values* undergo tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal (see **EXPANSION** below). If the variable has its **integer** attribute set, then *value* is evaluated as an arithmetic expression even if the $((...)) expansion is not used (see **Arithmetic Expansion** below). Word splitting is not performed, with the exception of **"$@"** as explained below under **Special Parameters**. Pathname expansion is not performed. Assignment statements may also appear as arguments to the **alias**, **declare**, **typeset**, **export**, **readonly**, and **local** builtin commands (*declaration* commands). When in *posix mode*, these builtins may appear in a command after one or more instances of the **command** builtin and retain these assignment statement properties.

In the context where an assignment statement is assigning a value to a shell variable or array index, the += operator can be used to append to or add to the variable's previous value. This includes arguments to builtin commands such as **declare** that accept assignment statements (*declaration* commands). When += is applied to a variable for which the *integer* attribute has been set, *value* is evaluated as an arithmetic expression and added to the variable's current value, which is also evaluated. When += is applied to an array variable using compound assignment (see **Arrays** below), the variable's value is not unset (as it is when using =), and new values are appended to the array beginning at one greater than the array's maximum index (for indexed arrays) or added as additional key−value pairs in an associative array. When applied to a string-valued variable, *value* is expanded and appended to the variable's value.

A variable can be assigned the *nameref* attribute using the **−n** option to the **declare** or **local** builtin commands (see the descriptions of **declare** and **local** below) to create a *nameref*, or a reference to another variable. This allows variables to be manipulated indirectly. Whenever the nameref variable is referenced, assigned to, unset, or has its attributes modified (other than using or changing the *nameref* attribute itself), the operation is actually performed on the variable specified by the nameref variable's value. A nameref is commonly used within shell functions to refer to a variable whose name is passed as an argument to the function. For instance, if a variable name is passed to a shell function as its first argument, running

```
declare −n ref=$1
```

inside the function creates a nameref variable **ref** whose value is the variable name passed as the first argument. References and assignments to **ref**, and changes to its attributes, are treated as references, assignments, and attribute modifications to the variable whose name was passed as **$1**. If the control variable in a **for** loop has the nameref attribute, the list of words can be a list of shell variables, and a name reference will be established for each word in the list, in turn, when the loop is executed. Array variables cannot be given the **nameref** attribute. However, nameref variables can reference array variables and subscripted array variables. Namerefs can be unset using the **−n** option to the **unset** builtin. Otherwise, if **unset** is executed with the name of a nameref variable as an argument, the variable referenced by the nameref variable will be unset.

Positional Parameters

A *positional parameter* is a parameter denoted by one or more digits, other than the single digit 0. Positional parameters are assigned from the shell's arguments when it is invoked, and may be reassigned using the **set** builtin command. Positional parameters may not be assigned to with assignment statements. The positional parameters are temporarily replaced when a shell function is executed (see **FUNCTIONS** below).

When a positional parameter consisting of more than a single digit is expanded, it must be enclosed in braces (see **EXPANSION** below).

Special Parameters

The shell treats several parameters specially. These parameters may only be referenced; assignment to them is not allowed.

***** Expands to the positional parameters, starting from one. When the expansion is not within double quotes, each positional parameter expands to a separate word. In contexts where it is performed, those words are subject to further word splitting and pathname expansion. When the expansion occurs within double quotes, it expands to a single word with the value of each parameter

separated by the first character of the **IFS** special variable. That is, "**$***" is equivalent to "**$1***c***$2***c***...**", where *c* is the first character of the value of the **IFS** variable. If **IFS** is unset, the parameters are separated by spaces. If **IFS** is null, the parameters are joined without intervening separators.

@ Expands to the positional parameters, starting from one. When the expansion occurs within double quotes, each parameter expands to a separate word. That is, "**$@**" is equivalent to "**$1**" "**$2**" ... If the double-quoted expansion occurs within a word, the expansion of the first parameter is joined with the beginning part of the original word, and the expansion of the last parameter is joined with the last part of the original word. When there are no positional parameters, "**$@**" and **$@** expand to nothing (i.e., they are removed).

Expands to the number of positional parameters in decimal.

? Expands to the exit status of the most recently executed foreground pipeline.

- Expands to the current option flags as specified upon invocation, by the **set** builtin command, or those set by the shell itself (such as the **-i** option).

$ Expands to the process ID of the shell. In a () subshell, it expands to the process ID of the current shell, not the subshell.

! Expands to the process ID of the job most recently placed into the background, whether executed as an asynchronous command or using the **bg** builtin (see **JOB CONTROL** below).

0 Expands to the name of the shell or shell script. This is set at shell initialization. If **bash** is invoked with a file of commands, **$0** is set to the name of that file. If **bash** is started with the **-c** option, then **$0** is set to the first argument after the string to be executed, if one is present. Otherwise, it is set to the filename used to invoke **bash**, as given by argument zero.

_ At shell startup, set to the absolute pathname used to invoke the shell or shell script being executed as passed in the environment or argument list. Subsequently, expands to the last argument to the previous command, after expansion. Also set to the full pathname used to invoke each command executed and placed in the environment exported to that command. When checking mail, this parameter holds the name of the mail file currently being checked.

Shell Variables

The following variables are set by the shell:

BASH Expands to the full filename used to invoke this instance of **bash**.

BASHOPTS

A colon-separated list of enabled shell options. Each word in the list is a valid argument for the **-s** option to the **shopt** builtin command (see **SHELL BUILTIN COMMANDS** below). The options appearing in **BASHOPTS** are those reported as *on* by **shopt**. If this variable is in the environment when **bash** starts up, each shell option in the list will be enabled before reading any startup files. This variable is read-only.

BASHPID

Expands to the process ID of the current **bash** process. This differs from **$$** under certain circumstances, such as subshells that do not require **bash** to be re-initialized.

BASH_ALIASES

An associative array variable whose members correspond to the internal list of aliases as maintained by the **alias** builtin. Elements added to this array appear in the alias list; however, unsetting array elements currently does not cause aliases to be removed from the alias list. If **BASH_ALIASES** is unset, it loses its special properties, even if it is subsequently reset.

BASH_ARGC

An array variable whose values are the number of parameters in each frame of the current **bash** execution call stack. The number of parameters to the current subroutine (shell function or script executed with **.** or **source**) is at the top of the stack. When a subroutine is executed, the number of parameters passed is pushed onto **BASH_ARGC**. The shell sets **BASH_ARGC** only when in extended debugging mode (see the description of the **extdebug** option to the **shopt** builtin below)

BASH_ARGV

An array variable containing all of the parameters in the current **bash** execution call stack. The final parameter of the last subroutine call is at the top of the stack; the first parameter of the initial call is at the bottom. When a subroutine is executed, the parameters supplied are pushed onto

BASH_ARGV. The shell sets **BASH_ARGV** only when in extended debugging mode (see the description of the **extdebug** option to the **shopt** builtin below)

BASH_CMDS
An associative array variable whose members correspond to the internal hash table of commands as maintained by the **hash** builtin. Elements added to this array appear in the hash table; however, unsetting array elements currently does not cause command names to be removed from the hash table. If **BASH_CMDS** is unset, it loses its special properties, even if it is subsequently reset.

BASH_COMMAND
The command currently being executed or about to be executed, unless the shell is executing a command as the result of a trap, in which case it is the command executing at the time of the trap.

BASH_EXECUTION_STRING
The command argument to the **−c** invocation option.

BASH_LINENO
An array variable whose members are the line numbers in source files where each corresponding member of **FUNCNAME** was invoked. **${BASH_LINENO[***$i***]}** is the line number in the source file (**${BASH_SOURCE[***$i+1***]}**) where **${FUNCNAME[***$i***]}** was called (or **${BASH_LINENO[***$i-1***]}** if referenced within another shell function). Use **LINENO** to obtain the current line number.

BASH_LOADABLES_PATH
A colon-separated list of directories in which the shell looks for dynamically loadable builtins specified by the **enable** command.

BASH_REMATCH
An array variable whose members are assigned by the **=~** binary operator to the **[[** conditional command. The element with index 0 is the portion of the string matching the entire regular expression. The element with index *n* is the portion of the string matching the *n*th parenthesized subexpression. This variable is read-only.

BASH_SOURCE
An array variable whose members are the source filenames where the corresponding shell function names in the **FUNCNAME** array variable are defined. The shell function **${FUNCNAME[***$i***]}** is defined in the file **${BASH_SOURCE[***$i***]}** and called from **${BASH_SOURCE[***$i+1***]}**.

BASH_SUBSHELL
Incremented by one within each subshell or subshell environment when the shell begins executing in that environment. The initial value is 0.

BASH_VERSINFO
A readonly array variable whose members hold version information for this instance of **bash**. The values assigned to the array members are as follows:

BASH_VERSINFO[0**]**	The major version number (the *release*).
BASH_VERSINFO[1**]**	The minor version number (the *version*).
BASH_VERSINFO[2**]**	The patch level.
BASH_VERSINFO[3**]**	The build version.
BASH_VERSINFO[4**]**	The release status (e.g., *beta1*).
BASH_VERSINFO[5**]**	The value of **MACHTYPE**.

BASH_VERSION
Expands to a string describing the version of this instance of **bash**.

COMP_CWORD
An index into **${COMP_WORDS}** of the word containing the current cursor position. This variable is available only in shell functions invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_KEY
The key (or final key of a key sequence) used to invoke the current completion function.

COMP_LINE
The current command line. This variable is available only in shell functions and external commands invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_POINT
The index of the current cursor position relative to the beginning of the current command. If the current cursor position is at the end of the current command, the value of this variable is equal to **${#COMP_LINE}**. This variable is available only in shell functions and external commands invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_TYPE
Set to an integer value corresponding to the type of completion attempted that caused a completion function to be called: *TAB*, for normal completion, *?*, for listing completions after successive tabs, *!*, for listing alternatives on partial word completion, *@*, to list completions if the word is not unmodified, or *%*, for menu completion. This variable is available only in shell functions and external commands invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_WORDBREAKS
The set of characters that the **readline** library treats as word separators when performing word completion. If **COMP_WORDBREAKS** is unset, it loses its special properties, even if it is subsequently reset.

COMP_WORDS
An array variable (see **Arrays** below) consisting of the individual words in the current command line. The line is split into words as **readline** would split it, using **COMP_WORDBREAKS** as described above. This variable is available only in shell functions invoked by the programmable completion facilities (see **Programmable Completion** below).

COPROC
An array variable (see **Arrays** below) created to hold the file descriptors for output from and input to an unnamed coprocess (see **Coprocesses** above).

DIRSTACK
An array variable (see **Arrays** below) containing the current contents of the directory stack. Directories appear in the stack in the order they are displayed by the **dirs** builtin. Assigning to members of this array variable may be used to modify directories already in the stack, but the **pushd** and **popd** builtins must be used to add and remove directories. Assignment to this variable will not change the current directory. If **DIRSTACK** is unset, it loses its special properties, even if it is subsequently reset.

EUID Expands to the effective user ID of the current user, initialized at shell startup. This variable is readonly.

FUNCNAME
An array variable containing the names of all shell functions currently in the execution call stack. The element with index 0 is the name of any currently-executing shell function. The bottom-most element (the one with the highest index) is `"main"`. This variable exists only when a shell function is executing. Assignments to **FUNCNAME** have no effect. If **FUNCNAME** is unset, it loses its special properties, even if it is subsequently reset.

This variable can be used with **BASH_LINENO** and **BASH_SOURCE**. Each element of **FUNCNAME** has corresponding elements in **BASH_LINENO** and **BASH_SOURCE** to describe the call stack. For instance, **${FUNCNAME[*$i*]}** was called from the file **${BASH_SOURCE[*$i+1*]}** at line number **${BASH_LINENO[*$i*]}**. The **caller** builtin displays the current call stack using this information.

GROUPS
An array variable containing the list of groups of which the current user is a member. Assignments to **GROUPS** have no effect. If **GROUPS** is unset, it loses its special properties, even if it is subsequently reset.

HISTCMD
The history number, or index in the history list, of the current command. If **HISTCMD** is unset, it loses its special properties, even if it is subsequently reset.

HOSTNAME
Automatically set to the name of the current host.

HOSTTYPE
Automatically set to a string that uniquely describes the type of machine on which **bash** is executing. The default is system-dependent.

LINENO
Each time this parameter is referenced, the shell substitutes a decimal number representing the current sequential line number (starting with 1) within a script or function. When not in a script or function, the value substituted is not guaranteed to be meaningful. If **LINENO** is unset, it loses its special properties, even if it is subsequently reset.

MACHTYPE
Automatically set to a string that fully describes the system type on which **bash** is executing, in the standard GNU *cpu-company-system* format. The default is system-dependent.

MAPFILE
An array variable (see **Arrays** below) created to hold the text read by the **mapfile** builtin when no variable name is supplied.

OLDPWD
The previous working directory as set by the **cd** command.

OPTARG
The value of the last option argument processed by the **getopts** builtin command (see **SHELL BUILTIN COMMANDS** below).

OPTIND
The index of the next argument to be processed by the **getopts** builtin command (see **SHELL BUILTIN COMMANDS** below).

OSTYPE
Automatically set to a string that describes the operating system on which **bash** is executing. The default is system-dependent.

PIPESTATUS
An array variable (see **Arrays** below) containing a list of exit status values from the processes in the most-recently-executed foreground pipeline (which may contain only a single command).

PPID The process ID of the shell's parent. This variable is readonly.

PWD The current working directory as set by the **cd** command.

RANDOM
Each time this parameter is referenced, a random integer between 0 and 32767 is generated. The sequence of random numbers may be initialized by assigning a value to **RANDOM**. If **RANDOM** is unset, it loses its special properties, even if it is subsequently reset.

READLINE_LINE
The contents of the **readline** line buffer, for use with `bind -x` (see **SHELL BUILTIN COMMANDS** below).

READLINE_POINT
The position of the insertion point in the **readline** line buffer, for use with `bind -x` (see **SHELL BUILTIN COMMANDS** below).

REPLY
Set to the line of input read by the **read** builtin command when no arguments are supplied.

SECONDS
Each time this parameter is referenced, the number of seconds since shell invocation is returned. If a value is assigned to **SECONDS**, the value returned upon subsequent references is the number of seconds since the assignment plus the value assigned. If **SECONDS** is unset, it loses its special properties, even if it is subsequently reset.

SHELLOPTS
A colon-separated list of enabled shell options. Each word in the list is a valid argument for the **-o** option to the **set** builtin command (see **SHELL BUILTIN COMMANDS** below). The options appearing in **SHELLOPTS** are those reported as *on* by **set -o**. If this variable is in the environment when **bash** starts up, each shell option in the list will be enabled before reading any startup files. This variable is read-only.

SHLVL
Incremented by one each time an instance of **bash** is started.

UID Expands to the user ID of the current user, initialized at shell startup. This variable is readonly.

The following variables are used by the shell. In some cases, **bash** assigns a default value to a variable; these cases are noted below.

BASH_COMPAT
The value is used to set the shell's compatibility level. See the description of the **shopt** builtin below under **SHELL BUILTIN COMMANDS** for a description of the various compatibility levels and their effects. The value may be a decimal number (e.g., 4.2) or an integer (e.g., 42) corresponding to the desired compatibility level. If **BASH_COMPAT** is unset or set to the empty string, the compatibility level is set to the default for the current version. If **BASH_COMPAT** is set to a value that is not one of the valid compatibility levels, the shell prints an error message and sets the compatibility level to the default for the current version. The valid compatibility levels correspond to the compatibility options accepted by the **shopt** builtin described below (for example, **compat42** means that 4.2 and 42 are valid values). The current version is also a valid value.

BASH_ENV
If this parameter is set when **bash** is executing a shell script, its value is interpreted as a filename containing commands to initialize the shell, as in *~/.bashrc*. The value of **BASH_ENV** is subjected to parameter expansion, command substitution, and arithmetic expansion before being interpreted as a filename. **PATH** is not used to search for the resultant filename.

BASH_XTRACEFD
If set to an integer corresponding to a valid file descriptor, **bash** will write the trace output generated when `set -x` is enabled to that file descriptor. The file descriptor is closed when **BASH_XTRACEFD** is unset or assigned a new value. Unsetting **BASH_XTRACEFD** or assigning it the empty string causes the trace output to be sent to the standard error. Note that setting **BASH_XTRACEFD** to 2 (the standard error file descriptor) and then unsetting it will result in the standard error being closed.

CDPATH
The search path for the **cd** command. This is a colon-separated list of directories in which the shell looks for destination directories specified by the **cd** command. A sample value is `".:~:/usr"`.

CHILD_MAX
Set the number of exited child status values for the shell to remember. Bash will not allow this value to be decreased below a POSIX-mandated minimum, and there is a maximum value (currently 8192) that this may not exceed. The minimum value is system-dependent.

COLUMNS
Used by the **select** compound command to determine the terminal width when printing selection lists. Automatically set if the **checkwinsize** option is enabled or in an interactive shell upon receipt of a **SIGWINCH**.

COMPREPLY
An array variable from which **bash** reads the possible completions generated by a shell function invoked by the programmable completion facility (see **Programmable Completion** below). Each array element contains one possible completion.

EMACS
If **bash** finds this variable in the environment when the shell starts with value `t`, it assumes that the shell is running in an Emacs shell buffer and disables line editing.

ENV Similar to **BASH_ENV**; used when the shell is invoked in POSIX mode.

EXECIGNORE
A colon-separated list of shell patterns (see **Pattern Matching**) defining the list of filenames to be ignored by command search using **PATH**. Files whose full pathnames match one of these patterns are not considered executable files for the purposes of completion and command execution via **PATH** lookup. This does not affect the behavior of the **[**, **test**, and **[[** commands. Full pathnames in the command hash table are not subject to **EXECIGNORE**. Use this variable to ignore shared library files that have the executable bit set, but are not executable files. The pattern matching

honors the setting of the **extglob** shell option.

FCEDIT
The default editor for the **fc** builtin command.

FIGNORE
A colon-separated list of suffixes to ignore when performing filename completion (see **READLINE** below). A filename whose suffix matches one of the entries in **FIGNORE** is excluded from the list of matched filenames. A sample value is `".o:~"`. (Quoting is needed when assigning a value to this variable, which contains tildes).

FUNCNEST
If set to a numeric value greater than 0, defines a maximum function nesting level. Function invocations that exceed this nesting level will cause the current command to abort.

GLOBIGNORE
A colon-separated list of patterns defining the set of filenames to be ignored by pathname expansion. If a filename matched by a pathname expansion pattern also matches one of the patterns in **GLOBIGNORE**, it is removed from the list of matches.

HISTCONTROL
A colon-separated list of values controlling how commands are saved on the history list. If the list of values includes *ignorespace*, lines which begin with a **space** character are not saved in the history list. A value of *ignoredups* causes lines matching the previous history entry to not be saved. A value of *ignoreboth* is shorthand for *ignorespace* and *ignoredups*. A value of *erasedups* causes all previous lines matching the current line to be removed from the history list before that line is saved. Any value not in the above list is ignored. If **HISTCONTROL** is unset, or does not include a valid value, all lines read by the shell parser are saved on the history list, subject to the value of **HISTIGNORE**. The second and subsequent lines of a multi-line compound command are not tested, and are added to the history regardless of the value of **HISTCONTROL**.

HISTFILE
The name of the file in which command history is saved (see **HISTORY** below). The default value is *~/.bash_history*. If unset, the command history is not saved when a shell exits.

HISTFILESIZE
The maximum number of lines contained in the history file. When this variable is assigned a value, the history file is truncated, if necessary, to contain no more than that number of lines by removing the oldest entries. The history file is also truncated to this size after writing it when a shell exits. If the value is 0, the history file is truncated to zero size. Non-numeric values and numeric values less than zero inhibit truncation. The shell sets the default value to the value of **HISTSIZE** after reading any startup files.

HISTIGNORE
A colon-separated list of patterns used to decide which command lines should be saved on the history list. Each pattern is anchored at the beginning of the line and must match the complete line (no implicit '*****' is appended). Each pattern is tested against the line after the checks specified by **HISTCONTROL** are applied. In addition to the normal shell pattern matching characters, '**&**' matches the previous history line. '**&**' may be escaped using a backslash; the backslash is removed before attempting a match. The second and subsequent lines of a multi-line compound command are not tested, and are added to the history regardless of the value of **HISTIGNORE**. The pattern matching honors the setting of the **extglob** shell option.

HISTSIZE
The number of commands to remember in the command history (see **HISTORY** below). If the value is 0, commands are not saved in the history list. Numeric values less than zero result in every command being saved on the history list (there is no limit). The shell sets the default value to 500 after reading any startup files.

HISTTIMEFORMAT
If this variable is set and not null, its value is used as a format string for *strftime*(3) to print the time stamp associated with each history entry displayed by the **history** builtin. If this variable is set, time stamps are written to the history file so they may be preserved across shell sessions. This uses the history comment character to distinguish timestamps from other history lines.

HOME
The home directory of the current user; the default argument for the **cd** builtin command. The value of this variable is also used when performing tilde expansion.

HOSTFILE
Contains the name of a file in the same format as */etc/hosts* that should be read when the shell needs to complete a hostname. The list of possible hostname completions may be changed while the shell is running; the next time hostname completion is attempted after the value is changed, **bash** adds the contents of the new file to the existing list. If **HOSTFILE** is set, but has no value, or does not name a readable file, **bash** attempts to read */etc/hosts* to obtain the list of possible hostname completions. When **HOSTFILE** is unset, the hostname list is cleared.

IFS The *Internal Field Separator* that is used for word splitting after expansion and to split lines into words with the **read** builtin command. The default value is ‘‘<space><tab><newline>’’.

IGNOREEOF
Controls the action of an interactive shell on receipt of an **EOF** character as the sole input. If set, the value is the number of consecutive **EOF** characters which must be typed as the first characters on an input line before **bash** exits. If the variable exists but does not have a numeric value, or has no value, the default value is 10. If it does not exist, **EOF** signifies the end of input to the shell.

INPUTRC
The filename for the **readline** startup file, overriding the default of *~/.inputrc* (see **READLINE** below).

LANG Used to determine the locale category for any category not specifically selected with a variable starting with **LC_**.

LC_ALL
This variable overrides the value of **LANG** and any other **LC_** variable specifying a locale category.

LC_COLLATE
This variable determines the collation order used when sorting the results of pathname expansion, and determines the behavior of range expressions, equivalence classes, and collating sequences within pathname expansion and pattern matching.

LC_CTYPE
This variable determines the interpretation of characters and the behavior of character classes within pathname expansion and pattern matching.

LC_MESSAGES
This variable determines the locale used to translate double-quoted strings preceded by a **$**.

LC_NUMERIC
This variable determines the locale category used for number formatting.

LC_TIME
This variable determines the locale category used for data and time formatting.

LINES Used by the **select** compound command to determine the column length for printing selection lists. Automatically set if the **checkwinsize** option is enabled or in an interactive shell upon receipt of a **SIGWINCH**.

MAIL If this parameter is set to a file or directory name and the **MAILPATH** variable is not set, **bash** informs the user of the arrival of mail in the specified file or Maildir-format directory.

MAILCHECK
Specifies how often (in seconds) **bash** checks for mail. The default is 60 seconds. When it is time to check for mail, the shell does so before displaying the primary prompt. If this variable is unset, or set to a value that is not a number greater than or equal to zero, the shell disables mail checking.

MAILPATH
A colon-separated list of filenames to be checked for mail. The message to be printed when mail arrives in a particular file may be specified by separating the filename from the message with a ‘?’. When used in the text of the message, **$_** expands to the name of the current mailfile. Example:
MAILPATH='/var/mail/bfox?"You have mail":~/shell−mail?"$_ has mail!"'
Bash can be configured to supply a default value for this variable (there is no value by default), but the location of the user mail files that it uses is system dependent (e.g., /var/mail/**$USER**).

OPTERR
If set to the value 1, **bash** displays error messages generated by the **getopts** builtin command (see **SHELL BUILTIN COMMANDS** below). **OPTERR** is initialized to 1 each time the shell is invoked or a shell script is executed.

PATH The search path for commands. It is a colon-separated list of directories in which the shell looks for commands (see **COMMAND EXECUTION** below). A zero-length (null) directory name in the value of **PATH** indicates the current directory. A null directory name may appear as two adjacent colons, or as an initial or trailing colon. The default path is system-dependent, and is set by the administrator who installs **bash**. A common value is `/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin`.

POSIXLY_CORRECT
If this variable is in the environment when **bash** starts, the shell enters *posix mode* before reading the startup files, as if the **−−posix** invocation option had been supplied. If it is set while the shell is running, **bash** enables *posix mode*, as if the command `set -o posix` had been executed.

PROMPT_COMMAND
If set, the value is executed as a command prior to issuing each primary prompt.

PROMPT_DIRTRIM
If set to a number greater than zero, the value is used as the number of trailing directory components to retain when expanding the **\w** and **\W** prompt string escapes (see **PROMPTING** below). Characters removed are replaced with an ellipsis.

PS0 The value of this parameter is expanded (see **PROMPTING** below) and displayed by interactive shells after reading a command and before the command is executed.

PS1 The value of this parameter is expanded (see **PROMPTING** below) and used as the primary prompt string. The default value is ‘‘**\s−\v\$** ’’.

PS2 The value of this parameter is expanded as with **PS1** and used as the secondary prompt string. The default is ‘‘> ’’.

PS3 The value of this parameter is used as the prompt for the **select** command (see **SHELL GRAMMAR** above).

PS4 The value of this parameter is expanded as with **PS1** and the value is printed before each command **bash** displays during an execution trace. The first character of **PS4** is replicated multiple times, as necessary, to indicate multiple levels of indirection. The default is ‘‘**+** ’’.

SHELL
The full pathname to the shell is kept in this environment variable. If it is not set when the shell starts, **bash** assigns to it the full pathname of the current user’s login shell.

TIMEFORMAT
The value of this parameter is used as a format string specifying how the timing information for pipelines prefixed with the **time** reserved word should be displayed. The **%** character introduces an escape sequence that is expanded to a time value or other information. The escape sequences and their meanings are as follows; the braces denote optional portions.

%%	A literal **%**.
%[*p***][l]R**	The elapsed time in seconds.
%[*p***][l]U**	The number of CPU seconds spent in user mode.
%[*p***][l]S**	The number of CPU seconds spent in system mode.
%P	The CPU percentage, computed as (%U + %S) / %R.

The optional *p* is a digit specifying the *precision*, the number of fractional digits after a decimal point. A value of 0 causes no decimal point or fraction to be output. At most three places after the decimal point may be specified; values of *p* greater than 3 are changed to 3. If *p* is not specified, the value 3 is used.

The optional **l** specifies a longer format, including minutes, of the form *MM*m*SS.FF*s. The value of *p* determines whether or not the fraction is included.

If this variable is not set, **bash** acts as if it had the value **$'\nreal\t%3lR\nuser\t%3lU\nsys\t%3lS'**. If the value is null, no timing information is displayed. A trailing newline is added when the format string is displayed.

TMOUT

If set to a value greater than zero, **TMOUT** is treated as the default timeout for the **read** builtin. The **select** command terminates if input does not arrive after **TMOUT** seconds when input is coming from a terminal. In an interactive shell, the value is interpreted as the number of seconds to wait for a line of input after issuing the primary prompt. **Bash** terminates after waiting for that number of seconds if a complete line of input does not arrive.

TMPDIR

If set, **bash** uses its value as the name of a directory in which **bash** creates temporary files for the shell's use.

auto_resume

This variable controls how the shell interacts with the user and job control. If this variable is set, single word simple commands without redirections are treated as candidates for resumption of an existing stopped job. There is no ambiguity allowed; if there is more than one job beginning with the string typed, the job most recently accessed is selected. The *name* of a stopped job, in this context, is the command line used to start it. If set to the value *exact*, the string supplied must match the name of a stopped job exactly; if set to *substring*, the string supplied needs to match a substring of the name of a stopped job. The *substring* value provides functionality analogous to the **%?** job identifier (see **JOB CONTROL** below). If set to any other value, the supplied string must be a prefix of a stopped job's name; this provides functionality analogous to the **%***string* job identifier.

histchars

The two or three characters which control history expansion and tokenization (see **HISTORY EXPANSION** below). The first character is the *history expansion* character, the character which signals the start of a history expansion, normally '**!**'. The second character is the *quick substitution* character, which is used as shorthand for re-running the previous command entered, substituting one string for another in the command. The default is '^'. The optional third character is the character which indicates that the remainder of the line is a comment when found as the first character of a word, normally '**#**'. The history comment character causes history substitution to be skipped for the remaining words on the line. It does not necessarily cause the shell parser to treat the rest of the line as a comment.

Arrays

Bash provides one-dimensional indexed and associative array variables. Any variable may be used as an indexed array; the **declare** builtin will explicitly declare an array. There is no maximum limit on the size of an array, nor any requirement that members be indexed or assigned contiguously. Indexed arrays are referenced using integers (including arithmetic expressions) and are zero-based; associative arrays are referenced using arbitrary strings. Unless otherwise noted, indexed array indices must be non-negative integers.

An indexed array is created automatically if any variable is assigned to using the syntax *name*[*subscript*]=*value*. The *subscript* is treated as an arithmetic expression that must evaluate to a number. To explicitly declare an indexed array, use **declare −a** *name* (see **SHELL BUILTIN COMMANDS** below). **declare −a** *name***[***subscript***]** is also accepted; the *subscript* is ignored.

Associative arrays are created using **declare −A** *name*.

Attributes may be specified for an array variable using the **declare** and **readonly** builtins. Each attribute applies to all members of an array.

Arrays are assigned to using compound assignments of the form *name*=**(**value*1* ... value*n***)**, where each *value* is of the form [*subscript*]=*string*. Indexed array assignments do not require anything but *string*. When assigning to indexed arrays, if the optional brackets and subscript are supplied, that index is assigned to; otherwise the index of the element assigned is the last index assigned to by the statement plus one. Indexing starts at zero.

When assigning to an associative array, the subscript is required.

This syntax is also accepted by the **declare** builtin. Individual array elements may be assigned to using the *name*[*subscript*]=*value* syntax introduced above. When assigning to an indexed array, if *name* is subscripted by a negative number, that number is interpreted as relative to one greater than the maximum index

of *name*, so negative indices count back from the end of the array, and an index of −1 references the last element.

Any element of an array may be referenced using ${*name*[*subscript*]}. The braces are required to avoid conflicts with pathname expansion. If *subscript* is **@** or *****, the word expands to all members of *name*. These subscripts differ only when the word appears within double quotes. If the word is double-quoted, ${*name*[*]} expands to a single word with the value of each array member separated by the first character of the **IFS** special variable, and ${*name*[@]} expands each element of *name* to a separate word. When there are no array members, ${*name*[@]} expands to nothing. If the double-quoted expansion occurs within a word, the expansion of the first parameter is joined with the beginning part of the original word, and the expansion of the last parameter is joined with the last part of the original word. This is analogous to the expansion of the special parameters ***** and **@** (see **Special Parameters** above). ${*#name*[*subscript*]} expands to the length of ${*name*[*subscript*]}. If *subscript* is ***** or **@**, the expansion is the number of elements in the array. If the *subscript* used to reference an element of an indexed array evaluates to a number less than zero, it is interpreted as relative to one greater than the maximum index of the array, so negative indices count back from the end of the array, and an index of −1 references the last element.

Referencing an array variable without a subscript is equivalent to referencing the array with a subscript of 0. Any reference to a variable using a valid subscript is legal, and **bash** will create an array if necessary.

An array variable is considered set if a subscript has been assigned a value. The null string is a valid value.

It is possible to obtain the keys (indices) of an array as well as the values. ${**!***name*[@]} and ${**!***name*[*]} expand to the indices assigned in array variable *name*. The treatment when in double quotes is similar to the expansion of the special parameters @ and * within double quotes.

The **unset** builtin is used to destroy arrays. **unset** *name*[*subscript*] destroys the array element at index *subscript*. Negative subscripts to indexed arrays are interpreted as described above. Care must be taken to avoid unwanted side effects caused by pathname expansion. **unset** *name*, where *name* is an array, or **unset** *name*[*subscript*], where *subscript* is ***** or **@**, removes the entire array.

The **declare**, **local**, and **readonly** builtins each accept a **−a** option to specify an indexed array and a **−A** option to specify an associative array. If both options are supplied, **−A** takes precedence. The **read** builtin accepts a **−a** option to assign a list of words read from the standard input to an array. The **set** and **declare** builtins display array values in a way that allows them to be reused as assignments.

EXPANSION

Expansion is performed on the command line after it has been split into words. There are seven kinds of expansion performed: *brace expansion*, *tilde expansion*, *parameter and variable expansion*, *command substitution*, *arithmetic expansion*, *word splitting*, and *pathname expansion*.

The order of expansions is: brace expansion; tilde expansion, parameter and variable expansion, arithmetic expansion, and command substitution (done in a left-to-right fashion); word splitting; and pathname expansion.

On systems that can support it, there is an additional expansion available: *process substitution*. This is performed at the same time as tilde, parameter, variable, and arithmetic expansion and command substitution.

After these expansions are performed, quote characters present in the original word are removed unless they have been quoted themselves (*quote removal*).

Only brace expansion, word splitting, and pathname expansion can change the number of words of the expansion; other expansions expand a single word to a single word. The only exceptions to this are the expansions of "**$@**" and "**${***name***[@]}**" as explained above (see **PARAMETERS**).

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 either a series of comma-separated strings or a sequence expression between a pair of braces, followed by an optional *postscript*. The preamble is prefixed to each string contained within the braces, and the postscript 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’.

A sequence expression takes the form **{***x***..***y***[..***incr***]}**, where *x* and *y* are either integers or single characters, and *incr*, an optional increment, is an integer. When integers are supplied, the expression expands to each number between *x* and *y*, inclusive. Supplied integers may be prefixed with *0* to force each term to have the same width. When either *x* or *y* begins with a zero, the shell attempts to force all generated terms to contain the same number of digits, zero-padding where necessary. When characters are supplied, the expression expands to each character lexicographically between *x* and *y*, inclusive, using the default C locale. Note that both *x* and *y* must be of the same type. When the increment is supplied, it is used as the difference between each term. The default increment is 1 or -1 as appropriate.

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 or a valid sequence expression. Any incorrectly formed brace expansion is left unchanged. A **{** or **,** may be quoted with a backslash to prevent its being considered part of a brace expression. To avoid conflicts with parameter expansion, the string **${** is not considered eligible for brace expansion.

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

```
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 historical versions of **sh**. **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 **+B** option or disable brace expansion with the **+B** option to the **set** command (see **SHELL BUILTIN COMMANDS** below).

Tilde Expansion

If a word begins with an unquoted tilde character (‘**~**’), all of the characters preceding the first unquoted slash (or all characters, if there is no unquoted slash) are considered a *tilde-prefix*. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the tilde are treated as a possible *login name*. If this login name is the null string, the tilde is replaced with the value of the shell parameter **HOME**. If **HOME** is unset, the home directory of the user executing the shell is substituted instead. Otherwise, the tilde-prefix is replaced with the home directory associated with the specified login name.

If the tilde-prefix is a ‘~+’, the value of the shell variable **PWD** replaces the tilde-prefix. If the tilde-prefix is a ‘~−’, the value of the shell variable **OLDPWD**, if it is set, is substituted. If the characters following the tilde in the tilde-prefix consist of a number *N*, optionally prefixed by a ‘+’ or a ‘−’, the tilde-prefix is replaced with the corresponding element from the directory stack, as it would be displayed by the **dirs** builtin invoked with the tilde-prefix as an argument. If the characters following the tilde in the tilde-prefix consist of a number without a leading ‘+’ or ‘−’, ‘+’ is assumed.

If the login name is invalid, or the tilde expansion fails, the word is unchanged.

Each variable assignment is checked for unquoted tilde-prefixes immediately following a **:** or the first **=**. In these cases, tilde expansion is also performed. Consequently, one may use filenames 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.

When braces are used, the matching ending brace is the first '**}**' not escaped by a backslash or within a quoted string, and not within an embedded arithmetic expansion, command substitution, or parameter expansion.

${*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 which is not to be interpreted as part of its name. The *parameter* is a shell parameter as described above **PARAMETERS**) or an array reference (**Arrays**).

If the first character of *parameter* is an exclamation point (**!**), and *parameter* is not a *nameref*, it introduces a level of variable indirection. **Bash** uses the value of the variable formed from the rest of *parameter* as the name of the variable; this variable is then expanded and that value is used in the rest of the substitution, rather than the value of *parameter* itself. This is known as *indirect expansion*. If *parameter* is a nameref, this expands to the name of the variable referenced by *parameter* instead of performing the complete indirect expansion. The exceptions to this are the expansions of ${**!***prefix******} and ${**!***name*[@]} described below. The exclamation point must immediately follow the left brace in order to introduce indirection.

In each of the cases below, *word* is subject to tilde expansion, parameter expansion, command substitution, and arithmetic expansion.

When not performing substring expansion, using the forms documented below (e.g., **:-**), **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***:***offset*}
${*parameter***:***offset***:***length*}
Substring Expansion. Expands to up to *length* characters of the value of *parameter* starting at the character specified by *offset*. If *parameter* is **@**, an indexed array subscripted by **@** or *****, or an associative array name, the results differ as described below. If *length* is omitted, expands to the substring of the value of *parameter* starting at the character specified by *offset* and extending to the end of the value. *length* and *offset* are arithmetic expressions (see **ARITHMETIC EVALUATION** below).

If *offset* evaluates to a number less than zero, the value is used as an offset in characters from the end of the value of *parameter*. If *length* evaluates to a number less than zero, it is interpreted as an offset in characters from the end of the value of *parameter* rather than a number of characters, and the expansion is the characters between *offset* and that result. Note that a negative offset must be separated from the colon by at least one space to avoid being confused with the **:-** expansion.

If *parameter* is **@**, the result is *length* positional parameters beginning at *offset*. A negative *offset* is taken relative to one greater than the greatest positional parameter, so an offset of -1 evaluates to the last positional parameter. It is an expansion error if *length* evaluates to a number less than

zero.

If *parameter* is an indexed array name subscripted by @ or *, the result is the *length* members of the array beginning with ${*parameter*[*offset*]}. A negative *offset* is taken relative to one greater than the maximum index of the specified array. It is an expansion error if *length* evaluates to a number less than zero.

Substring expansion applied to an associative array produces undefined results.

Substring indexing is zero-based unless the positional parameters are used, in which case the indexing starts at 1 by default. If *offset* is 0, and the positional parameters are used, **$0** is prefixed to the list.

${**!***prefix******}
${**!***prefix***@**}
Names matching prefix. Expands to the names of variables whose names begin with *prefix*, separated by the first character of the **IFS** special variable. When @ is used and the expansion appears within double quotes, each variable name expands to a separate word.

${**!***name*[@]}
${**!***name*[*]}
List of array keys. If *name* is an array variable, expands to the list of array indices (keys) assigned in *name*. If *name* is not an array, expands to 0 if *name* is set and null otherwise. When @ is used and the expansion appears within double quotes, each key expands to a separate word.

${#*parameter*}
Parameter length. The length in characters of the value of *parameter* is substituted. If *parameter* is ***** or **@**, the value substituted is the number of positional parameters. If *parameter* is an array name subscripted by ***** or **@**, the value substituted is the number of elements in the array. If *parameter* is an indexed array name subscripted by a negative number, that number is interpreted as relative to one greater than the maximum index of *parameter*, so negative indices count back from the end of the array, and an index of −1 references the last element.

${*parameter***#***word*}
${*parameter***##***word*}
Remove matching prefix pattern. 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 result of the expansion is the expanded value of *parameter* with the shortest matching pattern (the ''#'' case) or the longest matching pattern (the ''##'' case) deleted. If *parameter* is **@** or *****, the pattern removal operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with **@** or *****, the pattern removal operation is applied to each member of the array in turn, and the expansion is the resultant list.

${*parameter***%***word*}
${*parameter***%%***word*}
Remove matching suffix pattern. The *word* is expanded to produce a pattern just as in pathname expansion. If the pattern matches a trailing portion of the expanded value of *parameter*, then the result of the expansion is the expanded value of *parameter* with the shortest matching pattern (the ''**%**'' case) or the longest matching pattern (the ''**%%**'' case) deleted. If *parameter* is **@** or *****, the pattern removal operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with **@** or *****, the pattern removal operation is applied to each member of the array in turn, and the expansion is the resultant list.

${*parameter*/*pattern*/*string*}
Pattern substitution. The *pattern* is expanded to produce a pattern just as in pathname expansion. *Parameter* is expanded and the longest match of *pattern* against its value is replaced with *string*. If *pattern* begins with /, all matches of *pattern* are replaced with *string*. Normally only the first match is replaced. If *pattern* begins with #, it must match at the beginning of the expanded

value of *parameter*. If *pattern* begins with **%**, it must match at the end of the expanded value of *parameter*. If *string* is null, matches of *pattern* are deleted and the / following *pattern* may be omitted. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. If *parameter* is **@** or *****, the substitution operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with **@** or *****, the substitution operation is applied to each member of the array in turn, and the expansion is the resultant list.

${*parameter*^*pattern*}
${*parameter*^^*pattern*}
${*parameter*,*pattern*}
${*parameter*,,*pattern*}

Case modification. This expansion modifies the case of alphabetic characters in *parameter*. The *pattern* is expanded to produce a pattern just as in pathname expansion. Each character in the expanded value of *parameter* is tested against *pattern*, and, if it matches the pattern, its case is converted. The pattern should not attempt to match more than one character. The ^ operator converts lowercase letters matching *pattern* to uppercase; the **,** operator converts matching uppercase letters to lowercase. The ^^ and **,,** expansions convert each matched character in the expanded value; the ^ and **,** expansions match and convert only the first character in the expanded value. If *pattern* is omitted, it is treated like a **?**, which matches every character. If *parameter* is **@** or *****, the case modification operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with **@** or *****, the case modification operation is applied to each member of the array in turn, and the expansion is the resultant list.

${*parameter*@*operator*}

Parameter transformation. The expansion is either a transformation of the value of *parameter* or information about *parameter* itself, depending on the value of *operator*. Each *operator* is a single letter:

- **Q** The expansion is a string that is the value of *parameter* quoted in a format that can be reused as input.
- **E** The expansion is a string that is the value of *parameter* with backslash escape sequences expanded as with the **$'...'** quoting mechansim.
- **P** The expansion is a string that is the result of expanding the value of *parameter* as if it were a prompt string (see **PROMPTING** below).
- **A** The expansion is a string in the form of an assignment statement or **declare** command that, if evaluated, will recreate *parameter* with its attributes and value.
- **a** The expansion is a string consisting of flag values representing *parameter*'s attributes.

If *parameter* is **@** or *****, the operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with **@** or *****, the case modification operation is applied to each member of the array in turn, and the expansion is the resultant list.

The result of the expansion is subject to word splitting and pathname expansion as described below.

Command Substitution

Command substitution allows the output of a command to replace the command name. There are two forms:

$(*command* **)**

or

`` `command` ``

Bash performs the expansion by executing *command* in a subshell environment and replacing the command substitution with the standard output of the command, with any trailing newlines deleted. Embedded newlines are not deleted, but they may be removed during word splitting. The command substitution **$(cat** *file***)**

can be replaced by the equivalent but faster **$(<** *file***)**.

When the old-style backquote form of substitution is used, backslash retains its literal meaning except when followed by **$**, `, or \. The first backquote not preceded by a backslash terminates the command substitution. 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 backquoted 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. The format for arithmetic expansion is:

$((*expression***))**

The old format **$[***expression***]** is deprecated and will be removed in upcoming versions of bash.

The *expression* is treated as if it were within double quotes, but a double quote inside the parentheses is not treated specially. All tokens in the expression undergo parameter and variable expansion, command substitution, and quote removal. The result is treated as the arithmetic expression to be evaluated. Arithmetic expansions may be nested.

The evaluation is performed according to the rules listed below under **ARITHMETIC EVALUATION**. If *expression* is invalid, **bash** prints a message indicating failure and no substitution occurs.

Process Substitution

Process substitution allows a process's input or output to be referred to using a filename. It takes the form of **<(***list***)** or **>(***list***)**. The process *list* is run asynchronously, and its input or output appears as a filename. This filename 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*. Process substitution is supported on systems that support named pipes (*FIFOs*) or the **/dev/fd** method of naming open files.

When available, 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 using these characters as field terminators. If **IFS** is unset, or its value is exactly **<space><tab><newline>**, the default, then sequences of **<space>**, **<tab>**, and **<newline>** at the beginning and end of the results of the previous expansions are ignored, and any sequence of **IFS** characters not at the beginning or end serves to delimit words. If **IFS** has a value other than the default, then sequences of the whitespace characters **space**, **tab**, and **newline** 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.

Explicit null arguments (`""` or `''`) are retained and passed to commands as empty strings. Unquoted implicit null arguments, resulting from the expansion of parameters that have no values, are removed. If a parameter with no value is expanded within double quotes, a null argument results and is retained and passed to a command as an empty string. When a quoted null argument appears as part of a word whose expansion is non-null, the null argument is removed. That is, the word `-d''` becomes `-d` after word splitting and null argument removal.

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 filenames matching the pattern (see **Pattern Matching** below). If no matching filenames are found, and the shell option **nullglob** is not enabled, the word is left unchanged. If the **nullglob** option is set, and no matches are found, the word is removed. If the **failglob** shell option is set, and no matches are found, an error message is printed and the command is not executed. If the shell option **nocaseglob** is enabled, the match is performed without regard to the case of alphabetic characters. Note that when using range expressions like [a-z] (see below), letters of the other case may be included, depending on the setting of **LC_COLLATE.** When a pattern is used for pathname expansion, the character **".."** at the start of a name or immediately following a slash must be matched explicitly, unless the shell option **dotglob** is set. When matching a pathname, the slash character must always be matched explicitly. In other cases, the **"."** character is not treated specially. See the description of **shopt** below under **SHELL BUILTIN COMMANDS** for a description of the **nocaseglob**, **nullglob**, **failglob**, and **dotglob** shell options.

The **GLOBIGNORE** shell variable may be used to restrict the set of filenames matching a *pattern*. If **GLOBIGNORE** is set, each matching filename that also matches one of the patterns in **GLOBIGNORE** is removed from the list of matches. If the **nocaseglob** option is set, the matching against the patterns in **GLOBIGNORE** is performed without regard to case. The filenames **"."** and **".."** are always ignored when **GLOBIGNORE** is set and not null. However, setting **GLOBIGNORE** to a non-null value has the effect of enabling the **dotglob** shell option, so all other filenames beginning with a **"."** will match. To get the old behavior of ignoring filenames beginning with a **"."**, make **".*"** one of the patterns in **GLOBIGNORE**. The **dotglob** option is disabled when **GLOBIGNORE** is unset. The pattern matching honors the setting of the **extglob** shell option.

Pattern Matching

Any character that appears in a pattern, other than the special pattern characters described below, matches itself. The NUL character may not occur in a pattern. A backslash escapes the following character; the escaping backslash is discarded when matching. The special pattern characters must be quoted if they are to be matched literally.

The special pattern characters have the following meanings:

***** Matches any string, including the null string. When the **globstar** shell option is enabled, and ***** is used in a pathname expansion context, two adjacent *****s used as a single pattern will match all files and zero or more directories and subdirectories. If followed by a /, two adjacent *****s will match only directories and subdirectories.

? Matches any single character.

[...] Matches any one of the enclosed characters. A pair of characters separated by a hyphen denotes a *range expression*; any character that falls between those two characters, inclusive, using the current locale's collating sequence and character set, is matched. If the first character following the **[** is a **!** or a ^ then any character not enclosed is matched. The sorting order of characters in range expressions is determined by the current locale and the values of the **LC_COLLATE** or **LC_ALL** shell variables, if set. To obtain the traditional interpretation of range expressions, where **[a−d]** is equivalent to **[abcd]**, set value of the **LC_ALL** shell variable to **C**, or enable the **globasciiranges** shell option. A **−** may be matched by including it as the first or last character in the set. A **]** may be matched by including it as the first character in the set.

Within **[** and **]**, *character classes* can be specified using the syntax **[:***class***:]**, where *class* is one of the following classes defined in the POSIX standard:
alnum alpha ascii blank cntrl digit graph lower print punct space upper word xdigit
A character class matches any character belonging to that class. The **word** character class matches letters, digits, and the character _.

Within **[** and **]**, an *equivalence class* can be specified using the syntax **[=***c***=]**, which

matches all characters with the same collation weight (as defined by the current locale) as the character *c*.

Within **[** and **]**, the syntax **[.***symbol***.]** matches the collating symbol *symbol*.

If the **extglob** shell option is enabled using the **shopt** builtin, several extended pattern matching operators are recognized. In the following description, a *pattern-list* is a list of one or more patterns separated by a |. Composite patterns may be formed using one or more of the following sub-patterns:

?(*pattern-list***)**
Matches zero or one occurrence of the given patterns
(pattern-list***)**
Matches zero or more occurrences of the given patterns
+(*pattern-list***)**
Matches one or more occurrences of the given patterns
@(*pattern-list***)**
Matches one of the given patterns
!(*pattern-list***)**
Matches anything except one of the given patterns

Quote Removal

After the preceding expansions, all unquoted occurrences of the characters \, ', and " that did not result from one of the above expansions are removed.

REDIRECTION

Before a command is executed, its input and output may be *redirected* using a special notation interpreted by the shell. Redirection allows commands' file handles to be duplicated, opened, closed, made to refer to different files, and can change the files the command reads from and writes to. Redirection may also be used to modify file handles in 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.

Each redirection that may be preceded by a file descriptor number may instead be preceded by a word of the form {*varname*}. In this case, for each redirection operator except >&- and <&-, the shell will allocate a file descriptor greater than or equal to 10 and assign it to *varname*. If >&- or <&- is preceded by {*varname*}, the value of *varname* defines the file descriptor to close.

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 following the redirection operator in the following descriptions, unless otherwise noted, is subjected to brace expansion, tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, quote removal, pathname expansion, and word splitting. 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 from the standard output before the standard output was redirected to *dirlist*.

Bash handles several filenames specially when they are used in redirections, as described in the following table. If the operating system on which **bash** is running provides these special files, bash will use them; otherwise it will emulate them internally with the behavior described below.

/dev/fd/*fd*
If *fd* is a valid integer, file descriptor *fd* is duplicated.
/dev/stdin
File descriptor 0 is duplicated.
/dev/stdout
File descriptor 1 is duplicated.
/dev/stderr
File descriptor 2 is duplicated.
/dev/tcp/*host*/*port*
If *host* is a valid hostname or Internet address, and *port* is an integer port number or service name, **bash** attempts to open the corresponding TCP socket.
/dev/udp/*host*/*port*
If *host* is a valid hostname or Internet address, and *port* is an integer port number or service name, **bash** attempts to open the corresponding UDP socket.

A failure to open or create a file causes the redirection to fail.

Redirections using file descriptors greater than 9 should be used with care, as they may conflict with file descriptors the shell uses internally.

Note that the **exec** builtin command can make redirections take effect in the current shell.

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 >, and the **noclobber** option to the **set** builtin has been enabled, the redirection will fail if the file whose name results from the expansion of *word* exists and is a regular file. If the redirection operator is >|, or the redirection operator is > and the **noclobber** option to the **set** builtin command is not enabled, the redirection is attempted even if the file named by *word* exists.

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
This construct 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*.

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

When using the second form, *word* may not expand to a number or −. If it does, other redirection operators apply (see **Duplicating File Descriptors** below) for compatibility reasons.

Appending Standard Output and Standard Error

This construct allows both the standard output (file descriptor 1) and the standard error output (file descriptor 2) to be appended to the file whose name is the expansion of *word*.

The format for appending standard output and standard error is:

&>>*word*

This is semantically equivalent to

>>*word* 2>**&**1

(see **Duplicating File Descriptors** below).

Here Documents

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

The format of here-documents is:

[*n*]<<[−]*word*
here-document
delimiter

No parameter and variable expansion, command substitution, arithmetic expansion, or pathname expansion is performed on *word*. If any part of *word* is quoted, the *delimiter* is the result of quote removal on *word*, and the lines in the here-document are not expanded. If *word* is unquoted, all lines of the here-document are subjected to parameter expansion, command substitution, and arithmetic expansion, the character sequence \<**newline**> is ignored, and \ must be used to quote the characters \, **$**, and `.

If the redirection operator is <<−, then all leading tab characters are stripped from input lines and the line containing *delimiter*. This allows here-documents within shell scripts to be indented in a natural fashion.

Here Strings

A variant of here documents, the format is:

[*n*]<<<*word*

The *word* undergoes brace expansion, tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal. Pathname expansion and word splitting are not performed. The result is supplied as a single string, with a newline appended, to the command on its standard input (or file descriptor *n* if *n* is specified).

Duplicating File Descriptors

The redirection operator

[*n*]<**&***word*

is used to duplicate input file descriptors. If *word* expands to one or more digits, the file descriptor denoted by *n* is made to be a copy of that file descriptor. If the digits in *word* do not specify a file descriptor open for input, a redirection error occurs. If *word* evaluates to −, file descriptor *n* is closed. If *n* is not specified, the standard input (file descriptor 0) is used.

The operator

[*n*]>**&***word*

is used similarly to duplicate output file descriptors. If *n* is not specified, the standard output (file descriptor 1) is used. If the digits in *word* do not specify a file descriptor open for output, a redirection error occurs. If *word* evaluates to −, file descriptor *n* is closed. As a special case, if *n* is omitted, and *word* does not expand to one or more digits or −, the standard output and standard error are redirected as described

previously.

Moving File Descriptors

The redirection operator

[*n*]<**&***digit*−

moves the file descriptor *digit* to file descriptor *n*, or the standard input (file descriptor 0) if *n* is not specified. *digit* is closed after being duplicated to *n*.

Similarly, the redirection operator

[*n*]>**&***digit*−

moves the file descriptor *digit* to file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified.

Opening File Descriptors for Reading and Writing

The redirection operator

[*n*]<>*word*

causes the file whose name is the expansion of *word* to be opened for both reading and writing on file descriptor *n*, or on file descriptor 0 if *n* is not specified. If the file does not exist, it is created.

ALIASES

Aliases allow a string to be substituted for a word when it is used as the first word of a simple command. The shell maintains a list of aliases that may be set and unset with the **alias** and **unalias** builtin commands (see **SHELL BUILTIN COMMANDS** below). The first word of each simple command, if unquoted, is checked to see if it has an alias. If so, that word is replaced by the text of the alias. The characters /, **$**, `, and = and any of the shell *metacharacters* or quoting characters listed above may not appear in an alias name. The replacement text may contain any valid shell input, including shell metacharacters. The first word of the replacement text 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. If arguments are needed, a shell function should be used (see **FUNCTIONS** below).

Aliases are not expanded when the shell is not interactive, unless the **expand_aliases** shell option is set using **shopt** (see the description of **shopt** under **SHELL BUILTIN COMMANDS** below).

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. 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 a function definition is read, not when the function is executed, because a function definition is itself a 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.

For almost every purpose, aliases are superseded by shell functions.

FUNCTIONS

A shell function, defined as described above under **SHELL GRAMMAR**, stores a series of commands for later execution. When the name of a shell function is used as a simple command name, the list of commands associated with that function name is executed. Functions are executed in the context of the current shell; no new process is created to interpret them (contrast this with the execution of a shell script). When a function is executed, the arguments to the function become the positional parameters during its execution. The special parameter **#** is updated to reflect the change. Special parameter **0** is unchanged. The first

element of the **FUNCNAME** variable is set to the name of the function while the function is executing.

All other aspects of the shell execution environment are identical between a function and its caller with these exceptions: the **DEBUG** and **RETURN** traps (see the description of the **trap** builtin under **SHELL BUILTIN COMMANDS** below) are not inherited unless the function has been given the **trace** attribute (see the description of the **declare** builtin below) or the **−o functrace** shell option has been enabled with the **set** builtin (in which case all functions inherit the **DEBUG** and **RETURN** traps), and the **ERR** trap is not inherited unless the **−o errtrace** shell option has been enabled.

Variables local to the function may be declared with the **local** builtin command. Ordinarily, variables and their values are shared between the function and its caller.

The **FUNCNEST** variable, if set to a numeric value greater than 0, defines a maximum function nesting level. Function invocations that exceed the limit cause the entire command to abort.

If the builtin command **return** is executed in a function, the function completes and execution resumes with the next command after the function call. Any command associated with the **RETURN** trap is executed before execution resumes. When a function completes, the values of the positional parameters and the special parameter **#** are restored to the values they had prior to the function's execution.

Function names and definitions may be listed with the **−f** option to the **declare** or **typeset** builtin commands. The **−F** option to **declare** or **typeset** will list the function names only (and optionally the source file and line number, if the **extdebug** shell option is enabled). Functions may be exported so that subshells automatically have them defined with the **−f** option to the **export** builtin. A function definition may be deleted using the **−f** option to the **unset** builtin. Note that shell functions and variables with the same name may result in multiple identically-named entries in the environment passed to the shell's children. Care should be taken in cases where this may cause a problem.

Functions may be recursive. The **FUNCNEST** variable may be used to limit the depth of the function call stack and restrict the number of function invocations. By default, no limit is imposed on the number of recursive calls.

ARITHMETIC EVALUATION

The shell allows arithmetic expressions to be evaluated, under certain circumstances (see the **let** and **declare** builtin commands, the **((** compound command, and **Arithmetic Expansion**). Evaluation is done in fixed-width integers with no check for overflow, though division by 0 is trapped and flagged as an error. The operators and their precedence, associativity, and values are the same as in the C language. The following list of operators is grouped into levels of equal-precedence operators. The levels are listed in order of decreasing precedence.

id++ *id*−−
: variable post-increment and post-decrement

++*id* −−*id*
: variable pre-increment and pre-decrement

− + : unary minus and plus

! ~ : logical and bitwise negation

****** : exponentiation

*** / %** : multiplication, division, remainder

+ − : addition, subtraction

<< >> : left and right bitwise shifts

<= >= < >
: comparison

== != : equality and inequality

& : bitwise AND

^ : bitwise exclusive OR

| : bitwise OR

&& : logical AND

|| : logical OR

*expr***?***expr***:***expr*
conditional operator
= *= /= %= += −= <<= >>= &= ^= |=
assignment
expr1 **,** *expr2*
comma

Shell variables are allowed as operands; parameter expansion is performed before the expression is evaluated. Within an expression, shell variables may also be referenced by name without using the parameter expansion syntax. A shell variable that is null or unset evaluates to 0 when referenced by name without using the parameter expansion syntax. The value of a variable is evaluated as an arithmetic expression when it is referenced, or when a variable which has been given the *integer* attribute using **declare -i** is assigned a value. A null value evaluates to 0. A shell variable need not have its *integer* attribute turned on to be used in an expression.

Constants with a leading 0 are interpreted as octal numbers. A leading 0x or 0X denotes hexadecimal. Otherwise, numbers take the form [*base#*]n, where the optional *base* is a decimal number between 2 and 64 representing the arithmetic base, and *n* is a number in that base. If *base#* is omitted, then base 10 is used. When specifying *n*, the digits greater than 9 are represented by the lowercase letters, the uppercase letters, @, and _, in that order. If *base* is less than or equal to 36, lowercase and uppercase letters may be used interchangeably to represent numbers between 10 and 35.

Operators are evaluated in order of precedence. Sub-expressions in parentheses are evaluated first and may override the precedence rules above.

CONDITIONAL EXPRESSIONS

Conditional expressions are used by the **[[** compound command and the **test** and **[** builtin commands to test file attributes and perform string and arithmetic comparisons. Expressions are formed from the following unary or binary primaries. **Bash** handles several filenames specially when they are used in expressions. If the operating system on which **bash** is running provides these special files, bash will use them; otherwise it will emulate them internally with this behavior: If any *file* argument to one of the primaries is of the form */dev/fd/n*, then file descriptor *n* is checked. If the *file* argument to one of the primaries is one of */dev/stdin*, */dev/stdout*, or */dev/stderr*, file descriptor 0, 1, or 2, respectively, is checked.

Unless otherwise specified, primaries that operate on files follow symbolic links and operate on the target of the link, rather than the link itself.

When used with **[[**, the < and > operators sort lexicographically using the current locale. The **test** command sorts using ASCII ordering.

−a *file* True if *file* exists.
−b *file* True if *file* exists and is a block special file.
−c *file* True if *file* exists and is a character special file.
−d *file* True if *file* exists and is a directory.
−e *file* True if *file* exists.
−f *file* True if *file* exists and is a regular file.
−g *file* True if *file* exists and is set-group-id.
−h *file* True if *file* exists and is a symbolic link.
−k *file* True if *file* exists and its ‘‘sticky’’ bit is set.
−p *file* True if *file* exists and is a named pipe (FIFO).
−r *file* True if *file* exists and is readable.
−s *file* True if *file* exists and has a size greater than zero.
−t *fd* True if file descriptor *fd* is open and refers to a terminal.
−u *file* True if *file* exists and its set-user-id bit is set.
−w *file* True if *file* exists and is writable.
−x *file* True if *file* exists and is executable.
−G *file* True if *file* exists and is owned by the effective group id.

−L *file* True if *file* exists and is a symbolic link.
−N *file* True if *file* exists and has been modified since it was last read.
−O *file* True if *file* exists and is owned by the effective user id.
−S *file* True if *file* exists and is a socket.
file1 **−ef** *file2*
True if *file1* and *file2* refer to the same device and inode numbers.
file1 **−nt** *file2*
True if *file1* is newer (according to modification date) than *file2*, or if *file1* exists and *file2* does not.
file1 **−ot** *file2*
True if *file1* is older than *file2*, or if *file2* exists and *file1* does not.
−o *optname*
True if the shell option *optname* is enabled. See the list of options under the description of the **−o** option to the **set** builtin below.
−v *varname*
True if the shell variable *varname* is set (has been assigned a value).
−R *varname*
True if the shell variable *varname* is set and is a name reference.
−z *string*
True if the length of *string* is zero.
string
−n *string*
True if the length of *string* is non-zero.

string1 **==** *string2*
string1 **=** *string2*
True if the strings are equal. **=** should be used with the **test** command for POSIX conformance. When used with the **[[** command, this performs pattern matching as described above (**Compound Commands**).

string1 **!=** *string2*
True if the strings are not equal.

string1 **<** *string2*
True if *string1* sorts before *string2* lexicographically.

string1 **>** *string2*
True if *string1* sorts after *string2* lexicographically.

arg1 **OP** *arg2*
OP is one of **−eq**, **−ne**, **−lt**, **−le**, **−gt**, or **−ge**. These arithmetic binary operators return true if *arg1* is equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to *arg2*, respectively. *Arg1* and *arg2* may be positive or negative integers.

SIMPLE COMMAND EXPANSION

When a simple command is executed, the shell performs the following expansions, assignments, and redirections, from left to right.

1. The words that the parser has marked as variable assignments (those preceding the command name) and redirections are saved for later processing.
2. The words that are not variable assignments or redirections are expanded. If any words remain after expansion, the first word is taken to be the name of the command and the remaining words are the arguments.
3. Redirections are performed as described above under **REDIRECTION**.
4. The text after the **=** in each variable assignment undergoes tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote removal before being assigned to the variable.

If no command name results, the variable assignments affect the current shell environment. Otherwise, the

variables are added to the environment of the executed command and do not affect the current shell environment. If any of the assignments attempts to assign a value to a readonly variable, an error occurs, and the command exits with a non-zero status.

If no command name results, redirections are performed, but do not affect the current shell environment. A redirection error causes the command to exit with a non-zero status.

If there is a command name left after expansion, execution proceeds as described below. Otherwise, the command exits. If one of the expansions contained a command substitution, the exit status of the command is the exit status of the last command substitution performed. If there were no command substitutions, the command exits with a status of zero.

COMMAND EXECUTION

After a command has been split into words, if it results in a simple command and an optional list of arguments, the following actions are taken.

If the command name contains no slashes, the shell attempts to locate it. If there exists a shell function by that name, that function is invoked as described above in **FUNCTIONS**. If the name does not match a function, the shell searches for it in the list of shell builtins. If a match is found, that builtin is invoked.

If the name is neither a shell function nor a builtin, and contains no slashes, **bash** searches each element of the **PATH** for a directory containing an executable file by that name. **Bash** uses a hash table to remember the full pathnames of executable files (see **hash** under **SHELL BUILTIN COMMANDS** below). A full search of the directories in **PATH** is performed only if the command is not found in the hash table. If the search is unsuccessful, the shell searches for a defined shell function named **command_not_found_handle**. If that function exists, it is invoked with the original command and the original command's arguments as its arguments, and the function's exit status becomes the exit status of the shell. If that function is not defined, the shell prints an error message and returns an exit status of 127.

If the search is successful, or if the command name contains one or more slashes, the shell executes the named program in a separate execution environment. Argument 0 is set to the name given, and the remaining arguments to the command are set to the arguments given, if any.

If this execution fails because the file is not in executable format, and the file is not a directory, it is assumed to be a *shell script*, a file containing shell commands. A subshell is spawned to execute it. This subshell reinitializes itself, so that the effect is as if a new shell had been invoked to handle the script, with the exception that the locations of commands remembered by the parent (see **hash** below under **SHELL BUILTIN COMMANDS**) are retained by the child.

If the program is a file beginning with **#!**, the remainder of the first line specifies an interpreter for the program. The shell executes the specified interpreter on operating systems that do not handle this executable format themselves. The arguments to the interpreter consist of a single optional argument following the interpreter name on the first line of the program, followed by the name of the program, followed by the command arguments, if any.

COMMAND EXECUTION ENVIRONMENT

The shell has an *execution environment*, which consists of the following:

- open files inherited by the shell at invocation, as modified by redirections supplied to the **exec** builtin
- the current working directory as set by **cd**, **pushd**, or **popd**, or inherited by the shell at invocation
- the file creation mode mask as set by **umask** or inherited from the shell's parent
- current traps set by **trap**
- shell parameters that are set by variable assignment or with **set** or inherited from the shell's parent in the environment
- shell functions defined during execution or inherited from the shell's parent in the environment
- options enabled at invocation (either by default or with command-line arguments) or by **set**

- options enabled by **shopt**
- shell aliases defined with **alias**
- various process IDs, including those of background jobs, the value of **$$**, and the value of **PPID**

When a simple command other than a builtin or shell function is to be executed, it is invoked in a separate execution environment that consists of the following. Unless otherwise noted, the values are inherited from the shell.

- the shell's open files, plus any modifications and additions specified by redirections to the command
- the current working directory
- the file creation mode mask
- shell variables and functions marked for export, along with variables exported for the command, passed in the environment
- traps caught by the shell are reset to the values inherited from the shell's parent, and traps ignored by the shell are ignored

A command invoked in this separate environment cannot affect the shell's execution environment.

Command substitution, commands grouped with parentheses, and asynchronous commands are invoked in a subshell environment that is a duplicate of the shell environment, except that traps caught by the shell are reset to the values that the shell inherited from its parent at invocation. Builtin commands that are invoked as part of a pipeline are also executed in a subshell environment. Changes made to the subshell environment cannot affect the shell's execution environment.

Subshells spawned to execute command substitutions inherit the value of the **−e** option from the parent shell. When not in *posix* mode, **bash** clears the **−e** option in such subshells.

If a command is followed by a **&** and job control is not active, the default standard input for the command is the empty file */dev/null*. Otherwise, the invoked command inherits the file descriptors of the calling shell as modified by redirections.

ENVIRONMENT

When a program is invoked it is given an array of strings called the *environment*. This is a list of *name−value* pairs, of the form *name=value*.

The shell provides several ways to manipulate the environment. On invocation, the shell scans its own environment and creates a parameter for each name found, automatically marking it for *export* to child processes. Executed commands inherit the environment. The **export** and **declare −x** commands allow parameters and functions to be added to and deleted from the environment. If the value of a parameter in the environment is modified, the new value becomes part of the environment, replacing the old. The environment inherited by any executed command consists of the shell's initial environment, whose values may be modified in the shell, less any pairs removed by the **unset** command, plus any additions via the **export** and **declare −x** commands.

The environment for any *simple command* or function may be augmented temporarily by prefixing it with parameter assignments, as described above in **PARAMETERS**. These assignment statements affect only the environment seen by that command.

If the **−k** option is set (see the **set** builtin command below), then *all* parameter assignments are placed in the environment for a command, not just those that precede the command name.

When **bash** invokes an external command, the variable **_** is set to the full filename of the command and passed to that command in its environment.

EXIT STATUS

The exit status of an executed command is the value returned by the *waitpid* system call or equivalent function. Exit statuses fall between 0 and 255, though, as explained below, the shell may use values above 125 specially. Exit statuses from shell builtins and compound commands are also limited to this range. Under

certain circumstances, the shell will use special values to indicate specific failure modes.

For the shell's purposes, a command which exits with a zero exit status has succeeded. An exit status of zero indicates success. A non-zero exit status indicates failure. When a command terminates on a fatal signal *N*, **bash** uses the value of 128+*N* as the exit status.

If a command is not found, the child process created to execute it returns a status of 127. If a command is found but is not executable, the return status is 126.

If a command fails because of an error during expansion or redirection, the exit status is greater than zero.

Shell builtin commands return a status of 0 (*true*) if successful, and non-zero (*false*) if an error occurs while they execute. All builtins return an exit status of 2 to indicate incorrect usage, generally invalid options or missing arguments.

Bash itself returns the exit status of the last command executed, unless a syntax error occurs, in which case it exits with a non-zero value. See also the **exit** builtin command below.

SIGNALS

When **bash** is interactive, in the absence of any traps, it ignores **SIGTERM** (so that **kill 0** does not kill an interactive shell), and **SIGINT** is caught and handled (so that the **wait** builtin is interruptible). In all cases, **bash** ignores **SIGQUIT**. If job control is in effect, **bash** ignores **SIGTTIN**, **SIGTTOU**, and **SIGTSTP**.

Non-builtin commands run by **bash** have signal handlers set to the values inherited by the shell from its parent. When job control is not in effect, asynchronous commands ignore **SIGINT** and **SIGQUIT** in addition to these inherited handlers. Commands run as a result of command substitution ignore the keyboard-generated job control signals **SIGTTIN**, **SIGTTOU**, and **SIGTSTP**.

The shell exits by default upon receipt of a **SIGHUP**. Before exiting, an interactive shell resends the **SIGHUP** to all jobs, running or stopped. Stopped jobs are sent **SIGCONT** to ensure that they receive the **SIGHUP**. To prevent the shell from sending the signal to a particular job, it should be removed from the jobs table with the **disown** builtin (see **SHELL BUILTIN COMMANDS** below) or marked to not receive **SIGHUP** using **disown −h**.

If the **huponexit** shell option has been set with **shopt**, **bash** sends a **SIGHUP** to all jobs when an interactive login shell exits.

If **bash** is waiting for a command to complete and receives a signal for which a trap has been set, the trap will not be executed until the command completes. When **bash** is waiting for an asynchronous command via the **wait** builtin, the reception of a signal for which a trap has been set will cause the **wait** builtin to return immediately with an exit status greater than 128, immediately after which the trap is executed.

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 operating system kernel'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:

```
[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 operating 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, if the user so specifies with `stty tostop`, write to the

terminal. Background processes which attempt to read from (write to when `stty tostop` is in effect) the terminal are sent a **SIGTTIN (SIGTTOU)** signal by the kernel's terminal driver, which, unless caught, suspends the process.

If the operating system on which **bash** is running supports job control, **bash** contains facilities to use it. Typing the *suspend* character (typically **^Z**, Control-Z) while a process is running causes that process to be stopped and returns control 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**. The user 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 **^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 specification (*jobspec*). 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 stopped while it was in the foreground or started in the background. The *previous job* may be referenced using **%−**. If there is only a single job, **%+** and **%−** can both be used to refer to that job. In output pertaining to jobs (e.g., the output of the **jobs** command), the current job is always flagged with a **+**, and the previous job with a **−**. A single % (with no accompanying job specification) also refers to the current job.

Simply naming a job can be used to bring it into the foreground: **%1** is a synonym for **"fg %1"**, bringing job 1 from the background into the foreground. Similarly, **"%1 &"** resumes job 1 in the background, equivalent to **"bg %1"**.

The shell learns immediately whenever a job changes state. Normally, **bash** waits until it is about to print a prompt before reporting changes in a job's status so as to not interrupt any other output. If the **−b** option to the **set** builtin command is enabled, **bash** reports such changes immediately. Any trap on **SIGCHLD** is executed for each child that exits.

If an attempt to exit **bash** is made while jobs are stopped (or, if the **checkjobs** shell option has been enabled using the **shopt** builtin, running), the shell prints a warning message, and, if the **checkjobs** option is enabled, lists the jobs and their statuses. The **jobs** command may then be used to inspect their status. If a second attempt to exit is made without an intervening command, the shell does not print another warning, and any stopped jobs are terminated.

PROMPTING

When executing interactively, **bash** displays the primary prompt **PS1** when it is ready to read a command, and the secondary prompt **PS2** when it needs more input to complete a command. **Bash** displays **PS0** after it reads a command but before executing it. **Bash** allows these prompt strings to be customized by inserting a number of backslash-escaped special characters that are decoded as follows:

\a an ASCII bell character (07)

\d the date in "Weekday Month Date" format (e.g., "Tue May 26")

\D{*format***}**
: the *format* is passed to *strftime*(3) and the result is inserted into the prompt string; an empty *format* results in a locale-specific time representation. The braces are required

\e an ASCII escape character (033)

\h the hostname up to the first '.'

\H the hostname

\j the number of jobs currently managed by the shell

\l the basename of the shell's terminal device name

\n newline

\r carriage return

\s the name of the shell, the basename of **$0** (the portion following the final slash)
\t the current time in 24-hour HH:MM:SS format
\T the current time in 12-hour HH:MM:SS format
\@ the current time in 12-hour am/pm format
\A the current time in 24-hour HH:MM format
\u the username of the current user
\v the version of **bash** (e.g., 2.00)
\V the release of **bash**, version + patch level (e.g., 2.00.0)
\w the current working directory, with **$HOME** abbreviated with a tilde (uses the value of the **PROMPT_DIRTRIM** variable)
\W the basename of the current working directory, with **$HOME** abbreviated with a tilde
\! the history number of this command
\# the command number of this command
\$ if the effective UID is 0, a **#**, otherwise a **$**
\nnn the character corresponding to the octal number *nnn*
\\ a backslash
\[begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
\] end a sequence of non-printing characters

The command number and the history number are usually different: the history number of a command is its position in the history list, which may include commands restored from the history file (see **HISTORY** below), while the command number is the position in the sequence of commands executed during the current shell session. After the string is decoded, it is expanded via parameter expansion, command substitution, arithmetic expansion, and quote removal, subject to the value of the **promptvars** shell option (see the description of the **shopt** command under **SHELL BUILTIN COMMANDS** below).

READLINE

This is the library that handles reading input when using an interactive shell, unless the **−−noediting** option is given at shell invocation. Line editing is also used when using the **−e** option to the **read** builtin. By default, the line editing commands are similar to those of Emacs. A vi-style line editing interface is also available. Line editing can be enabled at any time using the **−o emacs** or **−o vi** options to the **set** builtin (see **SHELL BUILTIN COMMANDS** below). To turn off line editing after the shell is running, use the **+o emacs** or **+o vi** options to the **set** builtin.

Readline Notation

In this section, the Emacs-style notation is used to denote keystrokes. Control keys are denoted by C−*key*, e.g., C−n means Control−N. Similarly, *meta* keys are denoted by M−*key*, so M−x means Meta−X. (On keyboards without a *meta* key, M−*x* means ESC *x*, i.e., press the Escape key then the *x* key. This makes ESC the *meta prefix*. The combination M−C−*x* means ESC−Control−*x*, or press the Escape key then hold the Control key while pressing the *x* key.)

Readline commands may be given numeric *arguments*, which normally act as a repeat count. Sometimes, however, it is the sign of the argument that is significant. Passing a negative argument to a command that acts in the forward direction (e.g., **kill−line**) causes that command to act in a backward direction. Commands whose behavior with arguments deviates from this are noted below.

When a command is described as *killing* text, the text deleted is saved for possible future retrieval (*yanking*). The killed text is saved in a *kill ring*. Consecutive kills cause the text to be accumulated into one unit, which can be yanked all at once. Commands which do not kill text separate the chunks of text on the kill ring.

Readline Initialization

Readline is customized by putting commands in an initialization file (the *inputrc* file). The name of this file is taken from the value of the **INPUTRC** variable. If that variable is unset, the default is *~/.inputrc*. When a program which uses the readline library starts up, the initialization file is read, and the key bindings and variables are set. There are only a few basic constructs allowed in the readline initialization file. Blank lines are ignored. Lines beginning with a **#** are comments. Lines beginning with a **$** indicate conditional

constructs. Other lines denote key bindings and variable settings.

The default key-bindings may be changed with an *inputrc* file. Other programs that use this library may add their own commands and bindings.

For example, placing

M−Control−u: universal−argument

or

C−Meta−u: universal−argument

into the *inputrc* would make M−C−u execute the readline command *universal−argument*.

The following symbolic character names are recognized: *RUBOUT*, *DEL*, *ESC*, *LFD*, *NEWLINE*, *RET*, *RETURN*, *SPC*, *SPACE*, and *TAB*.

In addition to command names, readline allows keys to be bound to a string that is inserted when the key is pressed (a *macro*).

Readline Key Bindings

The syntax for controlling key bindings in the *inputrc* file is simple. All that is required is the name of the command or the text of a macro and a key sequence to which it should be bound. The name may be specified in one of two ways: as a symbolic key name, possibly with *Meta−* or *Control−* prefixes, or as a key sequence.

When using the form **keyname**:*function−name* or *macro*, *keyname* is the name of a key spelled out in English. For example:

```
Control-u: universal−argument
Meta-Rubout: backward-kill-word
Control-o: "> output"
```

In the above example, *C−u* is bound to the function **universal−argument**, *M−DEL* is bound to the function **backward−kill−word**, and *C−o* is bound to run the macro expressed on the right hand side (that is, to insert the text `> output` into the line).

In the second form, **"keyseq"**:*function−name* or *macro*, **keyseq** differs from **keyname** above in that strings denoting an entire key sequence may be specified by placing the sequence within double quotes. Some GNU Emacs style key escapes can be used, as in the following example, but the symbolic character names are not recognized.

```
"\C−u": universal−argument
"\C−x\C−r": re−read−init−file
"\e[11~": "Function Key 1"
```

In this example, *C−u* is again bound to the function **universal−argument**. *C−x C−r* is bound to the function **re−read−init−file**, and *ESC [1 1 ~* is bound to insert the text `Function Key 1`.

The full set of GNU Emacs style escape sequences is

\C− control prefix
\M− meta prefix
\e an escape character
\\ backslash
\" literal "
\' literal '

In addition to the GNU Emacs style escape sequences, a second set of backslash escapes is available:

\a alert (bell)
\b backspace
\d delete
\f form feed
\n newline

\r	carriage return
\t	horizontal tab
\v	vertical tab
nnn	the eight-bit character whose value is the octal value *nnn* (one to three digits)
\x*HH*	the eight-bit character whose value is the hexadecimal value *HH* (one or two hex digits)

When entering the text of a macro, single or double quotes must be used to indicate a macro definition. Unquoted text is assumed to be a function name. In the macro body, the backslash escapes described above are expanded. Backslash will quote any other character in the macro text, including " and '.

Bash allows the current readline key bindings to be displayed or modified with the **bind** builtin command. The editing mode may be switched during interactive use by using the **−o** option to the **set** builtin command (see **SHELL BUILTIN COMMANDS** below).

Readline Variables

Readline has variables that can be used to further customize its behavior. A variable may be set in the *inputrc* file with a statement of the form

set *variable−name value*

Except where noted, readline variables can take the values **On** or **Off** (without regard to case). Unrecognized variable names are ignored. When a variable value is read, empty or null values, "on" (case-insensitive), and "1" are equivalent to **On**. All other values are equivalent to **Off**. The variables and their default values are:

bell−style (audible)
Controls what happens when readline wants to ring the terminal bell. If set to **none**, readline never rings the bell. If set to **visible**, readline uses a visible bell if one is available. If set to **audible**, readline attempts to ring the terminal's bell.

bind−tty−special−chars (On)
If set to **On**, readline attempts to bind the control characters treated specially by the kernel's terminal driver to their readline equivalents.

blink−matching−paren (Off)
If set to **On**, readline attempts to briefly move the cursor to an opening parenthesis when a closing parenthesis is inserted.

colored−completion−prefix (Off)
If set to **On**, when listing completions, readline displays the common prefix of the set of possible completions using a different color. The color definitions are taken from the value of the **LS_COLORS** environment variable.

colored−stats (Off)
If set to **On**, readline displays possible completions using different colors to indicate their file type. The color definitions are taken from the value of the **LS_COLORS** environment variable.

comment−begin ("#")
The string that is inserted when the readline **insert−comment** command is executed. This command is bound to **M−#** in emacs mode and to **#** in vi command mode.

completion−display−width (-1)
The number of screen columns used to display possible matches when performing completion. The value is ignored if it is less than 0 or greater than the terminal screen width. A value of 0 will cause matches to be displayed one per line. The default value is -1.

completion−ignore−case (Off)
If set to **On**, readline performs filename matching and completion in a case−insensitive fashion.

completion−map−case (Off)
If set to **On**, and **completion−ignore−case** is enabled, readline treats hyphens (−) and underscores (_) as equivalent when performing case−insensitive filename matching and completion.

completion−prefix−display−length (0)
The length in characters of the common prefix of a list of possible completions that is displayed without modification. When set to a value greater than zero, common prefixes longer than this value are replaced with an ellipsis when displaying possible completions.

completion−query−items (100)
This determines when the user is queried about viewing the number of possible completions generated by the **possible−completions** command. It may be set to any integer value greater than or equal to zero. If the number of possible completions is greater than or equal to the value of this variable, the user is asked whether or not he wishes to view them; otherwise they are simply listed on the terminal.

convert−meta (On)
If set to **On**, readline will convert characters with the eighth bit set to an ASCII key sequence by stripping the eighth bit and prefixing an escape character (in effect, using escape as the *meta prefix*). The default is *On*, but readline will set it to *Off* if the locale contains eight-bit characters.

disable−completion (Off)
If set to **On**, readline will inhibit word completion. Completion characters will be inserted into the line as if they had been mapped to **self-insert**.

echo−control−characters (On)
When set to **On**, on operating systems that indicate they support it, readline echoes a character corresponding to a signal generated from the keyboard.

editing−mode (emacs)
Controls whether readline begins with a set of key bindings similar to *Emacs* or *vi*. **editing−mode** can be set to either **emacs** or **vi**.

enable−bracketed−paste (Off)
When set to **On**, readline will configure the terminal in a way that will enable it to insert each paste into the editing buffer as a single string of characters, instead of treating each character as if it had been read from the keyboard. This can prevent pasted characters from being interpreted as editing commands.

enable−keypad (Off)
When set to **On**, readline will try to enable the application keypad when it is called. Some systems need this to enable the arrow keys.

enable−meta−key (On)
When set to **On**, readline will try to enable any meta modifier key the terminal claims to support when it is called. On many terminals, the meta key is used to send eight-bit characters.

expand−tilde (Off)
If set to **On**, tilde expansion is performed when readline attempts word completion.

history−preserve−point (Off)
If set to **On**, the history code attempts to place point at the same location on each history line retrieved with **previous-history** or **next-history**.

history−size (unset)
Set the maximum number of history entries saved in the history list. If set to zero, any existing history entries are deleted and no new entries are saved. If set to a value less than zero, the number of history entries is not limited. By default, the number of history entries is set to the value of the **HISTSIZE** shell variable. If an attempt is made to set *history−size* to a non-numeric value, the maximum number of history entries will be set to 500.

horizontal−scroll−mode (Off)
When set to **On**, makes readline use a single line for display, scrolling the input horizontally on a single screen line when it becomes longer than the screen width rather than wrapping to a new line.

input−meta (Off)
If set to **On**, readline will enable eight-bit input (that is, it will not strip the eighth bit from the characters it reads), regardless of what the terminal claims it can support. The name **meta−flag** is a synonym for this variable. The default is *Off*, but readline will set it to *On* if the locale contains eight-bit characters.

isearch−terminators (“C−[C−J”)
The string of characters that should terminate an incremental search without subsequently executing the character as a command. If this variable has not been given a value, the characters *ESC* and *C−J* will terminate an incremental search.

keymap (emacs)
Set the current readline keymap. The set of valid keymap names is *emacs*, *emacs−standard*, *emacs−meta*, *emacs−ctlx*, *vi*, *vi−command*, and *vi−insert*. *vi* is equivalent to *vi−command*; *emacs* is equivalent to *emacs−standard*. The default value is *emacs*; the value of **editing−mode** also affects the default keymap.

emacs−mode−string (@)
This string is displayed immediately before the last line of the primary prompt when emacs editing mode is active. The value is expanded like a key binding, so the standard set of meta- and control prefixes and backslash escape sequences is available. Use the \1 and \2 escapes to begin and end sequences of non-printing characters, which can be used to embed a terminal control sequence into the mode string.

keyseq−timeout (500)
Specifies the duration *readline* will wait for a character when reading an ambiguous key sequence (one that can form a complete key sequence using the input read so far, or can take additional input to complete a longer key sequence). If no input is received within the timeout, *readline* will use the shorter but complete key sequence. The value is specified in milliseconds, so a value of 1000 means that *readline* will wait one second for additional input. If this variable is set to a value less than or equal to zero, or to a non-numeric value, *readline* will wait until another key is pressed to decide which key sequence to complete.

mark−directories (On)
If set to **On**, completed directory names have a slash appended.

mark−modified−lines (Off)
If set to **On**, history lines that have been modified are displayed with a preceding asterisk (*****).

mark−symlinked−directories (Off)
If set to **On**, completed names which are symbolic links to directories have a slash appended (subject to the value of **mark−directories**).

match−hidden−files (On)
This variable, when set to **On**, causes readline to match files whose names begin with a '.' (hidden files) when performing filename completion. If set to **Off**, the leading '.' must be supplied by the user in the filename to be completed.

menu−complete−display−prefix (Off)
If set to **On**, menu completion displays the common prefix of the list of possible completions (which may be empty) before cycling through the list.

output−meta (Off)
If set to **On**, readline will display characters with the eighth bit set directly rather than as a meta-prefixed escape sequence. The default is *Off*, but readline will set it to *On* if the locale contains eight-bit characters.

page−completions (On)
If set to **On**, readline uses an internal *more*-like pager to display a screenful of possible completions at a time.

print−completions−horizontally (Off)
If set to **On**, readline will display completions with matches sorted horizontally in alphabetical order, rather than down the screen.

revert−all−at−newline (Off)
If set to **On**, readline will undo all changes to history lines before returning when **accept−line** is executed. By default, history lines may be modified and retain individual undo lists across calls to **readline**.

show−all−if−ambiguous (Off)
This alters the default behavior of the completion functions. If set to **On**, words which have more than one possible completion cause the matches to be listed immediately instead of ringing the bell.

show−all−if−unmodified (Off)
This alters the default behavior of the completion functions in a fashion similar to **show−all−if−ambiguous**. If set to **On**, words which have more than one possible completion

without any possible partial completion (the possible completions don't share a common prefix) cause the matches to be listed immediately instead of ringing the bell.

show−mode−in−prompt (Off)
If set to **On**, add a character to the beginning of the prompt indicating the editing mode: emacs (@), vi command (:) or vi insertion (+).

skip−completed−text (Off)
If set to **On**, this alters the default completion behavior when inserting a single match into the line. It's only active when performing completion in the middle of a word. If enabled, readline does not insert characters from the completion that match characters after point in the word being completed, so portions of the word following the cursor are not duplicated.

vi−cmd−mode−string ((cmd))
This string is displayed immediately before the last line of the primary prompt when vi editing mode is active and in command mode. The value is expanded like a key binding, so the standard set of meta- and control prefixes and backslash escape sequences is available. Use the \1 and \2 escapes to begin and end sequences of non-printing characters, which can be used to embed a terminal control sequence into the mode string.

vi−ins−mode−string ((ins))
This string is displayed immediately before the last line of the primary prompt when vi editing mode is active and in insertion mode. The value is expanded like a key binding, so the standard set of meta- and control prefixes and backslash escape sequences is available. Use the \1 and \2 escapes to begin and end sequences of non-printing characters, which can be used to embed a terminal control sequence into the mode string.

visible−stats (Off)
If set to **On**, a character denoting a file's type as reported by *stat*(2) is appended to the filename when listing possible completions.

Readline Conditional Constructs

Readline implements a facility similar in spirit to the conditional compilation features of the C preprocessor which allows key bindings and variable settings to be performed as the result of tests. There are four parser directives used.

$if The **$if** construct allows bindings to be made based on the editing mode, the terminal being used, or the application using readline. The text of the test extends to the end of the line; no characters are required to isolate it.

mode The **mode=** form of the **$if** directive is used to test whether readline is in emacs or vi mode. This may be used in conjunction with the **set keymap** command, for instance, to set bindings in the *emacs−standard* and *emacs−ctlx* keymaps only if readline is starting out in emacs mode.

term The **term=** form may be used to include terminal-specific key bindings, perhaps to bind the key sequences output by the terminal's function keys. The word on the right side of the **=** is tested against both the full name of the terminal and the portion of the terminal name before the first **−**. This allows *sun* to match both *sun* and *sun−cmd*, for instance.

application
The **application** construct is used to include application-specific settings. Each program using the readline library sets the *application name*, and an initialization file can test for a particular value. This could be used to bind key sequences to functions useful for a specific program. For instance, the following command adds a key sequence that quotes the current or previous word in **bash**:

```
$if Bash
# Quote the current or previous word
"\C−xq": "\eb\"\ef\""
$endif
```

$endif This command, as seen in the previous example, terminates an **$if** command.

$else Commands in this branch of the **$if** directive are executed if the test fails.

$include
This directive takes a single filename as an argument and reads commands and bindings from that file. For example, the following directive would read */etc/inputrc*:

$include */etc/inputrc*

Searching

Readline provides commands for searching through the command history (see **HISTORY** below) for lines containing a specified string. There are two search modes: *incremental* and *non-incremental*.

Incremental searches begin before the user has finished typing the search string. As each character of the search string is typed, readline displays the next entry from the history matching the string typed so far. An incremental search requires only as many characters as needed to find the desired history entry. The characters present in the value of the **isearch-terminators** variable are used to terminate an incremental search. If that variable has not been assigned a value the Escape and Control-J characters will terminate an incremental search. Control-G will abort an incremental search and restore the original line. When the search is terminated, the history entry containing the search string becomes the current line.

To find other matching entries in the history list, type Control-S or Control-R as appropriate. This will search backward or forward in the history for the next entry matching the search string typed so far. Any other key sequence bound to a readline command will terminate the search and execute that command. For instance, a *newline* will terminate the search and accept the line, thereby executing the command from the history list.

Readline remembers the last incremental search string. If two Control-Rs are typed without any intervening characters defining a new search string, any remembered search string is used.

Non-incremental searches read the entire search string before starting to search for matching history lines. The search string may be typed by the user or be part of the contents of the current line.

Readline Command Names

The following is a list of the names of the commands and the default key sequences to which they are bound. Command names without an accompanying key sequence are unbound by default. In the following descriptions, *point* refers to the current cursor position, and *mark* refers to a cursor position saved by the **set−mark** command. The text between the point and mark is referred to as the *region*.

Commands for Moving

beginning−of−line (C−a)
Move to the start of the current line.

end−of−line (C−e)
Move to the end of the line.

forward−char (C−f)
Move forward a character.

backward−char (C−b)
Move back a character.

forward−word (M−f)
Move forward to the end of the next word. Words are composed of alphanumeric characters (letters and digits).

backward−word (M−b)
Move back to the start of the current or previous word. Words are composed of alphanumeric characters (letters and digits).

shell−forward−word
Move forward to the end of the next word. Words are delimited by non-quoted shell metacharacters.

shell−backward−word
Move back to the start of the current or previous word. Words are delimited by non-quoted shell metacharacters.

clear−screen (C−l)
Clear the screen leaving the current line at the top of the screen. With an argument, refresh the current line without clearing the screen.

redraw−current−line
Refresh the current line.

Commands for Manipulating the History

accept−line (Newline, Return)
Accept the line regardless of where the cursor is. If this line is non-empty, add it to the history list according to the state of the **HISTCONTROL** variable. If the line is a modified history line, then restore the history line to its original state.

previous−history (C−p)
Fetch the previous command from the history list, moving back in the list.

next−history (C−n)
Fetch the next command from the history list, moving forward in the list.

beginning−of−history (M−<)
Move to the first line in the history.

end−of−history (M−>)
Move to the end of the input history, i.e., the line currently being entered.

reverse−search−history (C−r)
Search backward starting at the current line and moving ‘up’ through the history as necessary. This is an incremental search.

forward−search−history (C−s)
Search forward starting at the current line and moving ‘down’ through the history as necessary. This is an incremental search.

non−incremental−reverse−search−history (M−p)
Search backward through the history starting at the current line using a non-incremental search for a string supplied by the user.

non−incremental−forward−search−history (M−n)
Search forward through the history using a non-incremental search for a string supplied by the user.

history−search−forward
Search forward through the history for the string of characters between the start of the current line and the point. This is a non-incremental search.

history−search−backward
Search backward through the history for the string of characters between the start of the current line and the point. This is a non-incremental search.

yank−nth−arg (M−C−y)
Insert the first argument to the previous command (usually the second word on the previous line) at point. With an argument *n*, insert the *n*th word from the previous command (the words in the previous command begin with word 0). A negative argument inserts the *n*th word from the end of the previous command. Once the argument *n* is computed, the argument is extracted as if the "!*n*" history expansion had been specified.

yank−last−arg (M−., M−_)
Insert the last argument to the previous command (the last word of the previous history entry). With a numeric argument, behave exactly like **yank−nth−arg**. Successive calls to **yank−last−arg** move back through the history list, inserting the last word (or the word specified by the argument to the first call) of each line in turn. Any numeric argument supplied to these successive calls determines the direction to move through the history. A negative argument switches the direction through the history (back or forward). The history expansion facilities are used to extract the last word, as if the "!$" history expansion had been specified.

shell−expand−line (M−C−e)
Expand the line as the shell does. This performs alias and history expansion as well as all of the shell word expansions. See **HISTORY EXPANSION** below for a description of history expansion.

history−expand−line (M−^)
Perform history expansion on the current line. See **HISTORY EXPANSION** below for a description of history expansion.

magic−space
Perform history expansion on the current line and insert a space. See **HISTORY EXPANSION** below for a description of history expansion.

alias−expand−line
Perform alias expansion on the current line. See **ALIASES** above for a description of alias expansion.

history−and−alias−expand−line
Perform history and alias expansion on the current line.

insert−last−argument (M−., M−_)
A synonym for **yank−last−arg**.

operate−and−get−next (C−o)
Accept the current line for execution and fetch the next line relative to the current line from the history for editing. Any argument is ignored.

edit−and−execute−command (C−xC−e)
Invoke an editor on the current command line, and execute the result as shell commands. **Bash** attempts to invoke **$VISUAL**, **$EDITOR**, and *emacs* as the editor, in that order.

Commands for Changing Text

end−of−file **(usually C−d)**
The character indicating end-of-file as set, for example, by `stty`. If this character is read when there are no characters on the line, and point is at the beginning of the line, Readline interprets it as the end of input and returns **EOF**.

delete−char (C−d)
Delete the character at point. If this function is bound to the same character as the tty **EOF** character, as **C−d** commonly is, see above for the effects.

backward−delete−char (Rubout)
Delete the character behind the cursor. When given a numeric argument, save the deleted text on the kill ring.

forward−backward−delete−char
Delete the character under the cursor, unless the cursor is at the end of the line, in which case the character behind the cursor is deleted.

quoted−insert (C−q, C−v)
Add the next character typed to the line verbatim. This is how to insert characters like **C−q**, for example.

tab−insert (C−v TAB)
Insert a tab character.

self−insert (a, b, A, 1, !, ...)
Insert the character typed.

transpose−chars (C−t)
Drag the character before point forward over the character at point, moving point forward as well. If point is at the end of the line, then this transposes the two characters before point. Negative arguments have no effect.

transpose−words (M−t)
Drag the word before point past the word after point, moving point over that word as well. If point is at the end of the line, this transposes the last two words on the line.

upcase−word (M−u)
Uppercase the current (or following) word. With a negative argument, uppercase the previous word, but do not move point.

downcase−word (M−l)
Lowercase the current (or following) word. With a negative argument, lowercase the previous word, but do not move point.

capitalize−word (M−c)
Capitalize the current (or following) word. With a negative argument, capitalize the previous word, but do not move point.

overwrite−mode
Toggle overwrite mode. With an explicit positive numeric argument, switches to overwrite mode. With an explicit non-positive numeric argument, switches to insert mode. This command affects only **emacs** mode; **vi** mode does overwrite differently. Each call to *readline()* starts in insert mode. In overwrite mode, characters bound to **self−insert** replace the text at point rather than pushing the text to the right. Characters bound to **backward−delete−char** replace the character before point with a space. By default, this command is unbound.

Killing and Yanking

kill−line (C−k)
Kill the text from point to the end of the line.

backward−kill−line (C−x Rubout)
Kill backward to the beginning of the line.

unix−line−discard (C−u)
Kill backward from point to the beginning of the line. The killed text is saved on the kill-ring.

kill−whole−line
Kill all characters on the current line, no matter where point is.

kill−word (M−d)
Kill from point to the end of the current word, or if between words, to the end of the next word. Word boundaries are the same as those used by **forward−word**.

backward−kill−word (M−Rubout)
Kill the word behind point. Word boundaries are the same as those used by **backward−word**.

shell−kill−word
Kill from point to the end of the current word, or if between words, to the end of the next word. Word boundaries are the same as those used by **shell−forward−word**.

shell−backward−kill−word
Kill the word behind point. Word boundaries are the same as those used by **shell−backward−word**.

unix−word−rubout (C−w)
Kill the word behind point, using white space as a word boundary. The killed text is saved on the kill-ring.

unix−filename−rubout
Kill the word behind point, using white space and the slash character as the word boundaries. The killed text is saved on the kill-ring.

delete−horizontal−space (M−\)
Delete all spaces and tabs around point.

kill−region
Kill the text in the current region.

copy−region−as−kill
Copy the text in the region to the kill buffer.

copy−backward−word
Copy the word before point to the kill buffer. The word boundaries are the same as **backward−word**.

copy−forward−word
Copy the word following point to the kill buffer. The word boundaries are the same as **forward−word**.

yank (C−y)
Yank the top of the kill ring into the buffer at point.

yank−pop (M−y)
Rotate the kill ring, and yank the new top. Only works following **yank** or **yank−pop**.

Numeric Arguments

digit−argument (M−0, M−1, ..., M−−)
Add this digit to the argument already accumulating, or start a new argument. M−− starts a negative argument.

universal−argument
This is another way to specify an argument. If this command is followed by one or more digits, optionally with a leading minus sign, those digits define the argument. If the command is followed by digits, executing **universal−argument** again ends the numeric argument, but is otherwise ignored. As a special case, if this command is immediately followed by a character that is neither a digit nor minus sign, the argument count for the next command is multiplied by four. The argument count is initially one, so executing this function the first time makes the argument count four, a second time makes the argument count sixteen, and so on.

Completing

complete (TAB)
Attempt to perform completion on the text before point. **Bash** attempts completion treating the text as a variable (if the text begins with **$**), username (if the text begins with **~**), hostname (if the text begins with **@**), or command (including aliases and functions) in turn. If none of these produces a match, filename completion is attempted.

possible−completions (M−?)
List the possible completions of the text before point.

insert−completions (M−*)
Insert all completions of the text before point that would have been generated by **possible−completions**.

menu−complete
Similar to **complete**, but replaces the word to be completed with a single match from the list of possible completions. Repeated execution of **menu−complete** steps through the list of possible completions, inserting each match in turn. At the end of the list of completions, the bell is rung (subject to the setting of **bell−style**) and the original text is restored. An argument of *n* moves *n* positions forward in the list of matches; a negative argument may be used to move backward through the list. This command is intended to be bound to **TAB**, but is unbound by default.

menu−complete−backward
Identical to **menu−complete**, but moves backward through the list of possible completions, as if **menu−complete** had been given a negative argument. This command is unbound by default.

delete−char−or−list
Deletes the character under the cursor if not at the beginning or end of the line (like **delete−char**). If at the end of the line, behaves identically to **possible−completions**. This command is unbound by default.

complete−filename (M−/)
Attempt filename completion on the text before point.

possible−filename−completions (C−x /)
List the possible completions of the text before point, treating it as a filename.

complete−username (M−~)
Attempt completion on the text before point, treating it as a username.

possible−username−completions (C−x ~)
List the possible completions of the text before point, treating it as a username.

complete−variable (M−$)
Attempt completion on the text before point, treating it as a shell variable.

possible−variable−completions (C−x $)
List the possible completions of the text before point, treating it as a shell variable.

complete−hostname (M−@)
Attempt completion on the text before point, treating it as a hostname.

possible−hostname−completions (C−x @)
List the possible completions of the text before point, treating it as a hostname.

complete−command (M−!)
Attempt completion on the text before point, treating it as a command name. Command completion attempts to match the text against aliases, reserved words, shell functions, shell builtins, and finally executable filenames, in that order.

possible−command−completions (C−x !)
List the possible completions of the text before point, treating it as a command name.

dynamic−complete−history (M−TAB)
Attempt completion on the text before point, comparing the text against lines from the history list for possible completion matches.

dabbrev−expand
Attempt menu completion on the text before point, comparing the text against lines from the history list for possible completion matches.

complete−into−braces (M−{)
Perform filename completion and insert the list of possible completions enclosed within braces so the list is available to the shell (see **Brace Expansion** above).

Keyboard Macros

start−kbd−macro (C−x ()
Begin saving the characters typed into the current keyboard macro.

end−kbd−macro (C−x))
Stop saving the characters typed into the current keyboard macro and store the definition.

call−last−kbd−macro (C−x e)
Re-execute the last keyboard macro defined, by making the characters in the macro appear as if typed at the keyboard.

print−last−kbd−macro ()
Print the last keyboard macro defined in a format suitable for the *inputrc* file.

Miscellaneous

re−read−init−file (C−x C−r)
Read in the contents of the *inputrc* file, and incorporate any bindings or variable assignments found there.

abort (C−g)
Abort the current editing command and ring the terminal's bell (subject to the setting of **bell−style**).

do−uppercase−version (M−a, M−b, M−*x*, ...)
If the metafied character *x* is lowercase, run the command that is bound to the corresponding uppercase character.

prefix−meta (ESC)
Metafy the next character typed. **ESC f** is equivalent to **Meta−f**.

undo (C−_, C−x C−u)
Incremental undo, separately remembered for each line.

revert−line (M−r)
Undo all changes made to this line. This is like executing the **undo** command enough times to return the line to its initial state.

tilde−expand (M−&)
Perform tilde expansion on the current word.

set−mark (C−@, M−<space>)
Set the mark to the point. If a numeric argument is supplied, the mark is set to that position.

exchange−point−and−mark (C−x C−x)
Swap the point with the mark. The current cursor position is set to the saved position, and the old cursor position is saved as the mark.

character−search (C−])
A character is read and point is moved to the next occurrence of that character. A negative count searches for previous occurrences.

character−search−backward (M−C−])
A character is read and point is moved to the previous occurrence of that character. A negative count searches for subsequent occurrences.

skip−csi−sequence
Read enough characters to consume a multi-key sequence such as those defined for keys like Home and End. Such sequences begin with a Control Sequence Indicator (CSI), usually ESC−[. If this sequence is bound to "\[", keys producing such sequences will have no effect unless explicitly bound to a readline command, instead of inserting stray characters into the editing buffer. This is unbound by default, but usually bound to ESC−[.

insert−comment (M−#)
Without a numeric argument, the value of the readline **comment−begin** variable is inserted at the beginning of the current line. If a numeric argument is supplied, this command acts as a toggle: if the characters at the beginning of the line do not match the value of **comment−begin**, the value is inserted, otherwise the characters in **comment−begin** are deleted from the beginning of the line. In either case, the line is accepted as if a newline had been typed. The default value of **comment−begin** causes this command to make the current line a shell comment. If a numeric argument causes the comment character to be removed, the line will be executed by the shell.

glob−complete−word (M−g)
The word before point is treated as a pattern for pathname expansion, with an asterisk implicitly appended. This pattern is used to generate a list of matching filenames for possible completions.

glob−expand−word (C−x *)
The word before point is treated as a pattern for pathname expansion, and the list of matching filenames is inserted, replacing the word. If a numeric argument is supplied, an asterisk is appended before pathname expansion.

glob−list−expansions (C−x g)
The list of expansions that would have been generated by **glob−expand−word** is displayed, and the line is redrawn. If a numeric argument is supplied, an asterisk is appended before pathname expansion.

dump−functions
Print all of the functions and their key bindings to the readline output stream. If a numeric argument is supplied, the output is formatted in such a way that it can be made part of an *inputrc* file.

dump−variables
Print all of the settable readline variables and their values to the readline output stream. If a numeric argument is supplied, the output is formatted in such a way that it can be made part of an *inputrc* file.

dump−macros
Print all of the readline key sequences bound to macros and the strings they output. If a numeric argument is supplied, the output is formatted in such a way that it can be made part of an *inputrc* file.

display−shell−version (C−x C−v)
Display version information about the current instance of **bash**.

Programmable Completion

When word completion is attempted for an argument to a command for which a completion specification (a *compspec*) has been defined using the **complete** builtin (see **SHELL BUILTIN COMMANDS** below), the programmable completion facilities are invoked.

First, the command name is identified. If the command word is the empty string (completion attempted at the beginning of an empty line), any compspec defined with the **−E** option to **complete** is used. If a compspec has been defined for that command, the compspec is used to generate the list of possible completions for the word. If the command word is a full pathname, a compspec for the full pathname is searched for first. If no compspec is found for the full pathname, an attempt is made to find a compspec for the portion following the final slash. If those searches do not result in a compspec, any compspec defined with the **−D** option to **complete** is used as the default.

Once a compspec has been found, it is used to generate the list of matching words. If a compspec is not

found, the default **bash** completion as described above under **Completing** is performed.

First, the actions specified by the compspec are used. Only matches which are prefixed by the word being completed are returned. When the **−f** or **−d** option is used for filename or directory name completion, the shell variable **FIGNORE** is used to filter the matches.

Any completions specified by a pathname expansion pattern to the **−G** option are generated next. The words generated by the pattern need not match the word being completed. The **GLOBIGNORE** shell variable is not used to filter the matches, but the **FIGNORE** variable is used.

Next, the string specified as the argument to the **−W** option is considered. The string is first split using the characters in the **IFS** special variable as delimiters. Shell quoting is honored. Each word is then expanded using brace expansion, tilde expansion, parameter and variable expansion, command substitution, and arithmetic expansion, as described above under **EXPANSION**. The results are split using the rules described above under **Word Splitting**. The results of the expansion are prefix-matched against the word being completed, and the matching words become the possible completions.

After these matches have been generated, any shell function or command specified with the **−F** and **−C** options is invoked. When the command or function is invoked, the **COMP_LINE**, **COMP_POINT**, **COMP_KEY**, and **COMP_TYPE** variables are assigned values as described above under **Shell Variables**. If a shell function is being invoked, the **COMP_WORDS** and **COMP_CWORD** variables are also set. When the function or command is invoked, the first argument (**$1**) is the name of the command whose arguments are being completed, the second argument (**$2**) is the word being completed, and the third argument (**$3**) is the word preceding the word being completed on the current command line. No filtering of the generated completions against the word being completed is performed; the function or command has complete freedom in generating the matches.

Any function specified with **−F** is invoked first. The function may use any of the shell facilities, including the **compgen** builtin described below, to generate the matches. It must put the possible completions in the **COMPREPLY** array variable, one per array element.

Next, any command specified with the **−C** option is invoked in an environment equivalent to command substitution. It should print a list of completions, one per line, to the standard output. Backslash may be used to escape a newline, if necessary.

After all of the possible completions are generated, any filter specified with the **−X** option is applied to the list. The filter is a pattern as used for pathname expansion; a **&** in the pattern is replaced with the text of the word being completed. A literal **&** may be escaped with a backslash; the backslash is removed before attempting a match. Any completion that matches the pattern will be removed from the list. A leading **!** negates the pattern; in this case any completion not matching the pattern will be removed. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters.

Finally, any prefix and suffix specified with the **−P** and **−S** options are added to each member of the completion list, and the result is returned to the readline completion code as the list of possible completions.

If the previously-applied actions do not generate any matches, and the **−o dirnames** option was supplied to **complete** when the compspec was defined, directory name completion is attempted.

If the **−o plusdirs** option was supplied to **complete** when the compspec was defined, directory name completion is attempted and any matches are added to the results of the other actions.

By default, if a compspec is found, whatever it generates is returned to the completion code as the full set of possible completions. The default **bash** completions are not attempted, and the readline default of filename completion is disabled. If the **−o bashdefault** option was supplied to **complete** when the compspec was defined, the **bash** default completions are attempted if the compspec generates no matches. If the **−o default** option was supplied to **complete** when the compspec was defined, readline's default completion will be performed if the compspec (and, if attempted, the default **bash** completions) generate no matches.

When a compspec indicates that directory name completion is desired, the programmable completion functions force readline to append a slash to completed names which are symbolic links to directories, subject to the value of the **mark−directories** readline variable, regardless of the setting of the **mark-symlinked−directories** readline variable.

There is some support for dynamically modifying completions. This is most useful when used in combination with a default completion specified with **complete -D**. It's possible for shell functions executed as completion handlers to indicate that completion should be retried by returning an exit status of 124. If a shell function returns 124, and changes the compspec associated with the command on which completion is being attempted (supplied as the first argument when the function is executed), programmable completion restarts from the beginning, with an attempt to find a new compspec for that command. This allows a set of completions to be built dynamically as completion is attempted, rather than being loaded all at once.

For instance, assuming that there is a library of compspecs, each kept in a file corresponding to the name of the command, the following default completion function would load completions dynamically:

```
_completion_loader()
{
      . "/etc/bash_completion.d/$1.sh" >/dev/null 2>&1 && return 124
}
complete -D -F _completion_loader -o bashdefault -o default
```

HISTORY

When the **−o history** option to the **set** builtin is enabled, the shell provides access to the *command history*, the list of commands previously typed. The value of the **HISTSIZE** variable is used as the number of commands to save in a history list. The text of the last **HISTSIZE** commands (default 500) is saved. The shell stores each command in the history list prior to parameter and variable expansion (see **EXPANSION** above) but after history expansion is performed, subject to the values of the shell variables **HISTIGNORE** and **HISTCONTROL**.

On startup, the history is initialized from the file named by the variable **HISTFILE** (default *~/.bash_history*). The file named by the value of **HISTFILE** is truncated, if necessary, to contain no more than the number of lines specified by the value of **HISTFILESIZE**. If **HISTFILESIZE** is unset, or set to null, a non-numeric value, or a numeric value less than zero, the history file is not truncated. When the history file is read, lines beginning with the history comment character followed immediately by a digit are interpreted as timestamps for the preceding history line. These timestamps are optionally displayed depending on the value of the **HISTTIMEFORMAT** variable. When a shell with history enabled exits, the last **$HISTSIZE** lines are copied from the history list to **$HISTFILE**. If the **histappend** shell option is enabled (see the description of **shopt** under **SHELL BUILTIN COMMANDS** below), the lines are appended to the history file, otherwise the history file is overwritten. If **HISTFILE** is unset, or if the history file is unwritable, the history is not saved. If the **HISTTIMEFORMAT** variable is set, time stamps are written to the history file, marked with the history comment character, so they may be preserved across shell sessions. This uses the history comment character to distinguish timestamps from other history lines. After saving the history, the history file is truncated to contain no more than **HISTFILESIZE** lines. If **HISTFILESIZE** is unset, or set to null, a non-numeric value, or a numeric value less than zero, the history file is not truncated.

The builtin command **fc** (see **SHELL BUILTIN COMMANDS** below) may be used to list or edit and re-execute a portion of the history list. The **history** builtin may be used to display or modify the history list and manipulate the history file. When using command-line editing, search commands are available in each editing mode that provide access to the history list.

The shell allows control over which commands are saved on the history list. The **HISTCONTROL** and **HISTIGNORE** variables may be set to cause the shell to save only a subset of the commands entered. The **cmdhist** shell option, if enabled, causes the shell to attempt to save each line of a multi-line command in the same history entry, adding semicolons where necessary to preserve syntactic correctness. The **lithist** shell option causes the shell to save the command with embedded newlines instead of semicolons. See the description of the **shopt** builtin below under **SHELL BUILTIN COMMANDS** for information on setting and unsetting shell options.

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** builtin command (see **SHELL BUILTIN COMMANDS**

below). Non-interactive shells do not perform history expansion by default.

History expansions introduce words from the history list into the input stream, making it easy to repeat commands, insert the arguments to a previous command into the current input line, or fix errors in previous commands quickly.

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 history list to use during substitution. The second is to select portions of that line for inclusion into the current one. The line selected from the history is the *event*, and the portions of that line that are acted upon are *words*. Various *modifiers* are available to manipulate the selected words. The line is broken into words in the same fashion as when reading input, so that several *metacharacter*-separated words surrounded by quotes are considered one word. History expansions are introduced by the appearance of the history expansion character, which is **!** by default. Only backslash (\) and single quotes can quote the history expansion character, but the history expansion character is also treated as quoted if it immediately precedes the closing double quote in a double-quoted string.

Several characters inhibit history expansion if found immediately following the history expansion character, even if it is unquoted: space, tab, newline, carriage return, and **=**. If the **extglob** shell option is enabled, **(** will also inhibit expansion.

Several shell options settable with the **shopt** builtin may be used to tailor the behavior of history expansion. If the **histverify** shell option is enabled (see the description of the **shopt** builtin below), and **readline** is being used, history substitutions are not immediately passed to the shell parser. Instead, the expanded line is reloaded into the **readline** editing buffer for further modification. If **readline** is being used, and the **histreedit** shell option is enabled, a failed history substitution will be reloaded into the **readline** editing buffer for correction. The **−p** option to the **history** builtin command may be used to see what a history expansion will do before using it. The **−s** option to the **history** builtin may be used to add commands to the end of the history list without actually executing them, so that they are available for subsequent recall.

The shell allows control of the various characters used by the history expansion mechanism (see the description of **histchars** above under **Shell Variables**). The shell uses the history comment character to mark history timestamps when writing the history file.

Event Designators

An event designator is a reference to a command line entry in the history list. Unless the reference is absolute, events are relative to the current position in the history list.

! Start a history substitution, except when followed by a **blank**, newline, carriage return, = or ((when the **extglob** shell option is enabled using the **shopt** builtin).

!*n* Refer to command line *n*.

!−*n* Refer to the current command minus *n*.

!! Refer to the previous command. This is a synonym for '!−1'.

!*string* Refer to the most recent command preceding the current position in the history list starting with *string*.

!?*string***[?]**
Refer to the most recent command preceding the current position in the history list containing *string*. The trailing **?** may be omitted if *string* is followed immediately by a newline.

^*string1***^***string2***^**
Quick substitution. Repeat the previous command, replacing *string1* with *string2*. Equivalent to "!!:s/*string1*/*string2*/" (see **Modifiers** below).

!# The entire command line typed so far.

Word Designators

Word designators are used to select desired words from the event. A **:** separates the event specification from the word designator. It may 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 0 (zero). Words are inserted into the current line separated by single spaces.

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 word. This is usually the last argument, but will expand to the zeroth word if there is only one word in the line.

% 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.

If a word designator is supplied without an event specification, the previous command is used as the event.

Modifiers

After the optional word designator, there may appear a sequence of one or more of the following modifiers, each preceded by a ':'.

h Remove a trailing filename component, leaving only the head.

t Remove all leading filename components, leaving the tail.

r Remove a trailing suffix of the form *.xxx*, leaving the basename.

e Remove all but the trailing suffix.

p Print the new command but do not execute it.

q Quote the substituted words, escaping further substitutions.

x Quote the substituted words as with **q**, but break into words at **blanks** and newlines.

s/*old*/*new*/
Substitute *new* for the first occurrence of *old* in the event line. Any delimiter can be used in place of /. The final delimiter is optional if it is the last character of the event line. The delimiter may be quoted in *old* and *new* with a single backslash. If & appears in *new*, it is replaced by *old*. A single backslash will quote the &. If *old* is null, it is set to the last *old* substituted, or, if no previous history substitutions took place, the last *string* in a **!?***string***[?]** search.

& Repeat the previous substitution.

g Cause changes to be applied over the entire event line. This is used in conjunction with '**:s**' (e.g., '**:gs**/*old*/*new*/') or '**:&**'. If used with '**:s**', any delimiter can be used in place of /, and the final delimiter is optional if it is the last character of the event line. An **a** may be used as a synonym for **g**.

G Apply the following '**s**' modifier once to each word in the event line.

SHELL BUILTIN COMMANDS

Unless otherwise noted, each builtin command documented in this section as accepting options preceded by − accepts −− to signify the end of the options. The **:**, **true**, **false**, and **test** builtins do not accept options and do not treat −− specially. The **exit**, **logout**, **return**, **break**, **continue**, **let**, and **shift** builtins accept and process arguments beginning with − without requiring −−. Other builtins that accept arguments but are not specified as accepting options interpret arguments beginning with − as invalid options and require −− to prevent this interpretation.

: [*arguments*]
No effect; the command does nothing beyond expanding *arguments* and performing any specified redirections. The return status is zero.

. *filename* [*arguments*]
source *filename* [*arguments*]
Read and execute commands from *filename* in the current shell environment and return the exit status of the last command executed from *filename*. If *filename* does not contain a slash, filenames in **PATH** are used to find the directory containing *filename*. The file searched for in **PATH** need not be executable. When **bash** is not in *posix mode*, the current directory is searched if no file is found in **PATH**. If the **sourcepath** option to the **shopt** builtin command is turned off, the

PATH is not searched. If any *arguments* are supplied, they become the positional parameters when *filename* is executed. Otherwise the positional parameters are unchanged. If the **−T** option is enabled, **source** inherits any trap on **DEBUG**; if it is not, any **DEBUG** trap string is saved and restored around the call to **source**, and **source** unsets the **DEBUG** trap while it executes. If **−T** is not set, and the sourced file changes the **DEBUG** trap, the new value is retained when **source** completes. The return status is the status of the last command exited within the script (0 if no commands are executed), and false if *filename* is not found or cannot be read.

alias [**−p**] [*name*[=*value*] ...]

Alias with no arguments or with the **−p** option prints the list of aliases in the form **alias** *name=value* on standard output. When arguments are supplied, an alias is defined for each *name* whose *value* is given. A trailing space in *value* causes the next word to be checked for alias substitution when the alias is expanded. For each *name* in the argument list for which no *value* is supplied, the name and value of the alias is printed. **Alias** returns true unless a *name* is given for which no alias has been defined.

bg [*jobspec* ...]

Resume each suspended job *jobspec* in the background, as if it had been started with **&**. If *jobspec* is not present, the shell's notion of the *current job* is used. **bg** *jobspec* returns 0 unless run when job control is disabled or, when run with job control enabled, any specified *jobspec* was not found or was started without job control.

bind [**−m** *keymap*] [**−lpsvPSVX**]
bind [**−m** *keymap*] [**−q** *function*] [**−u** *function*] [**−r** *keyseq*]
bind [**−m** *keymap*] **−f** *filename*
bind [**−m** *keymap*] **−x** *keyseq*:*shell−command*
bind [**−m** *keymap*] *keyseq*:*function−name*
bind [**−m** *keymap*] *keyseq*:*readline−command*

Display current **readline** key and function bindings, bind a key sequence to a **readline** function or macro, or set a **readline** variable. Each non-option argument is a command as it would appear in *.inputrc*, but each binding or command must be passed as a separate argument; e.g., '"\C−x\C−r": re−read−init−file'. Options, if supplied, have the following meanings:

−m *keymap*
: Use *keymap* as the keymap to be affected by the subsequent bindings. Acceptable *keymap* names are *emacs*, *emacs−standard*, *emacs−meta*, *emacs−ctlx*, *vi*, *vi−move*, *vi−command*, and *vi−insert*. *vi* is equivalent to *vi−command* (*vi−move* is also a synonym); *emacs* is equivalent to *emacs−standard*.

−l
: List the names of all **readline** functions.

−p
: Display **readline** function names and bindings in such a way that they can be re-read.

−P
: List current **readline** function names and bindings.

−s
: Display **readline** key sequences bound to macros and the strings they output in such a way that they can be re-read.

−S
: Display **readline** key sequences bound to macros and the strings they output.

−v
: Display **readline** variable names and values in such a way that they can be re-read.

−V
: List current **readline** variable names and values.

−f *filename*
: Read key bindings from *filename*.

−q *function*
: Query about which keys invoke the named *function*.

−u *function*
: Unbind all keys bound to the named *function*.

−r *keyseq*
: Remove any current binding for *keyseq*.

−x *keyseq*:*shell−command*
: Cause *shell−command* to be executed whenever *keyseq* is entered. When *shell−command* is executed, the shell sets the **READLINE_LINE** variable to the contents of the

readline line buffer and the **READLINE_POINT** variable to the current location of the insertion point. If the executed command changes the value of **READLINE_LINE** or **READLINE_POINT**, those new values will be reflected in the editing state.

−X List all key sequences bound to shell commands and the associated commands in a format that can be reused as input.

The return value is 0 unless an unrecognized option is given or an error occurred.

break [*n*]

Exit from within a **for**, **while**, **until**, or **select** loop. If *n* is specified, break *n* levels. *n* must be ≥ 1. If *n* is greater than the number of enclosing loops, all enclosing loops are exited. The return value is 0 unless *n* is not greater than or equal to 1.

builtin *shell−builtin* [*arguments*]

Execute the specified shell builtin, passing it *arguments*, and return its exit status. This is useful when defining a function whose name is the same as a shell builtin, retaining the functionality of the builtin within the function. The **cd** builtin is commonly redefined this way. The return status is false if *shell−builtin* is not a shell builtin command.

caller [*expr*]

Returns the context of any active subroutine call (a shell function or a script executed with the **.** or **source** builtins). Without *expr*, **caller** displays the line number and source filename of the current subroutine call. If a non-negative integer is supplied as *expr*, **caller** displays the line number, subroutine name, and source file corresponding to that position in the current execution call stack. This extra information may be used, for example, to print a stack trace. The current frame is frame 0. The return value is 0 unless the shell is not executing a subroutine call or *expr* does not correspond to a valid position in the call stack.

cd [**−L**|[**−P** [**−e**]] [−@]] [*dir*]

Change the current directory to *dir*. if *dir* is not supplied, the value of the **HOME** shell variable is the default. Any additional arguments following *dir* are ignored. The variable **CDPATH** defines the search path for the directory containing *dir*: each directory name in **CDPATH** is searched for *dir*. Alternative directory names in **CDPATH** are separated by a colon (:). A null directory name in **CDPATH** is the same as the current directory, i.e., ‘‘**.**’’. If *dir* begins with a slash (/), then **CDPATH** is not used. The **−P** option causes **cd** to use the physical directory structure by resolving symbolic links while traversing *dir* and before processing instances of .. in *dir* (see also the **−P** option to the **set** builtin command); the **−L** option forces symbolic links to be followed by resolving the link after processing instances of .. in *dir*. If .. appears in *dir*, it is processed by removing the immediately previous pathname component from *dir*, back to a slash or the beginning of *dir*. If the **−e** option is supplied with **−P**, and the current working directory cannot be successfully determined after a successful directory change, **cd** will return an unsuccessful status. On systems that support it, the **−@** option presents the extended attributes associated with a file as a directory. An argument of **−** is converted to **$OLDPWD** before the directory change is attempted. If a non-empty directory name from **CDPATH** is used, or if **−** is the first argument, and the directory change is successful, the absolute pathname of the new working directory is written to the standard output. The return value is true if the directory was successfully changed; false otherwise.

command [**−pVv**] *command* [*arg* ...]

Run *command* with *args* suppressing the normal shell function lookup. Only builtin commands or commands found in the **PATH** are executed. If the **−p** option is given, the search for *command* is performed using a default value for **PATH** that is guaranteed to find all of the standard utilities. If either the **−V** or **−v** option is supplied, a description of *command* is printed. The **−v** option causes a single word indicating the command or filename used to invoke *command* to be displayed; the **−V** option produces a more verbose description. If the **−V** or **−v** option is supplied, the exit status is 0 if *command* was found, and 1 if not. If neither option is supplied and an error occurred or *command* cannot be found, the exit status is 127. Otherwise, the exit status of the **command** builtin is the exit status of *command*.

compgen [*option*] [*word*]
Generate possible completion matches for *word* according to the *option*s, which may be any option accepted by the **complete** builtin with the exception of **−p** and **−r**, and write the matches to the standard output. When using the **−F** or **−C** options, the various shell variables set by the programmable completion facilities, while available, will not have useful values.

The matches will be generated in the same way as if the programmable completion code had generated them directly from a completion specification with the same flags. If *word* is specified, only those completions matching *word* will be displayed.

The return value is true unless an invalid option is supplied, or no matches were generated.

complete [**−abcdefgjksuv**] [**−o** *comp-option*] [**−DE**] [**−A** *action*] [**−G** *globpat*] [**−W** *wordlist*] [**−F** *function*] [**−C** *command*]
[**−X** *filterpat*] [**−P** *prefix*] [**−S** *suffix*] *name* [*name* ...]
complete −pr [**−DE**] [*name* ...]
Specify how arguments to each *name* should be completed. If the **−p** option is supplied, or if no options are supplied, existing completion specifications are printed in a way that allows them to be reused as input. The **−r** option removes a completion specification for each *name*, or, if no *name*s are supplied, all completion specifications. The **−D** option indicates that the remaining options and actions should apply to the ''default'' command completion; that is, completion attempted on a command for which no completion has previously been defined. The **−E** option indicates that the remaining options and actions should apply to ''empty'' command completion; that is, completion attempted on a blank line.

The process of applying these completion specifications when word completion is attempted is described above under **Programmable Completion**.

Other options, if specified, have the following meanings. The arguments to the **−G**, **−W**, and **−X** options (and, if necessary, the **−P** and **−S** options) should be quoted to protect them from expansion before the **complete** builtin is invoked.
−o *comp-option*
The *comp-option* controls several aspects of the compspec's behavior beyond the simple generation of completions. *comp-option* may be one of:
bashdefault
Perform the rest of the default **bash** completions if the compspec generates no matches.
default Use readline's default filename completion if the compspec generates no matches.
dirnames
Perform directory name completion if the compspec generates no matches.
filenames
Tell readline that the compspec generates filenames, so it can perform any filename−specific processing (like adding a slash to directory names, quoting special characters, or suppressing trailing spaces). Intended to be used with shell functions.
noquote Tell readline not to quote the completed words if they are filenames (quoting filenames is the default).
nosort Tell readline not to sort the list of possible completions alphabetically.
nospace Tell readline not to append a space (the default) to words completed at the end of the line.
plusdirs After any matches defined by the compspec are generated, directory name completion is attempted and any matches are added to the results of the other actions.

−A *action*

The *action* may be one of the following to generate a list of possible completions:

alias Alias names. May also be specified as **−a**.

arrayvar
Array variable names.

binding **Readline** key binding names.

builtin Names of shell builtin commands. May also be specified as **−b**.

command
Command names. May also be specified as **−c**.

directory
Directory names. May also be specified as **−d**.

disabled
Names of disabled shell builtins.

enabled Names of enabled shell builtins.

export Names of exported shell variables. May also be specified as **−e**.

file File names. May also be specified as **−f**.

function
Names of shell functions.

group Group names. May also be specified as **−g**.

helptopic
Help topics as accepted by the **help** builtin.

hostname
Hostnames, as taken from the file specified by the **HOSTFILE** shell variable.

job Job names, if job control is active. May also be specified as **−j**.

keyword
Shell reserved words. May also be specified as **−k**.

running Names of running jobs, if job control is active.

service Service names. May also be specified as **−s**.

setopt Valid arguments for the **−o** option to the **set** builtin.

shopt Shell option names as accepted by the **shopt** builtin.

signal Signal names.

stopped Names of stopped jobs, if job control is active.

user User names. May also be specified as **−u**.

variable Names of all shell variables. May also be specified as **−v**.

−C *command*

command is executed in a subshell environment, and its output is used as the possible completions.

−F *function*

The shell function *function* is executed in the current shell environment. When the function is executed, the first argument (**$1**) is the name of the command whose arguments are being completed, the second argument (**$2**) is the word being completed, and the third argument (**$3**) is the word preceding the word being completed on the current command line. When it finishes, the possible completions are retrieved from the value of the **COMPREPLY** array variable.

−G *globpat*

The pathname expansion pattern *globpat* is expanded to generate the possible completions.

−P *prefix*

prefix is added at the beginning of each possible completion after all other options have been applied.

−S *suffix* *suffix* is appended to each possible completion after all other options have been applied.

−W *wordlist*

The *wordlist* is split using the characters in the **IFS** special variable as delimiters, and each resultant word is expanded. The possible completions are the members of the

resultant list which match the word being completed.

−X *filterpat*

filterpat is a pattern as used for pathname expansion. It is applied to the list of possible completions generated by the preceding options and arguments, and each completion matching *filterpat* is removed from the list. A leading **!** in *filterpat* negates the pattern; in this case, any completion not matching *filterpat* is removed.

The return value is true unless an invalid option is supplied, an option other than **−p** or **−r** is supplied without a *name* argument, an attempt is made to remove a completion specification for a *name* for which no specification exists, or an error occurs adding a completion specification.

compopt [**−o** *option*] [**−DE**] [**+o** *option*] [*name*]

Modify completion options for each *name* according to the *option*s, or for the currently-executing completion if no *name*s are supplied. If no *option*s are given, display the completion options for each *name* or the current completion. The possible values of *option* are those valid for the **complete** builtin described above. The **−D** option indicates that the remaining options should apply to the ''default'' command completion; that is, completion attempted on a command for which no completion has previously been defined. The **−E** option indicates that the remaining options should apply to ''empty'' command completion; that is, completion attempted on a blank line.

The return value is true unless an invalid option is supplied, an attempt is made to modify the options for a *name* for which no completion specification exists, or an output error occurs.

continue [*n*]

Resume the next iteration of the enclosing **for**, **while**, **until**, or **select** loop. If *n* is specified, resume at the *n*th enclosing loop. *n* must be ≥ 1. If *n* is greater than the number of enclosing loops, the last enclosing loop (the ''top-level'' loop) is resumed. The return value is 0 unless *n* is not greater than or equal to 1.

declare [**−aAfFgilnrtux**] [**−p**] [*name*[*=value*] ...]
typeset [**−aAfFgilnrtux**] [**−p**] [*name*[*=value*] ...]

Declare variables and/or give them attributes. If no *name*s are given then display the values of variables. The **−p** option will display the attributes and values of each *name*. When **−p** is used with *name* arguments, additional options, other than **−f** and **−F**, are ignored. When **−p** is supplied without *name* arguments, it will display the attributes and values of all variables having the attributes specified by the additional options. If no other options are supplied with **−p**, **declare** will display the attributes and values of all shell variables. The **−f** option will restrict the display to shell functions. The **−F** option inhibits the display of function definitions; only the function name and attributes are printed. If the **extdebug** shell option is enabled using **shopt**, the source file name and line number where each *name* is defined are displayed as well. The **−F** option implies **−f**. The **−g** option forces variables to be created or modified at the global scope, even when **declare** is executed in a shell function. It is ignored in all other cases. The following options can be used to restrict output to variables with the specified attribute or to give variables attributes:

−a Each *name* is an indexed array variable (see **Arrays** above).

−A Each *name* is an associative array variable (see **Arrays** above).

−f Use function names only.

−i The variable is treated as an integer; arithmetic evaluation (see **ARITHMETIC EVALUATION** above) is performed when the variable is assigned a value.

−l When the variable is assigned a value, all upper-case characters are converted to lower-case. The upper-case attribute is disabled.

−n Give each *name* the *nameref* attribute, making it a name reference to another variable. That other variable is defined by the value of *name*. All references, assignments, and attribute modifications to *name*, except those using or changing the **−n** attribute itself, are performed on the variable referenced by *name*'s value. The nameref attribute cannot be applied to array variables.

−r Make *name*s readonly. These names cannot then be assigned values by subsequent assignment statements or unset.

−t Give each *name* the *trace* attribute. Traced functions inherit the **DEBUG** and **RETURN** traps from the calling shell. The trace attribute has no special meaning for variables.

−u When the variable is assigned a value, all lower-case characters are converted to upper-case. The lower-case attribute is disabled.

−x Mark *name*s for export to subsequent commands via the environment.

Using '+' instead of '−' turns off the attribute instead, with the exceptions that **+a** may not be used to destroy an array variable and **+r** will not remove the readonly attribute. When used in a function, **declare** and **typeset** make each *name* local, as with the **local** command, unless the **−g** option is supplied. If a variable name is followed by =*value*, the value of the variable is set to *value*. When using **−a** or **−A** and the compound assignment syntax to create array variables, additional attributes do not take effect until subsequent assignments. The return value is 0 unless an invalid option is encountered, an attempt is made to define a function using `−f foo=bar`, an attempt is made to assign a value to a readonly variable, an attempt is made to assign a value to an array variable without using the compound assignment syntax (see **Arrays** above), one of the *names* is not a valid shell variable name, an attempt is made to turn off readonly status for a readonly variable, an attempt is made to turn off array status for an array variable, or an attempt is made to display a non-existent function with **−f**.

dirs [−clpv] [+*n*] [−*n*]

Without options, displays the list of currently remembered directories. The default display is on a single line with directory names separated by spaces. Directories are added to the list with the **pushd** command; the **popd** command removes entries from the list. The current directory is always the first directory in the stack.

−c Clears the directory stack by deleting all of the entries.

−l Produces a listing using full pathnames; the default listing format uses a tilde to denote the home directory.

−p Print the directory stack with one entry per line.

−v Print the directory stack with one entry per line, prefixing each entry with its index in the stack.

+*n* Displays the *n*th entry counting from the left of the list shown by **dirs** when invoked without options, starting with zero.

−*n* Displays the *n*th entry counting from the right of the list shown by **dirs** when invoked without options, starting with zero.

The return value is 0 unless an invalid option is supplied or *n* indexes beyond the end of the directory stack.

disown [**−ar**] [**−h**] [*jobspec ... | pid ...*]

Without options, remove each *jobspec* from the table of active jobs. If *jobspec* is not present, and neither the **−a** nor the **−r** option is supplied, the *current job* is used. If the **−h** option is given, each *jobspec* is not removed from the table, but is marked so that **SIGHUP** is not sent to the job if the shell receives a **SIGHUP**. If no *jobspec* is supplied, the **−a** option means to remove or mark all jobs; the **−r** option without a *jobspec* argument restricts operation to running jobs. The return value is 0 unless a *jobspec* does not specify a valid job.

echo [**−neE**] [*arg ...*]

Output the *arg*s, separated by spaces, followed by a newline. The return status is 0 unless a write error occurs. If **−n** is specified, the trailing newline is suppressed. If the **−e** option is given, interpretation of the following backslash-escaped characters is enabled. The **−E** option disables the interpretation of these escape characters, even on systems where they are interpreted by default. The **xpg_echo** shell option may be used to dynamically determine whether or not **echo** expands these escape characters by default. **echo** does not interpret **−−** to mean the end of options. **echo** interprets the following escape sequences:

\a alert (bell)
\b backspace
\c suppress further output
\e
\E an escape character
\f form feed
\n new line
\r carriage return
\t horizontal tab
\v vertical tab
**** backslash
\0*nnn* the eight-bit character whose value is the octal value *nnn* (zero to three octal digits)
\x*HH* the eight-bit character whose value is the hexadecimal value *HH* (one or two hex digits)
\u*HHHH*
the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHH* (one to four hex digits)
\U*HHHHHHHH*
the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHHHHHH* (one to eight hex digits)

enable [**−a**] [**−dnps**] [**−f** *filename*] [*name* ...]
Enable and disable builtin shell commands. Disabling a builtin allows a disk command which has the same name as a shell builtin to be executed without specifying a full pathname, even though the shell normally searches for builtins before disk commands. If **−n** is used, each *name* is disabled; otherwise, *names* are enabled. For example, to use the **test** binary found via the **PATH** instead of the shell builtin version, run `enable -n test`. The **−f** option means to load the new builtin command *name* from shared object *filename*, on systems that support dynamic loading. The **−d** option will delete a builtin previously loaded with **−f**. If no *name* arguments are given, or if the **−p** option is supplied, a list of shell builtins is printed. With no other option arguments, the list consists of all enabled shell builtins. If **−n** is supplied, only disabled builtins are printed. If **−a** is supplied, the list printed includes all builtins, with an indication of whether or not each is enabled. If **−s** is supplied, the output is restricted to the POSIX *special* builtins. The return value is 0 unless a *name* is not a shell builtin or there is an error loading a new builtin from a shared object.

eval [*arg* ...]
The *arg*s are read and concatenated together into a single command. This command is then read and executed by the shell, and its exit status is returned as the value of **eval**. If there are no *args*, or only null arguments, **eval** returns 0.

exec [**−cl**] [**−a** *name*] [*command* [*arguments*]]
If *command* is specified, it replaces the shell. No new process is created. The *arguments* become the arguments to *command*. If the **−l** option is supplied, the shell places a dash at the beginning of the zeroth argument passed to *command*. This is what *login*(1) does. The **−c** option causes *command* to be executed with an empty environment. If **−a** is supplied, the shell passes *name* as the zeroth argument to the executed command. If *command* cannot be executed for some reason, a non-interactive shell exits, unless the **execfail** shell option is enabled. In that case, it returns failure. An interactive shell returns failure if the file cannot be executed. If *command* is not specified, any redirections take effect in the current shell, and the return status is 0. If there is a redirection error, the return status is 1.

exit [*n*] Cause the shell to exit with a status of *n*. If *n* is omitted, the exit status is that of the last command executed. A trap on **EXIT** is executed before the shell terminates.

export [**−fn**] [*name*[=*word*]] ...

export −p

The supplied *names* are marked for automatic export to the environment of subsequently executed commands. If the **−f** option is given, the *names* refer to functions. If no *names* are given, or if the **−p** option is supplied, a list of names of all exported variables is printed. The **−n** option causes the export property to be removed from each *name*. If a variable name is followed by =*word*, the value of the variable is set to *word*. **export** returns an exit status of 0 unless an invalid option is encountered, one of the *names* is not a valid shell variable name, or **−f** is supplied with a *name* that is not a function.

fc [**−e** *ename*] [**−lnr**] [*first*] [*last*]
fc −s [*pat=rep*] [*cmd*]

The first form selects a range of commands from *first* to *last* from the history list and displays or edits and re-executes them. *First* and *last* may be specified as a string (to locate the last command beginning with that string) or as a number (an index into the history list, where a negative number is used as an offset from the current command number). If *last* is not specified it is set to the current command for listing (so that `fc -l -10` prints the last 10 commands) and to *first* otherwise. If *first* is not specified it is set to the previous command for editing and −16 for listing.

The **−n** option suppresses the command numbers when listing. The **−r** option reverses the order of the commands. If the **−l** option is given, the commands are listed on standard output. Otherwise, the editor given by *ename* is invoked on a file containing those commands. If *ename* is not given, the value of the **FCEDIT** variable is used, and the value of **EDITOR** if **FCEDIT** is not set. If neither variable is set, *vi* is used. When editing is complete, the edited commands are echoed and executed.

In the second form, *command* is re-executed after each instance of *pat* is replaced by *rep*. *Command* is intepreted the same as *first* above. A useful alias to use with this is `r='fc -s'`, so that typing `r cc` runs the last command beginning with `cc` and typing `r` re-executes the last command.

If the first form is used, the return value is 0 unless an invalid option is encountered or *first* or *last* specify history lines out of range. If the **−e** option is supplied, the return value is the value of the last command executed or failure if an error occurs with the temporary file of commands. If the second form is used, the return status is that of the command re-executed, unless *cmd* does not specify a valid history line, in which case **fc** returns failure.

fg [*jobspec*]

Resume *jobspec* in the foreground, and make it the current job. If *jobspec* is not present, the shell's notion of the *current job* is used. The return value is that of the command placed into the foreground, or failure if run when job control is disabled or, when run with job control enabled, if *jobspec* does not specify a valid job or *jobspec* specifies a job that was started without job control.

getopts *optstring name* [*args*]

getopts is used by shell procedures to parse positional parameters. *optstring* contains the option characters to be recognized; if a character is followed by a colon, the option is expected to have an argument, which should be separated from it by white space. The colon and question mark characters may not be used as option characters. Each time it is invoked, **getopts** places the next option in the shell variable *name*, initializing *name* if it does not exist, and the index of the next argument to be processed into the variable **OPTIND**. **OPTIND** is initialized to 1 each time the shell or a shell script is invoked. When an option requires an argument, **getopts** places that argument into the variable **OPTARG**. The shell does not reset **OPTIND** automatically; it must be manually reset between multiple calls to **getopts** within the same shell invocation if a new set of parameters is to be used.

When the end of options is encountered, **getopts** exits with a return value greater than zero. **OPTIND** is set to the index of the first non-option argument, and *name* is set to ?.

getopts normally parses the positional parameters, but if more arguments are given in *args*, **getopts** parses those instead.

getopts can report errors in two ways. If the first character of *optstring* is a colon, *silent* error reporting is used. In normal operation, diagnostic messages are printed when invalid options or missing option arguments are encountered. If the variable **OPTERR** is set to 0, no error messages will be displayed, even if the first character of *optstring* is not a colon.

If an invalid option is seen, **getopts** places ? into *name* and, if not silent, prints an error message and unsets **OPTARG**. If **getopts** is silent, the option character found is placed in **OPTARG** and no diagnostic message is printed.

If a required argument is not found, and **getopts** is not silent, a question mark (**?**) is placed in *name*, **OPTARG** is unset, and a diagnostic message is printed. If **getopts** is silent, then a colon (**:**) is placed in *name* and **OPTARG** is set to the option character found.

getopts returns true if an option, specified or unspecified, is found. It returns false if the end of options is encountered or an error occurs.

hash [**−lr**] [**−p** *filename*] [**−dt**] [*name*]
Each time **hash** is invoked, the full pathname of the command *name* is determined by searching the directories in **$PATH** and remembered. Any previously-remembered pathname is discarded. If the **−p** option is supplied, no path search is performed, and *filename* is used as the full filename of the command. The **−r** option causes the shell to forget all remembered locations. The **−d** option causes the shell to forget the remembered location of each *name*. If the **−t** option is supplied, the full pathname to which each *name* corresponds is printed. If multiple *name* arguments are supplied with **−t**, the *name* is printed before the hashed full pathname. The **−l** option causes output to be displayed in a format that may be reused as input. If no arguments are given, or if only **−l** is supplied, information about remembered commands is printed. The return status is true unless a *name* is not found or an invalid option is supplied.

help [**−dms**] [*pattern*]
Display helpful information about builtin commands. If *pattern* is specified, **help** gives detailed help on all commands matching *pattern*; otherwise help for all the builtins and shell control structures is printed.
−d Display a short description of each *pattern*
−m Display the description of each *pattern* in a manpage-like format
−s Display only a short usage synopsis for each *pattern*

The return status is 0 unless no command matches *pattern*.

history [*n***]**
history −c
history −d *offset*
history −anrw [*filename*]
history −p *arg* [*arg ...*]
history −s *arg* [*arg ...*]
With no options, display the command history list with line numbers. Lines listed with a ***** have been modified. An argument of *n* lists only the last *n* lines. If the shell variable **HISTTIMEFORMAT** is set and not null, it is used as a format string for *strftime*(3) to display the time stamp associated with each displayed history entry. No intervening blank is printed between the formatted time stamp and the history line. If *filename* is supplied, it is used as the name of the history file; if not, the value of **HISTFILE** is used. Options, if supplied, have the following meanings:
−c Clear the history list by deleting all the entries.
−d *offset*
Delete the history entry at position *offset*.

−a Append the ''new'' history lines to the history file. These are history lines entered since the beginning of the current **bash** session, but not already appended to the history file.

−n Read the history lines not already read from the history file into the current history list. These are lines appended to the history file since the beginning of the current **bash** session.

−r Read the contents of the history file and append them to the current history list.

−w Write the current history list to the history file, overwriting the history file's contents.

−p Perform history substitution on the following *args* and display the result on the standard output. Does not store the results in the history list. Each *arg* must be quoted to disable normal history expansion.

−s Store the *args* in the history list as a single entry. The last command in the history list is removed before the *args* are added.

If the **HISTTIMEFORMAT** variable is set, the time stamp information associated with each history entry is written to the history file, marked with the history comment character. When the history file is read, lines beginning with the history comment character followed immediately by a digit are interpreted as timestamps for the following history entry. The return value is 0 unless an invalid option is encountered, an error occurs while reading or writing the history file, an invalid *offset* is supplied as an argument to **−d**, or the history expansion supplied as an argument to **−p** fails.

jobs [**−lnprs**] [*jobspec* ...]
jobs −x *command* [*args* ...]

The first form lists the active jobs. The options have the following meanings:

−l List process IDs in addition to the normal information.

−n Display information only about jobs that have changed status since the user was last notified of their status.

−p List only the process ID of the job's process group leader.

−r Display only running jobs.

−s Display only stopped jobs.

If *jobspec* is given, output is restricted to information about that job. The return status is 0 unless an invalid option is encountered or an invalid *jobspec* is supplied.

If the **−x** option is supplied, **jobs** replaces any *jobspec* found in *command* or *args* with the corresponding process group ID, and executes *command* passing it *args*, returning its exit status.

kill [**−s** *sigspec* | **−n** *signum* | **−***sigspec*] [*pid* | *jobspec*] ...
kill −l|**−L** [*sigspec* | *exit_status*]

Send the signal named by *sigspec* or *signum* to the processes named by *pid* or *jobspec*. *sigspec* is either a case-insensitive signal name such as **SIGKILL** (with or without the **SIG** prefix) or a signal number; *signum* is a signal number. If *sigspec* is not present, then **SIGTERM** is assumed. An argument of **−l** lists the signal names. If any arguments are supplied when **−l** is given, the names of the signals corresponding to the arguments are listed, and the return status is 0. The *exit_status* argument to **−l** is a number specifying either a signal number or the exit status of a process terminated by a signal. The **−L** option is equivalent to **−l**. **kill** returns true if at least one signal was successfully sent, or false if an error occurs or an invalid option is encountered.

let *arg* [*arg* ...]

Each *arg* is an arithmetic expression to be evaluated (see **ARITHMETIC EVALUATION** above). If the last *arg* evaluates to 0, **let** returns 1; 0 is returned otherwise.

local [*option*] [*name*[=*value*] ... | −]

For each argument, a local variable named *name* is created, and assigned *value*. The *option* can be any of the options accepted by **declare**. When **local** is used within a function, it causes the variable *name* to have a visible scope restricted to that function and its children. If *name* is −, the set of shell options is made local to the function in which **local** is invoked: shell options changed using the **set** builtin inside the function are restored to their original values when the function returns. With no operands, **local** writes a list of local variables to the standard output. It is an

error to use **local** when not within a function. The return status is 0 unless **local** is used outside a function, an invalid *name* is supplied, or *name* is a readonly variable.

logout Exit a login shell.

mapfile [**−d** *delim*] [**−n** *count*] [**−O** *origin*] [**−s** *count*] [**−t**] [**−u** *fd*] [**−C** *callback*] [**−c** *quantum*] [*array*]
readarray [**−d** *delim*] [**−n** *count*] [**−O** *origin*] [**−s** *count*] [**−t**] [**−u** *fd*] [**−C** *callback*] [**−c** *quantum*] [*array*]
Read lines from the standard input into the indexed array variable *array*, or from file descriptor *fd* if the **−u** option is supplied. The variable **MAPFILE** is the default *array*. Options, if supplied, have the following meanings:

−d The first character of *delim* is used to terminate each input line, rather than newline.
−n Copy at most *count* lines. If *count* is 0, all lines are copied.
−O Begin assigning to *array* at index *origin*. The default index is 0.
−s Discard the first *count* lines read.
−t Remove a trailing *delim* (default newline) from each line read.
−u Read lines from file descriptor *fd* instead of the standard input.
−C Evaluate *callback* each time *quantum* lines are read. The **−c** option specifies *quantum*.
−c Specify the number of lines read between each call to *callback*.

If **−C** is specified without **−c**, the default quantum is 5000. When *callback* is evaluated, it is supplied the index of the next array element to be assigned and the line to be assigned to that element as additional arguments. *callback* is evaluated after the line is read but before the array element is assigned.

If not supplied with an explicit origin, **mapfile** will clear *array* before assigning to it.

mapfile returns successfully unless an invalid option or option argument is supplied, *array* is invalid or unassignable, or if *array* is not an indexed array.

popd [−**n**] [+*n*] [−*n*]
Removes entries from the directory stack. With no arguments, removes the top directory from the stack, and performs a **cd** to the new top directory. Arguments, if supplied, have the following meanings:

−n Suppresses the normal change of directory when removing directories from the stack, so that only the stack is manipulated.
+*n* Removes the *n*th entry counting from the left of the list shown by **dirs**, starting with zero. For example: `popd +0` removes the first directory, `popd +1` the second.
−*n* Removes the *n*th entry counting from the right of the list shown by **dirs**, starting with zero. For example: `popd −0` removes the last directory, `popd −1` the next to last.

If the **popd** command is successful, a **dirs** is performed as well, and the return status is 0. **popd** returns false if an invalid option is encountered, the directory stack is empty, a non-existent directory stack entry is specified, or the directory change fails.

printf [**−v** *var*] *format* [*arguments*]
Write the formatted *arguments* to the standard output under the control of the *format*. The **−v** option causes the output to be assigned to the variable *var* rather than being printed to the standard output.

The *format* is a character string which contains three types of objects: plain characters, which are simply copied to standard output, character escape sequences, which are converted and copied to the standard output, and format specifications, each of which causes printing of the next successive *argument*. In addition to the standard *printf*(1) format specifications, **printf** interprets the following extensions:

%b causes **printf** to expand backslash escape sequences in the corresponding *argument* in the same way as **echo −e**.
%q causes **printf** to output the corresponding *argument* in a format that can be reused as shell input.

%(*datefmt***)T**
causes **printf** to output the date-time string resulting from using *datefmt* as a format string for *strftime*(3). The corresponding *argument* is an integer representing the number of seconds since the epoch. Two special argument values may be used: -1 represents the current time, and -2 represents the time the shell was invoked. If no argument is specified, conversion behaves as if -1 had been given. This is an exception to the usual **printf** behavior.

Arguments to non-string format specifiers are treated as C constants, except that a leading plus or minus sign is allowed, and if the leading character is a single or double quote, the value is the ASCII value of the following character.

The *format* is reused as necessary to consume all of the *arguments*. If the *format* requires more *arguments* than are supplied, the extra format specifications behave as if a zero value or null string, as appropriate, had been supplied. The return value is zero on success, non-zero on failure.

pushd [**−n**] [+*n*] [−*n*]
pushd [**−n**] [*dir*]
Adds a directory to the top of the directory stack, or rotates the stack, making the new top of the stack the current working directory. With no arguments, **pushd** exchanges the top two directories and returns 0, unless the directory stack is empty. Arguments, if supplied, have the following meanings:

−n Suppresses the normal change of directory when rotating or adding directories to the stack, so that only the stack is manipulated.

+*n* Rotates the stack so that the *n*th directory (counting from the left of the list shown by **dirs**, starting with zero) is at the top.

−*n* Rotates the stack so that the *n*th directory (counting from the right of the list shown by **dirs**, starting with zero) is at the top.

dir Adds *dir* to the directory stack at the top, making it the new current working directory as if it had been supplied as the argument to the **cd** builtin.

If the **pushd** command is successful, a **dirs** is performed as well. If the first form is used, **pushd** returns 0 unless the cd to *dir* fails. With the second form, **pushd** returns 0 unless the directory stack is empty, a non-existent directory stack element is specified, or the directory change to the specified new current directory fails.

pwd [**−LP**]
Print the absolute pathname of the current working directory. The pathname printed contains no symbolic links if the **−P** option is supplied or the **−o physical** option to the **set** builtin command is enabled. If the **−L** option is used, the pathname printed may contain symbolic links. The return status is 0 unless an error occurs while reading the name of the current directory or an invalid option is supplied.

read [**−ers**] [**−a** *aname*] [**−d** *delim*] [**−i** *text*] [**−n** *nchars*] [**−N** *nchars*] [**−p** *prompt*] [**−t** *timeout*] [**−u** *fd*] [*name ...*]
One line is read from the standard input, or from the file descriptor *fd* supplied as an argument to the **−u** option, split into words as described above under **Word Splitting**, and the first word is assigned to the first *name*, the second word to the second *name*, and so on. If there are more words than names, the remaining words and their intervening delimiters are assigned to the last *name*. If there are fewer words read from the input stream than names, the remaining names are assigned empty values. The characters in **IFS** are used to split the line into words using the same rules the shell uses for expansion (described above under **Word Splitting**). The backslash character (\) may be used to remove any special meaning for the next character read and for line continuation. Options, if supplied, have the following meanings:

−a *aname*
The words are assigned to sequential indices of the array variable *aname*, starting at 0. *aname* is unset before any new values are assigned. Other *name* arguments are ignored.

−d *delim*
The first character of *delim* is used to terminate the input line, rather than newline.

−e If the standard input is coming from a terminal, **readline** (see **READLINE** above) is used to obtain the line. Readline uses the current (or default, if line editing was not previously active) editing settings.

−i *text* If **readline** is being used to read the line, *text* is placed into the editing buffer before editing begins.

−n *nchars*
read returns after reading *nchars* characters rather than waiting for a complete line of input, but honors a delimiter if fewer than *nchars* characters are read before the delimiter.

−N *nchars*
read returns after reading exactly *nchars* characters rather than waiting for a complete line of input, unless EOF is encountered or **read** times out. Delimiter characters encountered in the input are not treated specially and do not cause **read** to return until *nchars* characters are read. The result is not split on the characters in **IFS**; the intent is that the variable is assigned exactly the characters read (with the exception of backslash; see the **−r** option below).

−p *prompt*
Display *prompt* on standard error, without a trailing newline, before attempting to read any input. The prompt is displayed only if input is coming from a terminal.

−r Backslash does not act as an escape character. The backslash is considered to be part of the line. In particular, a backslash-newline pair may not be used as a line continuation.

−s Silent mode. If input is coming from a terminal, characters are not echoed.

−t *timeout*
Cause **read** to time out and return failure if a complete line of input (or a specified number of characters) is not read within *timeout* seconds. *timeout* may be a decimal number with a fractional portion following the decimal point. This option is only effective if **read** is reading input from a terminal, pipe, or other special file; it has no effect when reading from regular files. If **read** times out, **read** saves any partial input read into the specified variable *name*. If *timeout* is 0, **read** returns immediately, without trying to read any data. The exit status is 0 if input is available on the specified file descriptor, non-zero otherwise. The exit status is greater than 128 if the timeout is exceeded.

−u *fd* Read input from file descriptor *fd*.

If no *names* are supplied, the line read is assigned to the variable **REPLY**. The exit status is zero, unless end-of-file is encountered, **read** times out (in which case the status is greater than 128), a variable assignment error (such as assigning to a readonly variable) occurs, or an invalid file descriptor is supplied as the argument to **−u**.

readonly [**−aAf**] [**−p**] [*name*[=*word*] ...]
The given *names* are marked readonly; the values of these *names* may not be changed by subsequent assignment. If the **−f** option is supplied, the functions corresponding to the *names* are so marked. The **−a** option restricts the variables to indexed arrays; the **−A** option restricts the variables to associative arrays. If both options are supplied, **−A** takes precedence. If no *name* arguments are given, or if the **−p** option is supplied, a list of all readonly names is printed. The other options may be used to restrict the output to a subset of the set of readonly names. The **−p** option causes output to be displayed in a format that may be reused as input. If a variable name is followed by =*word*, the value of the variable is set to *word*. The return status is 0 unless an invalid option is encountered, one of the *names* is not a valid shell variable name, or **−f** is supplied with a *name* that is not a function.

return [*n*]
Causes a function to stop executing and return the value specified by *n* to its caller. If *n* is omitted, the return status is that of the last command executed in the function body. If **return** is executed by a trap handler, the last command used to determine the status is the last command executed before the trap handler. if **return** is executed during a **DEBUG** trap, the last command used to

determine the status is the last command executed by the trap handler before **return** was invoked. If **return** is used outside a function, but during execution of a script by the **.** (**source**) command, it causes the shell to stop executing that script and return either *n* or the exit status of the last command executed within the script as the exit status of the script. If *n* is supplied, the return value is its least significant 8 bits. The return status is non-zero if **return** is supplied a non-numeric argument, or is used outside a function and not during execution of a script by **.** or **source**. Any command associated with the **RETURN** trap is executed before execution resumes after the function or script.

set [**−−abefhkmnptuvxBCEHPT**] [**−o** *option−name*] [*arg* ...]
set [**+abefhkmnptuvxBCEHPT**] [**+o** *option−name*] [*arg* ...]

Without options, the name and value of each shell variable are displayed in a format that can be reused as input for setting or resetting the currently-set variables. Read-only variables cannot be reset. In *posix* mode, only shell variables are listed. The output is sorted according to the current locale. When options are specified, they set or unset shell attributes. Any arguments remaining after option processing are treated as values for the positional parameters and are assigned, in order, to **$1**, **$2**, **...** **$***n*. Options, if specified, have the following meanings:

−a Each variable or function that is created or modified is given the export attribute and marked for export to the environment of subsequent commands.

−b Report the status of terminated background jobs immediately, rather than before the next primary prompt. This is effective only when job control is enabled.

−e Exit immediately if a *pipeline* (which may consist of a single *simple command*), a *list*, or a *compound command* (see **SHELL GRAMMAR** above), exits with a non-zero status. The shell does not exit if the command that fails is part of the command list immediately following a **while** or **until** keyword, part of the test following the **if** or **elif** reserved words, part of any command executed in a **&&** or || list except the command following the final **&&** or ||, any command in a pipeline but the last, or if the command's return value is being inverted with **!**. If a compound command other than a subshell returns a non-zero status because a command failed while **−e** was being ignored, the shell does not exit. A trap on **ERR**, if set, is executed before the shell exits. This option applies to the shell environment and each subshell environment separately (see **COMMAND EXECUTION ENVIRONMENT** above), and may cause subshells to exit before executing all the commands in the subshell.

If a compound command or shell function executes in a context where **−e** is being ignored, none of the commands executed within the compound command or function body will be affected by the **−e** setting, even if **−e** is set and a command returns a failure status. If a compound command or shell function sets **−e** while executing in a context where **−e** is ignored, that setting will not have any effect until the compound command or the command containing the function call completes.

−f Disable pathname expansion.

−h Remember the location of commands as they are looked up for execution. This is enabled by default.

−k All arguments in the form of assignment statements are placed in the environment for a command, not just those that precede the command name.

−m Monitor mode. Job control is enabled. This option is on by default for interactive shells on systems that support it (see **JOB CONTROL** above). All processes run in a separate process group. When a background job completes, the shell prints a line containing its exit status.

−n Read commands but do not execute them. This may be used to check a shell script for syntax errors. This is ignored by interactive shells.

−o *option−name*

The *option−name* can be one of the following:

allexport

Same as **−a**.

braceexpand
Same as **−B**.

emacs Use an emacs-style command line editing interface. This is enabled by default when the shell is interactive, unless the shell is started with the **−−noediting** option. This also affects the editing interface used for **read −e**.

errexit Same as **−e**.

errtrace Same as **−E**.

functrace
Same as **−T**.

hashall Same as **−h**.

histexpand
Same as **−H**.

history Enable command history, as described above under **HISTORY**. This option is on by default in interactive shells.

ignoreeof
The effect is as if the shell command `IGNOREEOF=10` had been executed (see **Shell Variables** above).

keyword
Same as **−k**.

monitor Same as **−m**.

noclobber
Same as **−C**.

noexec Same as **−n**.

noglob Same as **−f**.

nolog Currently ignored.

notify Same as **−b**.

nounset Same as **−u**.

onecmd Same as **−t**.

physical Same as **−P**.

pipefail If set, the return value of a pipeline is the value of the last (rightmost) command to exit with a non-zero status, or zero if all commands in the pipeline exit successfully. This option is disabled by default.

posix Change the behavior of **bash** where the default operation differs from the POSIX standard to match the standard (*posix mode*). See **SEE ALSO** below for a reference to a document that details how posix mode affects bash's behavior.

privileged
Same as **−p**.

verbose Same as **−v**.

vi Use a vi-style command line editing interface. This also affects the editing interface used for **read −e**.

xtrace Same as **−x**.

If **−o** is supplied with no *option−name*, the values of the current options are printed. If **+o** is supplied with no *option−name*, a series of **set** commands to recreate the current option settings is displayed on the standard output.

−p Turn on *privileged* mode. In this mode, the **$ENV** and **$BASH_ENV** files are not processed, shell functions are not inherited from the environment, and the **SHELLOPTS**, **BASHOPTS**, **CDPATH**, and **GLOBIGNORE** variables, if they appear in the environment, are ignored. If the shell is started with the effective user (group) id not equal to the real user (group) id, and the **−p** option is not supplied, these actions are taken and the effective user id is set to the real user id. If the **−p** option is supplied at startup, the effective user id is not reset. Turning this option off causes the effective user and group ids to be set to the real user and group ids.

−t Exit after reading and executing one command.

−u Treat unset variables and parameters other than the special parameters "@" and "*" as an error when performing parameter expansion. If expansion is attempted on an unset variable or parameter, the shell prints an error message, and, if not interactive, exits with a non-zero status.

−v Print shell input lines as they are read.

−x After expanding each *simple command*, **for** command, **case** command, **select** command, or arithmetic **for** command, display the expanded value of **PS4**, followed by the command and its expanded arguments or associated word list.

−B The shell performs brace expansion (see **Brace Expansion** above). This is on by default.

−C If set, **bash** does not overwrite an existing file with the >, **>&**, and <> redirection operators. This may be overridden when creating output files by using the redirection operator >| instead of >.

−E If set, any trap on **ERR** is inherited by shell functions, command substitutions, and commands executed in a subshell environment. The **ERR** trap is normally not inherited in such cases.

−H Enable **!** style history substitution. This option is on by default when the shell is interactive.

−P If set, the shell does not resolve symbolic links when executing commands such as **cd** that change the current working directory. It uses the physical directory structure instead. By default, **bash** follows the logical chain of directories when performing commands which change the current directory.

−T If set, any traps on **DEBUG** and **RETURN** are inherited by shell functions, command substitutions, and commands executed in a subshell environment. The **DEBUG** and **RETURN** traps are normally not inherited in such cases.

−− If no arguments follow this option, then the positional parameters are unset. Otherwise, the positional parameters are set to the *arg*s, even if some of them begin with a **−**.

− Signal the end of options, cause all remaining *arg*s to be assigned to the positional parameters. The **−x** and **−v** options are turned off. If there are no *arg*s, the positional parameters remain unchanged.

The options are off by default unless otherwise noted. Using + rather than − causes these options to be turned off. The options can also be specified as arguments to an invocation of the shell. The current set of options may be found in **$−**. The return status is always true unless an invalid option is encountered.

shift [*n*]

The positional parameters from *n*+1 ... are renamed to **$1 ....** Parameters represented by the numbers **$#** down to **$#**−*n*+1 are unset. *n* must be a non-negative number less than or equal to **$#**. If *n* is 0, no parameters are changed. If *n* is not given, it is assumed to be 1. If *n* is greater than **$#**, the positional parameters are not changed. The return status is greater than zero if *n* is greater than **$#** or less than zero; otherwise 0.

shopt [**−pqsu**] [**−o**] [*optname* ...]

Toggle the values of settings controlling optional shell behavior. The settings can be either those listed below, or, if the **−o** option is used, those available with the **−o** option to the **set** builtin command. With no options, or with the **−p** option, a list of all settable options is displayed, with an indication of whether or not each is set. The **−p** option causes output to be displayed in a form that may be reused as input. Other options have the following meanings:

−s Enable (set) each *optname*.

−u Disable (unset) each *optname*.

−q Suppresses normal output (quiet mode); the return status indicates whether the *optname* is set or unset. If multiple *optname* arguments are given with **−q**, the return status is zero if all *optnames* are enabled; non-zero otherwise.

−o Restricts the values of *optname* to be those defined for the **−o** option to the **set** builtin.

If either **−s** or **−u** is used with no *optname* arguments, **shopt** shows only those options which are set or unset, respectively. Unless otherwise noted, the **shopt** options are disabled (unset) by default.

The return status when listing options is zero if all *optnames* are enabled, non-zero otherwise. When setting or unsetting options, the return status is zero unless an *optname* is not a valid shell option.

The list of **shopt** options is:

autocd If set, a command name that is the name of a directory is executed as if it were the argument to the **cd** command. This option is only used by interactive shells.

cdable_vars
If set, an argument to the **cd** builtin command that is not a directory is assumed to be the name of a variable whose value is the directory to change to.

cdspell If set, minor errors in the spelling of a directory component in a **cd** command will be corrected. The errors checked for are transposed characters, a missing character, and one character too many. If a correction is found, the corrected filename is printed, and the command proceeds. This option is only used by interactive shells.

checkhash
If set, **bash** checks that a command found in the hash table exists before trying to execute it. If a hashed command no longer exists, a normal path search is performed.

checkjobs
If set, **bash** lists the status of any stopped and running jobs before exiting an interactive shell. If any jobs are running, this causes the exit to be deferred until a second exit is attempted without an intervening command (see **JOB CONTROL** above). The shell always postpones exiting if any jobs are stopped.

checkwinsize
If set, **bash** checks the window size after each command and, if necessary, updates the values of **LINES** and **COLUMNS**.

cmdhist If set, **bash** attempts to save all lines of a multiple-line command in the same history entry. This allows easy re-editing of multi-line commands.

compat31
If set, **bash** changes its behavior to that of version 3.1 with respect to quoted arguments to the **[[** conditional command's =~ operator and locale-specific string comparison when using the **[[** conditional command's < and > operators. Bash versions prior to bash-4.1 use ASCII collation and *strcmp*(3); bash-4.1 and later use the current locale's collation sequence and *strcoll*(3).

compat32
If set, **bash** changes its behavior to that of version 3.2 with respect to locale-specific string comparison when using the **[[** conditional command's < and > operators (see previous item) and the effect of interrupting a command list. Bash versions 3.2 and earlier continue with the next command in the list after one terminates due to an interrupt.

compat40
If set, **bash** changes its behavior to that of version 4.0 with respect to locale-specific string comparison when using the **[[** conditional command's < and > operators (see description of **compat31**) and the effect of interrupting a command list. Bash versions 4.0 and later interrupt the list as if the shell received the interrupt; previous versions continue with the next command in the list.

compat41
If set, **bash**, when in *posix* mode, treats a single quote in a double-quoted parameter expansion as a special character. The single quotes must match (an even number) and the characters between the single quotes are considered quoted. This is the behavior of posix mode through version 4.1. The default bash behavior remains as in previous versions.

compat42
If set, **bash** does not process the replacement string in the pattern substitution word expansion using quote removal.

compat43
If set, **bash** does not print a warning message if an attempt is made to use a quoted compound array assignment as an argument to **declare**, makes word expansion errors non-fatal errors that cause the current command to fail (the default behavior is to make them fatal errors that cause the shell to exit), and does not reset the loop state when a shell function is executed (this allows **break** or **continue** in a shell function to affect loops in the caller's context).

complete_fullquote
If set, **bash** quotes all shell metacharacters in filenames and directory names when performing completion. If not set, **bash** removes metacharacters such as the dollar sign from the set of characters that will be quoted in completed filenames when these metacharacters appear in shell variable references in words to be completed. This means that dollar signs in variable names that expand to directories will not be quoted; however, any dollar signs appearing in filenames will not be quoted, either. This is active only when bash is using backslashes to quote completed filenames. This variable is set by default, which is the default bash behavior in versions through 4.2.

direxpand
If set, **bash** replaces directory names with the results of word expansion when performing filename completion. This changes the contents of the readline editing buffer. If not set, **bash** attempts to preserve what the user typed.

dirspell If set, **bash** attempts spelling correction on directory names during word completion if the directory name initially supplied does not exist.

dotglob If set, **bash** includes filenames beginning with a '.' in the results of pathname expansion.

execfail If set, a non-interactive shell will not exit if it cannot execute the file specified as an argument to the **exec** builtin command. An interactive shell does not exit if **exec** fails.

expand_aliases
If set, aliases are expanded as described above under **ALIASES**. This option is enabled by default for interactive shells.

extdebug
If set at shell invocation, arrange to execute the debugger profile before the shell starts, identical to the **−−debugger** option. If set after invocation, behavior intended for use by debuggers is enabled:

1. The **−F** option to the **declare** builtin displays the source file name and line number corresponding to each function name supplied as an argument.
2. If the command run by the **DEBUG** trap returns a non-zero value, the next command is skipped and not executed.
3. If the command run by the **DEBUG** trap returns a value of 2, and the shell is executing in a subroutine (a shell function or a shell script executed by the **.** or **source** builtins), the shell simulates a call to **return**.
4. **BASH_ARGC** and **BASH_ARGV** are updated as described in their descriptions above.
5. Function tracing is enabled: command substitution, shell functions, and subshells invoked with **(** *command* **)** inherit the **DEBUG** and **RETURN** traps.
6. Error tracing is enabled: command substitution, shell functions, and subshells invoked with **(** *command* **)** inherit the **ERR** trap.

extglob If set, the extended pattern matching features described above under **Pathname Expansion** are enabled.

extquote
If set, **$**'*string*' and **$**"*string*" quoting is performed within **${***parameter***}** expansions enclosed in double quotes. This option is enabled by default.

failglob If set, patterns which fail to match filenames during pathname expansion result in an expansion error.

force_fignore
If set, the suffixes specified by the **FIGNORE** shell variable cause words to be ignored when performing word completion even if the ignored words are the only possible completions. See **SHELL VARIABLES** above for a description of **FIGNORE**. This option is enabled by default.

globasciiranges
If set, range expressions used in pattern matching bracket expressions (see **Pattern Matching** above) behave as if in the traditional C locale when performing comparisons. That is, the current locale's collating sequence is not taken into account, so **b** will not collate between **A** and **B**, and upper-case and lower-case ASCII characters will collate together.

globstar If set, the pattern ****** used in a pathname expansion context will match all files and zero or more directories and subdirectories. If the pattern is followed by a /, only directories and subdirectories match.

gnu_errfmt
If set, shell error messages are written in the standard GNU error message format.

histappend
If set, the history list is appended to the file named by the value of the **HISTFILE** variable when the shell exits, rather than overwriting the file.

histreedit
If set, and **readline** is being used, a user is given the opportunity to re-edit a failed history substitution.

histverify
If set, and **readline** is being used, the results of history substitution are not immediately passed to the shell parser. Instead, the resulting line is loaded into the **readline** editing buffer, allowing further modification.

hostcomplete
If set, and **readline** is being used, **bash** will attempt to perform hostname completion when a word containing a **@** is being completed (see **Completing** under **READLINE** above). This is enabled by default.

huponexit
If set, **bash** will send **SIGHUP** to all jobs when an interactive login shell exits.

inherit_errexit
If set, command substitution inherits the value of the **errexit** option, instead of unsetting it in the subshell environment. This option is enabled when *posix mode* is enabled.

interactive_comments
If set, allow a word beginning with **#** to cause that word and all remaining characters on that line to be ignored in an interactive shell (see **COMMENTS** above). This option is enabled by default.

lastpipe If set, and job control is not active, the shell runs the last command of a pipeline not executed in the background in the current shell environment.

lithist If set, and the **cmdhist** option is enabled, multi-line commands are saved to the history with embedded newlines rather than using semicolon separators where possible.

login_shell
The shell sets this option if it is started as a login shell (see **INVOCATION** above). The value may not be changed.

mailwarn
If set, and a file that **bash** is checking for mail has been accessed since the last time it was checked, the message ''The mail in *mailfile* has been read'' is displayed.

no_empty_cmd_completion
If set, and **readline** is being used, **bash** will not attempt to search the **PATH** for possible completions when completion is attempted on an empty line.

nocaseglob
If set, **bash** matches filenames in a case−insensitive fashion when performing pathname expansion (see **Pathname Expansion** above).

nocasematch
If set, **bash** matches patterns in a case−insensitive fashion when performing matching while executing **case** or **[[** conditional commands, when performing pattern substitution word expansions, or when filtering possible completions as part of programmable completion.

nullglob
If set, **bash** allows patterns which match no files (see **Pathname Expansion** above) to expand to a null string, rather than themselves.

progcomp
If set, the programmable completion facilities (see **Programmable Completion** above) are enabled. This option is enabled by default.

promptvars
If set, prompt strings undergo parameter expansion, command substitution, arithmetic expansion, and quote removal after being expanded as described in **PROMPTING** above. This option is enabled by default.

restricted_shell
The shell sets this option if it is started in restricted mode (see **RESTRICTED SHELL** below). The value may not be changed. This is not reset when the startup files are executed, allowing the startup files to discover whether or not a shell is restricted.

shift_verbose
If set, the **shift** builtin prints an error message when the shift count exceeds the number of positional parameters.

sourcepath
If set, the **source** (**.**) builtin uses the value of **PATH** to find the directory containing the file supplied as an argument. This option is enabled by default.

xpg_echo
If set, the **echo** builtin expands backslash-escape sequences by default.

suspend [**−f**]
Suspend the execution of this shell until it receives a **SIGCONT** signal. A login shell cannot be suspended; the **−f** option can be used to override this and force the suspension. The return status is 0 unless the shell is a login shell and **−f** is not supplied, or if job control is not enabled.

test *expr*
[*expr* **]** Return a status of 0 (true) or 1 (false) depending on the evaluation of the conditional expression *expr*. Each operator and operand must be a separate argument. Expressions are composed of the primaries described above under **CONDITIONAL EXPRESSIONS**. **test** does not accept any options, nor does it accept and ignore an argument of **−−** as signifying the end of options.

Expressions may be combined using the following operators, listed in decreasing order of precedence. The evaluation depends on the number of arguments; see below. Operator precedence is used when there are five or more arguments.

! *expr* True if *expr* is false.

(*expr* **)** Returns the value of *expr*. This may be used to override the normal precedence of operators.

expr1 **−a** *expr2*
True if both *expr1* and *expr2* are true.

expr1 **−o** *expr2*
True if either *expr1* or *expr2* is true.

test and **[** evaluate conditional expressions using a set of rules based on the number of arguments.

0 arguments
: The expression is false.

1 argument
: The expression is true if and only if the argument is not null.

2 arguments
: If the first argument is **!**, the expression is true if and only if the second argument is null. If the first argument is one of the unary conditional operators listed above under **CONDITIONAL EXPRESSIONS**, the expression is true if the unary test is true. If the first argument is not a valid unary conditional operator, the expression is false.

3 arguments
: The following conditions are applied in the order listed. If the second argument is one of the binary conditional operators listed above under **CONDITIONAL EXPRESSIONS**, the result of the expression is the result of the binary test using the first and third arguments as operands. The **−a** and **−o** operators are considered binary operators when there are three arguments. If the first argument is **!**, the value is the negation of the two-argument test using the second and third arguments. If the first argument is exactly **(** and the third argument is exactly **)**, the result is the one-argument test of the second argument. Otherwise, the expression is false.

4 arguments
: If the first argument is **!**, the result is the negation of the three-argument expression composed of the remaining arguments. Otherwise, the expression is parsed and evaluated according to precedence using the rules listed above.

5 or more arguments
: The expression is parsed and evaluated according to precedence using the rules listed above.

When used with **test** or **[**, the < and > operators sort lexicographically using ASCII ordering.

times Print the accumulated user and system times for the shell and for processes run from the shell. The return status is 0.

trap [**−lp**] [[*arg*] *sigspec* ...]
: The command *arg* is to be read and executed when the shell receives signal(s) *sigspec*. If *arg* is absent (and there is a single *sigspec*) or **−**, each specified signal is reset to its original disposition (the value it had upon entrance to the shell). If *arg* is the null string the signal specified by each *sigspec* is ignored by the shell and by the commands it invokes. If *arg* is not present and **−p** has been supplied, then the trap commands associated with each *sigspec* are displayed. If no arguments are supplied or if only **−p** is given, **trap** prints the list of commands associated with each signal. The **−l** option causes the shell to print a list of signal names and their corresponding numbers. Each *sigspec* is either a signal name defined in <*signal.h*>, or a signal number. Signal names are case insensitive and the **SIG** prefix is optional.

If a *sigspec* is **EXIT** (0) the command *arg* is executed on exit from the shell. If a *sigspec* is **DEBUG**, the command *arg* is executed before every *simple command*, *for* command, *case* command, *select* command, every arithmetic *for* command, and before the first command executes in a shell function (see **SHELL GRAMMAR** above). Refer to the description of the **extdebug** option to the **shopt** builtin for details of its effect on the **DEBUG** trap. If a *sigspec* is **RETURN**, the command *arg* is executed each time a shell function or a script executed with the **.** or **source** builtins finishes executing.

If a *sigspec* is **ERR**, the command *arg* is executed whenever a pipeline (which may consist of a single simple command), a list, or a compound command returns a non−zero exit status, subject to the following conditions. The **ERR** trap is not executed if the failed command is part of the command list immediately following a **while** or **until** keyword, part of the test in an *if* statement, part of a command executed in a **&&** or || list except the command following the final **&&** or ||, any command in a pipeline but the last, or if the command's return value is being inverted using **!**. These are the same conditions obeyed by the **errexit** (**−e**) option.

Signals ignored upon entry to the shell cannot be trapped or reset. Trapped signals that are not being ignored are reset to their original values in a subshell or subshell environment when one is created. The return status is false if any *sigspec* is invalid; otherwise **trap** returns true.

type [**−aftpP**] *name* [*name* ...]

With no options, indicate how each *name* would be interpreted if used as a command name. If the **−t** option is used, **type** prints a string which is one of *alias*, *keyword*, *function*, *builtin*, or *file* if *name* is an alias, shell reserved word, function, builtin, or disk file, respectively. If the *name* is not found, then nothing is printed, and an exit status of false is returned. If the **−p** option is used, **type** either returns the name of the disk file that would be executed if *name* were specified as a command name, or nothing if `type −t name` would not return *file*. The **−P** option forces a **PATH** search for each *name*, even if `type −t name` would not return *file*. If a command is hashed, **−p** and **−P** print the hashed value, which is not necessarily the file that appears first in **PATH**. If the **−a** option is used, **type** prints all of the places that contain an executable named *name*. This includes aliases and functions, if and only if the **−p** option is not also used. The table of hashed commands is not consulted when using **−a**. The **−f** option suppresses shell function lookup, as with the **command** builtin. **type** returns true if all of the arguments are found, false if any are not found.

ulimit [**−HSabcdefiklmnpqrstuvxPT** [*limit*]]

Provides control over the resources available to the shell and to processes started by it, on systems that allow such control. The **−H** and **−S** options specify that the hard or soft limit is set for the given resource. A hard limit cannot be increased by a non-root user once it is set; a soft limit may be increased up to the value of the hard limit. If neither **−H** nor **−S** is specified, both the soft and hard limits are set. The value of *limit* can be a number in the unit specified for the resource or one of the special values **hard**, **soft**, or **unlimited**, which stand for the current hard limit, the current soft limit, and no limit, respectively. If *limit* is omitted, the current value of the soft limit of the resource is printed, unless the **−H** option is given. When more than one resource is specified, the limit name and unit are printed before the value. Other options are interpreted as follows:

- **−a** All current limits are reported
- **−b** The maximum socket buffer size
- **−c** The maximum size of core files created
- **−d** The maximum size of a process's data segment
- **−e** The maximum scheduling priority ("nice")
- **−f** The maximum size of files written by the shell and its children
- **−i** The maximum number of pending signals
- **−k** The maximum number of kqueues that may be allocated
- **−l** The maximum size that may be locked into memory
- **−m** The maximum resident set size (many systems do not honor this limit)
- **−n** The maximum number of open file descriptors (most systems do not allow this value to be set)
- **−p** The pipe size in 512-byte blocks (this may not be set)
- **−q** The maximum number of bytes in POSIX message queues
- **−r** The maximum real-time scheduling priority
- **−s** The maximum stack size
- **−t** The maximum amount of cpu time in seconds
- **−u** The maximum number of processes available to a single user
- **−v** The maximum amount of virtual memory available to the shell and, on some systems, to its children
- **−x** The maximum number of file locks
- **−P** The maximum number of pseudoterminals
- **−T** The maximum number of threads

If *limit* is given, and the **−a** option is not used, *limit* is the new value of the specified resource. If no option is given, then **−f** is assumed. Values are in 1024-byte increments, except for **−t**, which is in seconds; **−p**, which is in units of 512-byte blocks; **−P**, **−T**, **−b**, **−k**, **−n**, and **−u**, which are

unscaled values; and, when in Posix mode, **−c** and **−f**, which are in 512-byte increments. The return status is 0 unless an invalid option or argument is supplied, or an error occurs while setting a new limit.

umask [**−p**] [**−S**] [*mode*]
The user file-creation mask is set to *mode*. If *mode* begins with a digit, it is interpreted as an octal number; otherwise it is interpreted as a symbolic mode mask similar to that accepted by *chmod*(1). If *mode* is omitted, the current value of the mask is printed. The **−S** option causes the mask to be printed in symbolic form; the default output is an octal number. If the **−p** option is supplied, and *mode* is omitted, the output is in a form that may be reused as input. The return status is 0 if the mode was successfully changed or if no *mode* argument was supplied, and false otherwise.

unalias [**−a**] [*name* ...]
Remove each *name* from the list of defined aliases. If **−a** is supplied, all alias definitions are removed. The return value is true unless a supplied *name* is not a defined alias.

unset [**−fv**] [**−n**] [*name* ...]
For each *name*, remove the corresponding variable or function. If the **−v** option is given, each *name* refers to a shell variable, and that variable is removed. Read-only variables may not be unset. If **−f** is specified, each *name* refers to a shell function, and the function definition is removed. If the **−n** option is supplied, and *name* is a variable with the *nameref* attribute, *name* will be unset rather than the variable it references. **−n** has no effect if the **−f** option is supplied. If no options are supplied, each *name* refers to a variable; if there is no variable by that name, any function with that name is unset. Each unset variable or function is removed from the environment passed to subsequent commands. If any of **COMP_WORDBREAKS**, **RANDOM**, **SECONDS**, **LINENO**, **HISTCMD**, **FUNCNAME**, **GROUPS**, or **DIRSTACK** are unset, they lose their special properties, even if they are subsequently reset. The exit status is true unless a *name* is readonly.

wait [**−n**] [*n* ...]
Wait for each specified child process and return its termination status. Each *n* may be a process ID or a job specification; if a job spec is given, all processes in that job's pipeline are waited for. If *n* is not given, all currently active child processes are waited for, and the return status is zero. If the **−n** option is supplied, **wait** waits for any job to terminate and returns its exit status. If *n* specifies a non-existent process or job, the return status is 127. Otherwise, the return status is the exit status of the last process or job waited for.

RESTRICTED SHELL

If **bash** is started with the name **rbash**, or the **−r** option is supplied at invocation, the shell becomes restricted. A restricted shell is used to set up an environment more controlled than the standard shell. It behaves identically to **bash** with the exception that the following are disallowed or not performed:

- changing directories with **cd**
- setting or unsetting the values of **SHELL**, **PATH**, **ENV**, or **BASH_ENV**
- specifying command names containing **/**
- specifying a filename containing a **/** as an argument to the **.** builtin command
- specifying a filename containing a slash as an argument to the **−p** option to the **hash** builtin command
- importing function definitions from the shell environment at startup
- parsing the value of **SHELLOPTS** from the shell environment at startup
- redirecting output using the >, >|, <>, >&, &>, and >> redirection operators
- using the **exec** builtin command to replace the shell with another command
- adding or deleting builtin commands with the **−f** and **−d** options to the **enable** builtin command
- using the **enable** builtin command to enable disabled shell builtins

- specifying the **−p** option to the **command** builtin command
- turning off restricted mode with **set +r** or **set +o restricted**.

These restrictions are enforced after any startup files are read.

When a command that is found to be a shell script is executed (see **COMMAND EXECUTION** above), **rbash** turns off any restrictions in the shell spawned to execute the script.

SEE ALSO

Bash Reference Manual, Brian Fox and Chet Ramey
The Gnu Readline Library, Brian Fox and Chet Ramey
The Gnu History Library, Brian Fox and Chet Ramey
Portable Operating System Interface (POSIX) Part 2: Shell and Utilities, IEEE -- http://pubs.opengroup.org/onlinepubs/9699919799/
http://tiswww.case.edu/~chet/bash/POSIX -- a description of posix mode
sh(1), *ksh*(1), *csh*(1)
emacs(1), *vi*(1)
readline(3)

FILES

/bin/bash
The **bash** executable
/etc/profile
The systemwide initialization file, executed for login shells
/etc/bash.bashrc
The systemwide per-interactive-shell startup file
/etc/bash.bash.logout
The systemwide login shell cleanup file, executed when a login shell exits
~/.bash_profile
The personal initialization file, executed for login shells
~/.bashrc
The individual per-interactive-shell startup file
~/.bash_logout
The individual login shell cleanup file, executed when a login shell exits
~/.inputrc
Individual *readline* initialization file

AUTHORS

Brian Fox, Free Software Foundation
bfox@gnu.org

Chet Ramey, Case Western Reserve University
chet.ramey@case.edu

BUG REPORTS

If you find a bug in **bash,** you should report it. But first, you should make sure that it really is a bug, and that it appears in the latest version of **bash**. The latest version is always available from *ftp://ftp.gnu.org/pub/gnu/bash/*.

Once you have determined that a bug actually exists, use the *bashbug* command to submit a bug report. If you have a fix, you are encouraged to mail that as well! Suggestions and ‘philosophical’ bug reports may be mailed to *bug-bash@gnu.org* or posted to the Usenet newsgroup **gnu.bash.bug**.

ALL bug reports should include:

The version number of **bash**
The hardware and operating system
The compiler used to compile

A description of the bug behaviour
A short script or ‘recipe’ which exercises the bug

bashbug inserts the first three items automatically into the template it provides for filing a bug report.

Comments and bug reports concerning this manual page should be directed to *chet.ramey@case.edu*.

BUGS

It’s too big and too slow.

There are some subtle differences between **bash** and traditional versions of **sh**, mostly because of the **POSIX** specification.

Aliases are confusing in some uses.

Shell builtin commands and functions are not stoppable/restartable.

Compound commands and command sequences of the form ‘a ; b ; c’ are not handled gracefully when process suspension is attempted. When a process is stopped, the shell immediately executes the next command in the sequence. It suffices to place the sequence of commands between parentheses to force it into a subshell, which may be stopped as a unit.

Array variables may not (yet) be exported.

There may be only one active coprocess at a time.

NAME

cat − concatenate files and print on the standard output

SYNOPSIS

cat [*OPTION*]... [*FILE*]...

DESCRIPTION

Concatenate FILE(s) to standard output.

With no FILE, or when FILE is −, read standard input.

−A, **−−show−all**
equivalent to **−vET**

−b, **−−number−nonblank**
number nonempty output lines, overrides **−n**

−e equivalent to **−vE**

−E, **−−show−ends**
display $ at end of each line

−n, **−−number**
number all output lines

−s, **−−squeeze−blank**
suppress repeated empty output lines

−t equivalent to **−vT**

−T, **−−show−tabs**
display TAB characters as ^I

−u (ignored)

−v, **−−show−nonprinting**
use ^ and M− notation, except for LFD and TAB

−−help display this help and exit

−−version
output version information and exit

EXAMPLES

cat f − g
Output f’s contents, then standard input, then g’s contents.

cat Copy standard input to standard output.

AUTHOR

Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Report cat translation bugs to <http://translationproject.org/team/>

COPYRIGHT

SEE ALSO

tac(1)

Full documentation at: <http://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

NAME

chmod − altera a permissões de acesso aos arquivos

SINOPSE

chmod [*opções***]** *modo arquivo...*

Opções POSIX: **[−R]**

Opções GNU (forma reduzida): **[−cfvR] [−−reference=***rarquivo***] [−−help] [−−version] [−−]**

DESCRIÇÃO

chmod altera a permissão para cada *arquivo* fornecido de acordo com *modo*, que ou pode ser uma representação simbólica das mudanças a serem feitas, ou um número octal que representa um padrão de bits para as novas permissões.

O formato de um argumento de mudança no modo simbólico é ‘[ugoa...][[+−=][rwxXstugo...]...][,...]’.

Tal argumento é uma lista de comandos de mudança em modo simbólico, separada por vírgulas. Cada comando de mudança simbólica começa com um zero, uma ou mais das letras ‘ugoa’; estas controlam qual acesso de usuário para o arquivo será alterado: o usuário que é proprietário do arquivo (u), outros usuários no grupo do arquivo (g), demais usuários do arquivo (o), ou todos os usuários (a). Assim, ’a’ é equivalente a ’ugo’. Se nenhum destes forem fornecidos, o efeito é o mesmo que se ’a’ fosse fornecido, mas os bits selecionados na umask não são afetados.

O operador ‘+’ faz com que as permissões selecionadas sejam adicionadas as já existentes em cada arquivo; ‘−’ faz com que sejam removidas; e ‘=’ troca as permissões existentes para as informadas.

As letras ‘rwxXstugo’ selecionam as novas permissões para o usuários afetados: ler (r), gravar (w), executar (ou acesso para diretórios) (x), executa só se o arquivo ou diretório já tem permissão de execução para algum usuário (X), seleciona o usuário ou identificação do grupo durante a execução (s), bit contrário (t), as permissões que o usuário proprietário do arquivo têm (u), as permissões que outros usuários do grupo do arquivo têm (g), e as permissões que os demais usuários têm (o). (Deste modo, ’chmod g−s arquivo’ remove o bit que seleciona a identificação do grupo (sgid), ’chmod ug+s arquivo’ marca o bit suid tanto quanto o sgid, enquanto ’chmod o+s arquivo’ não faz nada.)

O ’bit contrário’ não é descrito pela POSIX. O nome deriva do significado original: manter texto de programa em dispositivo de troca. Atualmente, quando selecionado para um diretório, significa que só o proprietário do arquivo e o proprietário daquele diretório pode remover o arquivo daquele diretório. (Isto é comumente usado em diretórios como /tmp que tem permissão geral de escrita.)

Um modo numérico é de um a quatro dígitos octais (0-7), derivados da adição dos bits com valores 4, 2, e 1. Quaisquer dígitos omitidos são assumidos como zero. O primeiro dígito seleciona a identificação do usuário (4) e a seleção do grupo (2) exceto imagem de texto [’Sticky’] (1) atributos. O segundo dígito seleciona permissões para o proprietário do arquivo: ler (4), escrever (2), e executar (1); o terçeiro seleciona permissões para os usuários pertencentes ao grupo do arquivo, com os mesmos valores; e o quarto, para os demais usuários, com os mesmos valores.

chmod nunca altera a permissões de ligações simbólicas, então a chamada de sistema **chmod** não pode fazê-lo. Isto não é um problema desde que as permissões de ligações simbólicas nunca sejam utilizadas. Porém, para cada ligação simbólica listada na linha de comando, **chmod** altera a permissão no arquivo apontado. Por outro lado, **chmod** ignora ligações simbólicas encontradas durante a opção recursiva no diretório.

OPÇÕES POSIX

−R Altera recursivamente as permissões dos diretórios e de seus conteúdos.

OPÇÕES GNU

−c, −−changes

Detalhadamente descreve a ação para cada *arquivo* cujas permissões estão sendo alteradas.

−f, −−silent, −−quiet
Não mostra mensagens de erro para o arquivo cujas permissões não podem ser alteradas.

−v, −−verbose
Descreve detalhadamente toda ação ocorrida para cada *arquivo*.

−R, −−recursive
Altera recursivamente a permissão dos diretórios e de seus conteúdos.

−−reference=*rarquivo*
(Novo no Utilitários de Arquivo GNU 4.0.) Altera o modo do *arquivo* para aquele do *rarquivo*.

OPÇÕES PADRÃO GNU

−−help Imprime a mensagem de uso na saída padrão e sai.

−−version
Imprime a versão na saída padrão e sai.

−− Encerra a lista de opção.

AMBIENTE

As variáveis LANG, LC_ALL, LC_CTYPE and LC_MESSAGES tem seu significado usual.

DE ACORDO COM

POSIX 1003.2 somente requer a opção −R. Uso de outras opções podem não ser portáveis. Este padrão não descreve a permissão do bit consistência limpando ou marcando os bits suid e sgid, isto é, quando todo os bits executados são limpos, ou se **chmod** honra o bit ‘S’ completamente.

MODOS NÃO PADRÃO

Acima nós descrevemos o uso do bit ‘t’ nos diretórios. Vários sistemas anexam significados especiais para conbinações de bits de modo sem significação. Em particular, o Linux, imita o System V (veja a definição de interface do System V (SVID) Versão 3), deixa o bit sgid para que arquivos sem permissão de execução de grupo marquem o arquivo para fechamento obrigatório. Para mais detalhes, veja o arquivo */usr/src/linux/Documentation/mandatory.txt*.

NOTAS

Esta página descreve **chmod** como é encontrado no pacote Utilitários de Arquivo 4.0; outras versões podem ser um pouco diferentes. Envie correções e adições para aeb@cwi.nl. Relatório de problemas no programa para fileutils-bugs@gnu.ai.mit.edu.

TRADUZIDO POR LDP-BR em 21/08/2000.

André L. Fassone Canova <lonelywolf@blv.com.br> (tradução) Ricardo C.O. Freitas <english.quest@best−service.com> (revisão)

NAME

clang − the Clang C, C++, and Objective-C compiler

SYNOPSIS

clang [*options*] *filename ...*

DESCRIPTION

clang is a C, C++, and Objective−C compiler which encompasses preprocessing, parsing, optimization, code generation, assembly, and linking. Depending on which high−level mode setting is passed, Clang will stop before doing a full link. While Clang is highly integrated, it is important to understand the stages of compilation, to understand how to invoke it. These stages are:

Driver The clang executable is actually a small driver which controls the overall execution of other tools such as the compiler, assembler and linker. Typically you do not need to interact with the driver, but you transparently use it to run the other tools.

Preprocessing

This stage handles tokenization of the input source file, macro expansion, #include expansion and handling of other preprocessor directives. The output of this stage is typically called a ".i" (for C), ".ii" (for C++), ".mi" (for Objective−C), or ".mii" (for Objective−C++) file.

Parsing and Semantic Analysis

This stage parses the input file, translating preprocessor tokens into a parse tree. Once in the form of a parse tree, it applies semantic analysis to compute types for expressions as well and determine whether the code is well formed. This stage is responsible for generating most of the compiler warnings as well as parse errors. The output of this stage is an "Abstract Syntax Tree" (AST).

Code Generation and Optimization

This stage translates an AST into low−level intermediate code (known as "LLVM IR") and ultimately to machine code. This phase is responsible for optimizing the generated code and handling target−specific code generation. The output of this stage is typically called a ".s" file or "assembly" file.

Clang also supports the use of an integrated assembler, in which the code generator produces object files directly. This avoids the overhead of generating the ".s" file and of calling the target assembler.

Assembler

This stage runs the target assembler to translate the output of the compiler into a target object file. The output of this stage is typically called a ".o" file or "object" file.

Linker This stage runs the target linker to merge multiple object files into an executable or dynamic library. The output of this stage is typically called an "a.out", ".dylib" or ".so" file.

Clang Static Analyzer

The Clang Static Analyzer is a tool that scans source code to try to find bugs through code analysis. This tool uses many parts of Clang and is built into the same driver. Please see <*http://clang−analyzer.llvm.org*> for more details on how to use the static analyzer.

OPTIONS

Stage Selection Options

−E Run the preprocessor stage.

−fsyntax−only

Run the preprocessor, parser and type checking stages.

−S Run the previous stages as well as LLVM generation and optimization stages and target−specific code generation, producing an assembly file.

−c Run all of the above, plus the assembler, generating a target ".o" object file.

no stage selection option
If no stage selection option is specified, all stages above are run, and the linker is run to combine the results into an executable or shared library.

Language Selection and Mode Options

−x <language>
Treat subsequent input files as having type language.

−std=<language>
Specify the language standard to compile for.

−stdlib=<library>
Specify the C++ standard library to use; supported options are libstdc++ and libc++.

−ansi Same as −std=c89.

−ObjC, −ObjC++
Treat source input files as Objective−C and Object−C++ inputs respectively.

−trigraphs
Enable trigraphs.

−ffreestanding
Indicate that the file should be compiled for a freestanding, not a hosted, environment.

−fno−builtin
Disable special handling and optimizations of builtin functions like **strlen()** and **malloc()**.

−fmath−errno
Indicate that math functions should be treated as updating **errno**.

−fpascal−strings
Enable support for Pascal−style strings with "\pfoo".

−fms−extensions
Enable support for Microsoft extensions.

−fmsc−version=
Set _MSC_VER. Defaults to 1300 on Windows. Not set otherwise.

−fborland−extensions
Enable support for Borland extensions.

−fwritable−strings
Make all string literals default to writable. This disables uniquing of strings and other optimizations.

−flax−vector−conversions
Allow loose type checking rules for implicit vector conversions.

−fblocks
Enable the "Blocks" language feature.

−fobjc−gc−only
Indicate that Objective−C code should be compiled in GC−only mode, which only works when Objective−C Garbage Collection is enabled.

−fobjc−gc
Indicate that Objective−C code should be compiled in hybrid−GC mode, which works with both GC and non−GC mode.

−fobjc−abi−version=version
Select the Objective−C ABI version to use. Available versions are 1 (legacy "fragile" ABI), 2 (non−fragile ABI 1), and 3 (non−fragile ABI 2).

−fobjc−nonfragile−abi−version=<version>
Select the Objective−C non−fragile ABI version to use by default. This will only be used as the Objective−C ABI when the non−fragile ABI is enabled (either via *−fobjc−nonfragile−abi*, or because it is the platform default).

−fobjc−nonfragile−abi
Enable use of the Objective−C non−fragile ABI. On platforms for which this is the default ABI, it can be disabled with **−fno−objc−nonfragile−abi**.

Target Selection Options
Clang fully supports cross compilation as an inherent part of its design. Depending on how your version of Clang is configured, it may have support for a number of cross compilers, or may only support a native target.

−arch <architecture>
Specify the architecture to build for.

−mmacosx−version−min=<version>
When building for Mac OS X, specify the minimum version supported by your application.

−miphoneos−version−min
When building for iPhone OS, specify the minimum version supported by your application.

−march=<cpu>
Specify that Clang should generate code for a specific processor family member and later. For example, if you specify −march=i486, the compiler is allowed to generate instructions that are valid on i486 and later processors, but which may not exist on earlier ones.

Code Generation Options
−O0, −O1, −O2, −O3, −Ofast, −Os, −Oz, −O, −O4
Specify which optimization level to use:
−O0 Means "no optimization": this level compiles the fastest and generates the most debuggable code.

−O1 Somewhere between *−O0* and *−O2*.

−O2 Moderate level of optimization which enables most optimizations.

−O3 Like *−O2*, except that it enables optimizations that take longer to perform or that may generate larger code (in an attempt to make the program run faster).

−Ofast Enables all the optimizations from *−O3* along with other aggressive optimizations that may violate strict compliance with language standards.

−Os Like *−O2* with extra optimizations to reduce code size.

−Oz Like *−Os* (and thus *−O2*), but reduces code size further.

−O Equivalent to *−O2*.

−O4 and higher
Currently equivalent to *−O3*

−g Generate debug information. Note that Clang debug information works best at −O0.

−gmodules
Generate debug information that contains external references to types defined in clang modules or precompiled headers instead of emitting redundant debug type information into every object file. This option implies **−fmodule−format=obj**.

This option should not be used when building static libraries for distribution to other machines because the debug info will contain references to the module cache on the machine the object files in the library were built on.

−fstandalone−debug −fno−standalone−debug
Clang supports a number of optimizations to reduce the size of debug information in the binary. They work based on the assumption that the debug type information can be spread out over multiple compilation units. For instance, Clang will not emit type definitions for types that are not needed by a module and could be replaced with a forward declaration. Further, Clang will only emit type info for a dynamic C++ class in the module that contains the vtable for the class.

The **−fstandalone−debug** option turns off these optimizations. This is useful when working with 3rd−party libraries that don't come with debug information. This is the default on Darwin. Note that Clang will never emit type information for types that are not referenced at all by the program.

−fexceptions
Enable generation of unwind information. This allows exceptions to be thrown through Clang compiled stack frames. This is on by default in x86−64.

−ftrapv
Generate code to catch integer overflow errors. Signed integer overflow is undefined in C. With this flag, extra code is generated to detect this and abort when it happens.

−fvisibility
This flag sets the default visibility level.

−fcommon
This flag specifies that variables without initializers get common linkage. It can be disabled with **−fno−common**.

−ftls−model=<model>
Set the default thread−local storage (TLS) model to use for thread−local variables. Valid values are: "global−dynamic", "local−dynamic", "initial−exec" and "local−exec". The default is "global−dynamic". The default model can be overridden with the tls_model attribute. The compiler will try to choose a more efficient model if possible.

−flto, −emit−llvm
Generate output files in LLVM formats, suitable for link time optimization. When used with *−S* this generates LLVM intermediate language assembly files, otherwise this generates LLVM bitcode format object files (which may be passed to the linker depending on the stage selection options).

Driver Options

−### Print (but do not run) the commands to run for this compilation.

−−help Display available options.

−Qunused−arguments
Do not emit any warnings for unused driver arguments.

−Wa,<args>
Pass the comma separated arguments in args to the assembler.

−Wl,<args>
Pass the comma separated arguments in args to the linker.

−Wp,<args>
Pass the comma separated arguments in args to the preprocessor.

−Xanalyzer <arg>
Pass arg to the static analyzer.

−Xassembler <arg>
Pass arg to the assembler.

−Xlinker <arg>
Pass arg to the linker.

−Xpreprocessor <arg>
Pass arg to the preprocessor.

−o <file>
Write output to file.

−print−file−name=<file>
Print the full library path of file.

−print−libgcc−file−name
Print the library path for "libgcc.a".

−print−prog−name=<name>
Print the full program path of name.

−print−search−dirs
Print the paths used for finding libraries and programs.

−save−temps
Save intermediate compilation results.

−integrated−as, −no−integrated−as
Used to enable and disable, respectively, the use of the integrated assembler. Whether the integrated assembler is on by default is target dependent.

−time Time individual commands.

−ftime−report
Print timing summary of each stage of compilation.

−v Show commands to run and use verbose output.

Diagnostics Options

−fshow−column, −fshow−source−location, −fcaret−diagnostics, −fdiagnostics−fixit−info, −fdiagnostics−parseable−fixits, −fdiagnostics−print−source−range−info, −fprint−source−range−info, −fdiagnostics−show−option, −fmessage−length
These options control how Clang prints out information about diagnostics (errors and warnings). Please see the Clang User's Manual for more information.

Preprocessor Options

−D<macroname>=<value>
Adds an implicit #define into the predefines buffer which is read before the source file is preprocessed.

−U<macroname>
Adds an implicit #undef into the predefines buffer which is read before the source file is preprocessed.

−include <filename>
Adds an implicit #include into the predefines buffer which is read before the source file is preprocessed.

−I<directory>
Add the specified directory to the search path for include files.

−F<directory>
Add the specified directory to the search path for framework include files.

−nostdinc
Do not search the standard system directories or compiler builtin directories for include files.

−nostdlibinc
Do not search the standard system directories for include files, but do search compiler builtin include directories.

−nobuiltininc
Do not search clang's builtin directory for include files.

ENVIRONMENT

TMPDIR, TEMP, TMP
These environment variables are checked, in order, for the location to write temporary files used during the compilation process.

CPATH
If this environment variable is present, it is treated as a delimited list of paths to be added to the default system include path list. The delimiter is the platform dependent delimiter, as used in the PATH environment variable.

Empty components in the environment variable are ignored.

C_INCLUDE_PATH, OBJC_INCLUDE_PATH, CPLUS_INCLUDE_PATH, OBJCPLUS_INCLUDE_PATH
These environment variables specify additional paths, as for *CPATH*, which are only used when processing the appropriate language.

MACOSX_DEPLOYMENT_TARGET
If *−mmacosx−version−min* is unspecified, the default deployment target is read from this environment variable. This option only affects Darwin targets.

BUGS

To report bugs, please visit <*http://llvm.org/bugs/*>. Most bug reports should include preprocessed source files (use the *−E* option) and the full output of the compiler, along with information to reproduce.

SEE ALSO

as(1), **ld(1)**

AUTHOR

Maintained by the Clang / LLVM Team (<http://clang.llvm.org>)

COPYRIGHT

NAME

cut − remove sections from each line of files

SYNOPSIS

cut *OPTION*... [*FILE*]...

DESCRIPTION

Print selected parts of lines from each FILE to standard output.

With no FILE, or when FILE is −, read standard input.

Mandatory arguments to long options are mandatory for short options too.

−b, **−−bytes**=*LIST*
: select only these bytes

−c, **−−characters**=*LIST*
: select only these characters

−d, **−−delimiter**=*DELIM*
: use DELIM instead of TAB for field delimiter

−f, **−−fields**=*LIST*
: select only these fields; also print any line that contains no delimiter character, unless the **−s** option is specified

−n (ignored)

−−complement
: complement the set of selected bytes, characters or fields

−s, **−−only−delimited**
: do not print lines not containing delimiters

−−output−delimiter=*STRING*
: use STRING as the output delimiter the default is to use the input delimiter

−z, **−−zero−terminated**
: line delimiter is NUL, not newline

−−help display this help and exit

−−version
: output version information and exit

Use one, and only one of **−b**, **−c** or **−f**. Each LIST is made up of one range, or many ranges separated by commas. Selected input is written in the same order that it is read, and is written exactly once. Each range is one of:

N N’th byte, character or field, counted from 1

N− from N’th byte, character or field, to end of line

N−M from N’th to M’th (included) byte, character or field

−M from first to M’th (included) byte, character or field

AUTHOR

Written by David M. Ihnat, David MacKenzie, and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Report cut translation bugs to <http://translationproject.org/team/>

COPYRIGHT

permitted by law.

SEE ALSO

Full documentation at: <http://www.gnu.org/software/coreutils/cut>
or available locally via: info '(coreutils) cut invocation'

NOME

dd − converte e copia um arquivo

SINOPSE

dd [−−help] [−−version] [if=*arquivo***] [of=***arquivo***] [ibs=***bytes***] [obs=***bytes***] [bs=***bytes***] [cbs=***bytes***] [skip=***blocos***] [seek=***blocos***] [count=***blocos***] [conv={ascii, ebcdic, ibm, block, unblock, lcase, ucase, swab, noerror, notrunc, sync}]**

DESCRIÇÃO

dd copia um arquivo (da entrada padrão para a saída padrão, por padrão) usando tamanhos de blocos de entrada e saída especificados, enquanto está fazendo opcionalmente conversões nele.

Ele lê um bloco de entrada de uma vez, usando o tamanho especificado de um bloco de entrada (o padrão é 512 bytes). Se a opção **bs=***bytes* é fornecida, e nenhuma conversão a não ser **sync**, **noerror**, ou **notrunc** são especificadas, ele escreve a quantia de dados lidos (o qual pode ser menor que o requerido) num bloco de saída separada. Este bloco de saída tem precisamente o mesmo tamanho que o lido a menos que a conversão **sync** seja especificada, neste caso os dados serão preenchidos com NULOS (ou espaços, veja abaixo).

De outra forma, a entrada, lê um bloco de uma vez, é processada e o resultado é coletado e escrito em blocos com o tamanho de bloco de saída especificado. O bloco final de saída pode ser menor.

As opções de valor numérico abaixo (bytes e blocos) podem ser seguidas de multiplicadores: ’k’=1024, ’b’=512, ’w’=2, ’c’=1 (’w’ e ’c’ são extensões GNU; ’w’ nunca deve ser usada - significa 2 em System V e 4 no 4.2BSD). Duas ou mais tais expressões numéricas podem ser multiplicas colocando ’x’ entre elas. A versão do Utilitários de Arquivo GNU 4.0 também permite os seguintes sufixos multiplicativos na especificação de tamanho de bloco (no bs=, cbs=, ibs=, obs=): M=1048576, G=1073741824, e assim para T, P, E, Z, Y. Um sufixo ’D’ torna-os decimais: kD=1000, MD=1000000, GD=1000000000, etc. (Note que para ls, df, du o tamanho de M etc. é determinado por variáveis de ambiente, mas para dd ela é fixa.)

OPÇÕES

if=*arquivo*

Lê a partir do *arquivo* ao invés da entrada padrão.

of=*arquivo*

Escreve no *arquivo* ao invés da saída padrão. A menos que **conv=notrunc** seja fornecido, **dd** trunca o *arquivo* para zero bytes (ou para o tamanho especificado com **seek=**).

ibs=*bytes*

Lê a quantidade de *bytes* de uma vez. O padrão é 512.

obs=*bytes*

Escreve a quantidade de *bytes* de uma vez. O padrão é 512.

bs=*bytes*

Lê e escreve a quantidade de *bytes* de uma vez. Isto cancela **ibs** e **obs**. (E selecionando **bs** não é equivalente com a seleção de ambos **ibs** e **obs** para este mesmo valor, pelo menos quando nenhuma conversão exceto **sync**, **noerror** e **notrunc** é especificada, desde que ela estipula que cada bloco de entrada será copiado para a saída como um único bloco sem agregar blocos pequenos.)

cbs=*bytes*

Especifica a conversão de tamanho de bloco para **bloco** e **não_bloco**.

skip=*blocps*

Salta *blocos* **ibs**-byte blocos no arquivo de entrada antes da cópia.

seek=*blocos*

Salta *blocos* **obs**-byte blocos no arquivo de saída antes da cópia.

count=*blocks*

Copia *blocos* **ibs**-byte blocos do arquivo de entrada, ao invés de tudo até o final do arquivo.

conv=*CONVERSÃO***[,***CONVERSÃO***]...**

Converte o arquivo conforme especificado pelo(s) argumento(s) de *CONVERSÃO*

Conversões:

ascii Converte EBCDIC para ASCII.

ebcdic Converte ASCII para EBCDIC.

ibm Converte ASCII para EBCDIC alternado.

block Para cada linha na entrada, a saída tem **cbs** bytes, recolocando a nova linha de entrada com espaço e enchendo com espaços se necessário.

unblock
Recoloca espaços em cada bloco de entrada **cbs**-sized como uma nova linha.

lcase Altera letras maiúsculas para minúsculas.

ucase Altera letras minúsculas para maiúsculas.

swab Troca todos os pares de bytes de entrada. Se um número impar de bytes são lidos o último é simplesmente copiado (desde que não haja troca com ele). [POSIX 1003.2b, interpretações PASC 1003.2 #3 and #4]

noerror
Continua depois de erros de leitura.

notrunc
Não trunca o arquivo de saída.

sync Enche todos os blocos de entradas para o tamanho de **ibs** com valores de zero no final.

OPÇÕES PADRÃO GNU

−−help Imprime a mensagem de uso na saída padrão e sai.

−−version
Imprime a versão na saída padrão e sai.

−− Encerra a lista de opção.

AMBIENTE

As variáveis LANG, LC_ALL, LC_CTYPE and LC_MESSAGES tem seu significado usual.

DE ACORDO COM

POSIX 1003.2

EXEMPLO

Freqüentemente um controlador de fita não aceita tamanhos de blocos arbitrariamente, e **dd** induziria um erro de E/S para o último fragmento de dados que não ocupa um bloco completo. Use ’dd if=meu_arquivo of=/dev/mytape conv=sync’ para conseguir tudo na fita. É claro, lendo ele novamente você verá um arquivo um pouco maior, com zeros adicionados no fim.

NOTAS

Esta página descreve **dd** como é encontrada no pacote Utilitários de Arquivo 4.0; outras versões podem ser um pouco diferentes. Envie correções e adições para aeb@cwi.nl. Relatório de problemas no programa para fileutils-bugs@gnu.ai.mit.edu.

TRADUZIDO POR LDP-BR em 21/08/2000.

André L. Fassone Canova <lonelywolf@blv.com.br> (tradução) Ricardo C.O. Freitas <english.quest@best-service.com> (revisão)

NAME

find − search for files in a directory hierarchy

SYNOPSIS

find [−H] [−L] [−P] [−D debugopts] [−Olevel] [starting-point...] [expression]

DESCRIPTION

This manual page documents the GNU version of **find**. GNU **find** searches the directory tree rooted at each given starting-point by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for *and* operations, true for *or*), at which point **find** moves on to the next file name. If no starting-point is specified, '.' is assumed.

If you are using **find** in an environment where security is important (for example if you are using it to search directories that are writable by other users), you should read the 'Security Considerations' chapter of the findutils documentation, which is called **Finding Files** and comes with findutils. That document also includes a lot more detail and discussion than this manual page, so you may find it a more useful source of information.

OPTIONS

The **−H**, **−L** and **−P** options control the treatment of symbolic links. Command-line arguments following these are taken to be names of files or directories to be examined, up to the first argument that begins with '−', or the argument '(' or '!'. That argument and any following arguments are taken to be the expression describing what is to be searched for. If no paths are given, the current directory is used. If no expression is given, the expression **−print** is used (but you should probably consider using **−print0** instead, anyway).

This manual page talks about 'options' within the expression list. These options control the behaviour of **find** but are specified immediately after the last path name. The five 'real' options **−H**, **−L**, **−P**, **−D** and **−O** must appear before the first path name, if at all. A double dash −− can also be used to signal that any remaining arguments are not options (though ensuring that all start points begin with either './' or '/' is generally safer if you use wildcards in the list of start points).

−P Never follow symbolic links. This is the default behaviour. When **find** examines or prints information a file, and the file is a symbolic link, the information used shall be taken from the properties of the symbolic link itself.

−L Follow symbolic links. When **find** examines or prints information about files, the information used shall be taken from the properties of the file to which the link points, not from the link itself (unless it is a broken symbolic link or **find** is unable to examine the file to which the link points). Use of this option implies **−noleaf**. If you later use the **−P** option, **−noleaf** will still be in effect. If **−L** is in effect and **find** discovers a symbolic link to a subdirectory during its search, the subdirectory pointed to by the symbolic link will be searched.

When the **−L** option is in effect, the **−type** predicate will always match against the type of the file that a symbolic link points to rather than the link itself (unless the symbolic link is broken). Actions that can cause symbolic links to become broken while **find** is executing (for example **−delete**) can give rise to confusing behaviour. Using **−L** causes the **−lname** and **−ilname** predicates always to return false.

−H Do not follow symbolic links, except while processing the command line arguments. When **find** examines or prints information about files, the information used shall be taken from the properties of the symbolic link itself. The only exception to this behaviour is when a file specified on the command line is a symbolic link, and the link can be resolved. For that situation, the information used is taken from whatever the link points to (that is, the link is followed). The information about the link itself is used as a fallback if the file pointed to by the symbolic link cannot be examined. If **−H** is in effect and one of the paths specified on the command line is a symbolic link to a directory, the contents of that directory will be examined (though of course −maxdepth 0 would prevent this).

If more than one of **−H**, **−L** and **−P** is specified, each overrides the others; the last one appearing on the command line takes effect. Since it is the default, the **−P** option should be considered to be in effect unless either **−H** or **−L** is specified.

GNU **find** frequently stats files during the processing of the command line itself, before any searching has begun. These options also affect how those arguments are processed. Specifically, there are a number of tests that compare files listed on the command line against a file we are currently considering. In each case, the file specified on the command line will have been examined and some of its properties will have been saved. If the named file is in fact a symbolic link, and the **−P** option is in effect (or if neither **−H** nor **−L** were specified), the information used for the comparison will be taken from the properties of the symbolic link. Otherwise, it will be taken from the properties of the file the link points to. If **find** cannot follow the link (for example because it has insufficient privileges or the link points to a nonexistent file) the properties of the link itself will be used.

When the **−H** or **−L options are in effect, any symbolic links listed** as the argument of **−newer** will be dereferenced, and the timestamp will be taken from the file to which the symbolic link points. The same consideration applies to **−newerXY**, **−anewer** and **−cnewer**.

The **−follow** option has a similar effect to **−L**, though it takes effect at the point where it appears (that is, if **−L** is not used but **−follow** is, any symbolic links appearing after **−follow** on the command line will be dereferenced, and those before it will not).

−D debugopts

Print diagnostic information; this can be helpful to diagnose problems with why **find** is not doing what you want. The list of debug options should be comma separated. Compatibility of the debug options is not guaranteed between releases of findutils. For a complete list of valid debug options, see the output of **find −D help**. Valid debug options include

exec Show diagnostic information relating to −exec, −execdir, −ok and −okdir

help Explain the debugging options.

opt Prints diagnostic information relating to the optimisation of the expression tree; see the −O option.

rates Prints a summary indicating how often each predicate succeeded or failed.

search Navigate the directory tree verbosely.

stat Print messages as files are examined with the **stat** and **lstat** system calls. The **find** program tries to minimise such calls.

tree Show the expression tree in its original and optimised form.

−Olevel

Enables query optimisation. The **find** program reorders tests to speed up execution while preserving the overall effect; that is, predicates with side effects are not reordered relative to each other. The optimisations performed at each optimisation level are as follows.

0 Equivalent to optimisation level 1.

1 This is the default optimisation level and corresponds to the traditional behaviour. Expressions are reordered so that tests based only on the names of files (for example **−name** and **−regex**) are performed first.

2 Any **−type** or **−xtype** tests are performed after any tests based only on the names of files, but before any tests that require information from the inode. On many modern versions of Unix, file types are returned by **readdir()** and so these predicates are faster to evaluate than predicates which need to stat the file first. If you use the **−fstype** *FOO* predicate and specify a filesystem type *FOO* which is not known (that is, present in '/etc/mtab') at the time **find** starts, that predicate is equivalent to **−false**.

3 At this optimisation level, the full cost-based query optimiser is enabled. The order of tests is modified so that cheap (i.e. fast) tests are performed first and more expensive ones are performed later, if necessary. Within each cost band, predicates are evaluated earlier or later according to whether they are likely to succeed or not. For **−o**, predicates which are likely to succeed are evaluated earlier, and for **−a**, predicates which are likely to fail are evaluated earlier.

The cost-based optimiser has a fixed idea of how likely any given test is to succeed. In some cases the probability takes account of the specific nature of the test (for example, **−type f** is assumed to be more likely to succeed than **−type c**). The cost-based optimiser is currently being evaluated. If it does not actually improve the performance of **find**, it will be removed again. Conversely, optimisations that prove to be reliable, robust and effective may be enabled at lower optimisation levels over time. However, the default behaviour (i.e. optimisation level 1) will not be changed in the 4.3.x release series. The findutils test suite runs all the tests on **find** at each optimisation level and ensures that the result is the same.

EXPRESSION

The part of the command line after the list of starting points is the *expression*. This is a kind of query specification describing how we match files and what we do with the files that were matched. An expression is composed of a sequence of things:

Tests Tests return a true or false value, usually on the basis of some property of a file we are considering. The **−empty** test for example is true only when the current file is empty.

Actions

Actions have side effects (such as printing something on the standard output) and return either true or false, usually based on whether or not they are successful. The **−print** action for example prints the name of the current file on the standard output.

Global options

Global options affect the operation of tests and actions specified on any part of the command line. Global options always return true. The **−depth** option for example makes **find** traverse the file system in a depth-first order.

Positional options

Positional options affect only tests or actions which follow them. Positional options always return true. The **−regextype** option for example is positional, specifying the regular expression dialect for regular expressions occurring later on the command line.

Operators

Operators join together the other items within the expression. They include for example **−o** (meaning logical OR) and **−a** (meaning logical AND). Where an operator is missing, **−a** is assumed.

If the whole expression contains no actions other than **−prune** or **−print**, **−print** is performed on all files for which the whole expression is true.

The **−delete** action also acts like an option (since it implies **−depth**).

POSITIONAL OPTIONS

Positional options always return true. They affect only tests occurring later on the command line.

−daystart
Measure times (for **−amin**, **−atime**, **−cmin**, **−ctime**, **−mmin**, and **−mtime**) from the beginning of today rather than from 24 hours ago. This option only affects tests which appear later on the command line.

−follow
Deprecated; use the **−L** option instead. Dereference symbolic links. Implies **−noleaf**. The **−follow** option affects only those tests which appear after it on the command line. Unless the **−H** or **−L** option has been specified, the position of the **−follow** option changes the behaviour of the **−newer** predicate; any files listed as the argument of **−newer** will be dereferenced if they are symbolic links. The same consideration applies to **−newerXY**, **−anewer** and **−cnewer**. Similarly, the **−type** predicate will always match against the type of the file that a symbolic link points to rather than the link itself. Using **−follow** causes the **−lname and −ilname** predicates always to return false.

−regextype *type*
Changes the regular expression syntax understood by **−regex** and **−iregex** tests which occur later on the command line. To see which regular expression types are known, use **-regextype help**. The Texinfo documentation (see **SEE ALSO**) explains the meaning of and differences between the various types of regular expression.

−warn, −nowarn
Turn warning messages on or off. These warnings apply only to the command line usage, not to any conditions that **find** might encounter when it searches directories. The default behaviour corresponds to **−warn** if standard input is a tty, and to **−nowarn** otherwise. If a warning message relating to command-line usage is produced, the exit status of **find** is not affected. If the POSIXLY_CORRECT environment variable is set, and **−warn is also used, it is not specified which, if any, warnings will be active.**

GLOBAL OPTIONS

Global options always return true. Global options take effect even for tests which occur earlier on the command line. To prevent confusion, global options should specified on the command-line after the list of start points, just before the first test, positional option or action. If you specify a global option in some other place, **find** will issue a warning message explaining that this can be confusing.

The global options occur after the list of start points, and so are not the same kind of option as **−L**, for example.

−d A synonym for −depth, for compatibility with FreeBSD, NetBSD, MacOS X and OpenBSD.

−depth Process each directory's contents before the directory itself. The −delete action also implies **−depth**.

−help, −−help
Print a summary of the command-line usage of **find** and exit.

−ignore_readdir_race
Normally, **find** will emit an error message when it fails to stat a file. If you give this option and a file is deleted between the time **find** reads the name of the file from the directory and the time it tries to stat the file, no error message will be issued. This also applies to files or directories whose names are given on the command line. This option takes effect at the time the command

line is read, which means that you cannot search one part of the filesystem with this option on and part of it with this option off (if you need to do that, you will need to issue two **find** commands instead, one with the option and one without it).

−maxdepth *levels*
Descend at most *levels* (a non-negative integer) levels of directories below the starting-points. **−maxdepth 0**
means only apply the tests and actions to the starting-points themselves.

−mindepth *levels*
Do not apply any tests or actions at levels less than *levels* (a non-negative integer). **−mindepth 1** means process all files except the starting-points.

−mount
Don't descend directories on other filesystems. An alternate name for **−xdev**, for compatibility with some other versions of **find**.

−noignore_readdir_race
Turns off the effect of **−ignore_readdir_race**.

−noleaf Do not optimize by assuming that directories contain 2 fewer subdirectories than their hard link count. This option is needed when searching filesystems that do not follow the Unix directory-link convention, such as CD-ROM or MS-DOS filesystems or AFS volume mount points. Each directory on a normal Unix filesystem has at least 2 hard links: its name and its '.' entry. Additionally, its subdirectories (if any) each have a '..' entry linked to that directory. When **find** is examining a directory, after it has statted 2 fewer subdirectories than the directory's link count, it knows that the rest of the entries in the directory are non-directories ('leaf' files in the directory tree). If only the files' names need to be examined, there is no need to stat them; this gives a significant increase in search speed.

−version, −−version
Print the **find** version number and exit.

−xdev Don't descend directories on other filesystems.

TESTS

Some tests, for example **−newerXY** and **-samefile**, allow comparison between the file currently being examined and some reference file specified on the command line. When these tests are used, the interpretation of the reference file is determined by the options **−H**, **−L** and **−P** and any previous **−follow**, but the reference file is only examined once, at the time the command line is parsed. If the reference file cannot be examined (for example, the **stat**(2) system call fails for it), an error message is issued, and **find** exits with a nonzero status.

Numeric arguments can be specified as

+*n* for greater than *n*,

−*n* for less than *n*,

n for exactly *n*.

−amin *n*
File was last accessed *n* minutes ago.

−anewer *file*
File was last accessed more recently than *file* was modified. If *file* is a symbolic link and the **−H** option or the **−L** option is in effect, the access time of the file it points to is always used.

−atime *n*
File was last accessed *n**24 hours ago. When find figures out how many 24-hour periods ago the file was last accessed, any fractional part is ignored, so to match **−atime +1**, a file has to have been accessed at least *two* days ago.

−cmin *n*
File's status was last changed *n* minutes ago.

−cnewer *file*
File's status was last changed more recently than *file* was modified. If *file* is a symbolic link and the **−H** option or the **−L** option is in effect, the status-change time of the file it points to is always used.

−ctime *n*
File's status was last changed *n**24 hours ago. See the comments for **−atime** to understand how rounding affects the interpretation of file status change times.

−empty File is empty and is either a regular file or a directory.

−executable
Matches files which are executable and directories which are searchable (in a file name resolution sense). This takes into account access control lists and other permissions artefacts which the **−perm** test ignores. This test makes use of the **access**(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access**(2) in the client's kernel and so cannot make use of the UID mapping information held on the server. Because this test is based only on the result of the **access**(2) system call, there is no guarantee that a file for which this test succeeds can actually be executed.

−false Always false.

−fstype *type*
File is on a filesystem of type *type*. The valid filesystem types vary among different versions of Unix; an incomplete list of filesystem types that are accepted on some version of Unix or another is: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. You can use **−printf** with the %F directive to see the types of your filesystems.

−gid *n* File's numeric group ID is *n*.

−group *gname*
File belongs to group *gname* (numeric group ID allowed).

−ilname *pattern*
Like **−lname**, but the match is case insensitive. If the **−L** option or the **−follow** option is in effect, this test returns false unless the symbolic link is broken.

−iname *pattern*
Like **−name**, but the match is case insensitive. For example, the patterns ‘fo*’ and ‘F??’ match the file names ‘Foo’, ‘FOO’, ‘foo’, ‘fOo’, etc. The pattern ‘*foo*‘ will also match a file called ’.foobar’.

−inum *n*
File has inode number *n*. It is normally easier to use the **−samefile** test instead.

−ipath *pattern*
Like **−path**. but the match is case insensitive.

−iregex *pattern*
Like **−regex**, but the match is case insensitive.

−iwholename *pattern*
See −ipath. This alternative is less portable than **−ipath**.

−links *n*
File has *n* links.

−lname *pattern*
File is a symbolic link whose contents match shell pattern *pattern*. The metacharacters do not treat ‘/’ or ‘.’ specially. If the **−L** option or the **−follow** option is in effect, this test returns false unless the symbolic link is broken.

−mmin *n*
File’s data was last modified *n* minutes ago.

−mtime *n*
File’s data was last modified *n**24 hours ago. See the comments for **−atime** to understand how rounding affects the interpretation of file modification times.

−name *pattern*
Base of file name (the path with the leading directories removed) matches shell pattern *pattern*. Because the leading directories are removed, the file names considered for a match with **−name** will never include a slash, so ‘−name a/b’ will never match anything (you probably need to use **−path** instead). A warning is issued if you try to do this, unless the environment variable POSIXLY_CORRECT is set. The metacharacters (‘*’, ‘?’, and ‘[]’) match a ‘.’ at the start of the base name (this is a change in findutils-4.2.2; see section STANDARDS CONFORMANCE below). To ignore a directory and the files under it, use **−prune**; see an example in the description of **−path**. Braces are not recognised as being special, despite the fact that some shells including Bash imbue braces with a special meaning in shell patterns. The filename matching is performed with the use of the **fnmatch**(3) library function. Don’t forget to enclose the pattern in quotes in order to protect it from expansion by the shell.

−newer *file*
File was modified more recently than *file*. If *file* is a symbolic link and the **−H** option or the **−L** option is in effect, the modification time of the file it points to is always used.

−newerXY *reference*

Succeeds if timestamp *X* of the file being considered is newer than timestamp *Y* of the file *reference*. The letters *X* and *Y* can be any of the following letters:

a The access time of the file *reference*
B The birth time of the file *reference*
c The inode status change time of *reference*
m The modification time of the file *reference*
t *reference* is interpreted directly as a time

Some combinations are invalid; for example, it is invalid for *X* to be *t*. Some combinations are not implemented on all systems; for example *B* is not supported on all systems. If an invalid or unsupported combination of *XY* is specified, a fatal error results. Time specifications are interpreted as for the argument to the **−d** option of GNU **date**. If you try to use the birth time of a reference file, and the birth time cannot be determined, a fatal error message results. If you specify a test which refers to the birth time of files being examined, this test will fail for any files where the birth time is unknown.

−nogroup

No group corresponds to file's numeric group ID.

−nouser

No user corresponds to file's numeric user ID.

−path *pattern*

File name matches shell pattern *pattern*. The metacharacters do not treat '/' or '.' specially; so, for example,

find . −path "./sr*sc"

will print an entry for a directory called './src/misc' (if one exists). To ignore a whole directory tree, use **−prune** rather than checking every file in the tree. For example, to skip the directory 'src/emacs' and all files and directories under it, and print the names of the other files found, do something like this:

find . −path ./src/emacs −prune −o −print

Note that the pattern match test applies to the whole file name, starting from one of the start points named on the command line. It would only make sense to use an absolute path name here if the relevant start point is also an absolute path. This means that this command will never match anything:

find bar −path /foo/bar/myfile −print

Find compares the **−path** argument with the concatenation of a directory name and the base name of the file it's examining. Since the concatenation will never end with a slash, **−path** arguments ending in a slash will match nothing (except perhaps a start point specified on the command line). The predicate **−path** is also supported by HP-UX **find** and is part of the POSIX 2008 standard.

−perm *mode*

File's permission bits are exactly *mode* (octal or symbolic). Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex mode string. For example '−perm g=w' will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the '/' or '-' forms, for example '−perm −g=w', which matches any file with group write permission. See the **EXAMPLES** section for some illustrative examples.

−perm −*mode*
All of the permission bits *mode* are set for the file. Symbolic modes are accepted in this form, and this is usually the way in which you would want to use them. You must specify ‘u’, ‘g’ or ‘o’ if you use a symbolic mode. See the **EXAMPLES** section for some illustrative examples.

−perm /*mode*
Any of the permission bits *mode* are set for the file. Symbolic modes are accepted in this form. You must specify ‘u’, ‘g’ or ‘o’ if you use a symbolic mode. See the **EXAMPLES** section for some illustrative examples. If no permission bits in *mode* are set, this test matches any file (the idea here is to be consistent with the behaviour of **−perm −000**).

−perm +*mode*
This is no longer supported (and has been deprecated since 2005). Use **−perm /***mode* instead.

−readable
Matches files which are readable. This takes into account access control lists and other permissions artefacts which the **−perm** test ignores. This test makes use of the **access**(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access**(2) in the client’s kernel and so cannot make use of the UID mapping information held on the server.

−regex *pattern*
File name matches regular expression *pattern*. This is a match on the whole path, not a search. For example, to match a file named ‘./fubar3’, you can use the regular expression ‘.*bar.’ or ‘.*b.*3’, but not ‘f.*r3’. The regular expressions understood by **find** are by default Emacs Regular Expressions, but this can be changed with the **−regextype** option.

−samefile *name*
File refers to the same inode as *name*. When **−L** is in effect, this can include symbolic links.

−size *n*[cwbkMG]
File uses *n* units of space, rounding up. The following suffixes can be used:

‘b’ for 512-byte blocks (this is the default if no suffix is used)

‘c’ for bytes

‘w’ for two-byte words

‘k’ for Kilobytes (units of 1024 bytes)

‘M’ for Megabytes (units of 1048576 bytes)

‘G’ for Gigabytes (units of 1073741824 bytes)

The size does not count indirect blocks, but it does count blocks in sparse files that are not actually allocated. Bear in mind that the ‘%k’ and ‘%b’ format specifiers of **−printf** handle sparse files differently. The ‘b’ suffix always denotes 512-byte blocks and never 1 Kilobyte blocks, which is different to the behaviour of **−ls**.

The + and - prefixes signify greater than and less than, as usual. Bear in mind that the size is rounded up to the next unit. Therefore **−size −1M** is not equivalent to **−size −1048576c**. The former only matches empty files, the latter matches files from 1 to 1,048,575 bytes.

−true Always true.

−type *c* File is of type *c*:

b block (buffered) special

c character (unbuffered) special

d directory

p named pipe (FIFO)

f regular file

l symbolic link; this is never true if the **−L** option or the **−follow** option is in effect, unless the symbolic link is broken. If you want to search for symbolic links when **−L** is in effect, use **−xtype**.

s socket

D door (Solaris)

To search for more than one type at once, you can supply the combined list of type letters separated by a comma ‘,’ (GNU extension).

−uid *n* File’s numeric user ID is *n*.

−used *n*
File was last accessed *n* days after its status was last changed.

−user *uname*
File is owned by user *uname* (numeric user ID allowed).

−wholename *pattern*
See −path. This alternative is less portable than **−path**.

−writable
Matches files which are writable. This takes into account access control lists and other permissions artefacts which the **−perm** test ignores. This test makes use of the **access**(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access**(2) in the client’s kernel and so cannot make use of the UID mapping information held on the server.

−xtype *c*
The same as **−type** unless the file is a symbolic link. For symbolic links: if the **−H** or **−P** option was specified, true if the file is a link to a file of type *c*; if the **−L** option has been given, true if *c* is ‘l’. In other words, for symbolic links, **−xtype** checks the type of the file that **−type** does not check.

−context *pattern*
(SELinux only) Security context of the file matches glob *pattern*.

ACTIONS

−delete Delete files; true if removal succeeded. If the removal failed, an error message is issued. If **−delete** fails, **find**’s exit status will be nonzero (when it eventually exits). Use of **−delete** automatically turns on the ‘**−depth**’ option.

Warnings: Don’t forget that the find command line is evaluated as an expression, so putting **−delete** first will make **find** try to delete everything below the starting points you specified. When testing a **find** command line that you later intend to use with **−delete**, you should explicitly specify **−depth** in order to avoid later surprises. Because **−delete** implies **−depth**, you cannot usefully use

−prune and **−delete** together.

−exec *command* ;
Execute *command*; true if 0 status is returned. All following arguments to **find** are taken to be arguments to the command until an argument consisting of ‘;’ is encountered. The string ‘{}’ is replaced by the current file name being processed everywhere it occurs in the arguments to the command, not just in arguments where it is alone, as in some versions of **find**. Both of these constructions might need to be escaped (with a ‘\’) or quoted to protect them from expansion by the shell. See the **EXAMPLES** section for examples of the use of the **−exec** option. The specified command is run once for each matched file. The command is executed in the starting directory. There are unavoidable security problems surrounding use of the **−exec** action; you should use the **−execdir** option instead.

−exec *command* {} +
This variant of the **−exec** action runs the specified command on the selected files, but the command line is built by appending each selected file name at the end; the total number of invocations of the command will be much less than the number of matched files. The command line is built in much the same way that **xargs** builds its command lines. Only one instance of ‘{}’ is allowed within the command, and (when **find** is being invoked from a shell) it should be quoted (for example, '{}') to protect it from interpretation by shells. The command is executed in the starting directory. If any invocation returns a non-zero value as exit status, then **find** returns a non-zero exit status. If **find** encounters an error, this can sometimes cause an immediate exit, so some pending commands may not be run at all. This variant of **−exec** always returns true.

−execdir *command* ;

−execdir *command* {} +
Like **−exec**, but the specified command is run from the subdirectory containing the matched file, which is not normally the directory in which you started **find**. As with −exec, the {} should be quoted if find is being invoked from a shell. This a much more secure method for invoking commands, as it avoids race conditions during resolution of the paths to the matched files. As with the **−exec** action, the ‘+’ form of **−execdir** will build a command line to process more than one matched file, but any given invocation of *command* will only list files that exist in the same subdirectory. If you use this option, you must ensure that your **$PATH** environment variable does not reference ‘.’; otherwise, an attacker can run any commands they like by leaving an appropriately-named file in a directory in which you will run **−execdir**. The same applies to having entries in **$PATH** which are empty or which are not absolute directory names. If any invocation returns a non-zero value as exit status, then **find** returns a non-zero exit status. If **find** encounters an error, this can sometimes cause an immediate exit, so some pending commands may not be run at all. The result of the action depends on whether the **+** or the **;** variant is being used; **−execdir** *command* **{} +** always returns true, while **−execdir** *command* **{} ;** returns true only if *command* returns 0.

−fls *file* True; like **−ls** but write to *file* like **−fprint**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

−fprint *file*
True; print the full file name into file *file*. If *file* does not exist when **find** is run, it is created; if it does exist, it is truncated. The file names ‘/dev/stdout’ and ‘/dev/stderr’ are handled specially; they refer to the standard output and standard error output, respectively. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for

information about how unusual characters in filenames are handled.

−fprint0 *file*
True; like **−print0** but write to *file* like **−fprint**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

−fprintf *file format*
True; like **−printf** but write to *file* like **−fprint**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

−ls True; list current file in **ls −dils** format on standard output. The block counts are of 1K blocks, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte blocks are used. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

−ok *command* ;
Like **−exec** but ask the user first. If the user agrees, run the command. Otherwise just return false. If the command is run, its standard input is redirected from **/dev/null**.

The response to the prompt is matched against a pair of regular expressions to determine if it is an affirmative or negative response. This regular expression is obtained from the system if the 'POSIXLY_CORRECT' environment variable is set, or otherwise from **find**'s message translations. If the system has no suitable definition, **find**'s own definition will be used. In either case, the interpretation of the regular expression itself will be affected by the environment variables 'LC_CTYPE' (character classes) and 'LC_COLLATE' (character ranges and equivalence classes).

−okdir *command* ;
Like **−execdir** but ask the user first in the same way as for **−ok**. If the user does not agree, just return false. If the command is run, its standard input is redirected from **/dev/null**.

−print True; print the full file name on the standard output, followed by a newline. If you are piping the output of **find** into another program and there is the faintest possibility that the files which you are searching for might contain a newline, then you should seriously consider using the **−print0** option instead of **−print**. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

−print0 True; print the full file name on the standard output, followed by a null character (instead of the newline character that **−print** uses). This allows file names that contain newlines or other types of white space to be correctly interpreted by programs that process the **find** output. This option corresponds to the **−0** option of **xargs**.

−printf *format*
True; print *format* on the standard output, interpreting '\' escapes and '%' directives. Field widths and precisions can be specified as with the 'printf' C function. Please note that many of the fields are printed as %s rather than %d, and this may mean that flags don't work as you might expect. This also means that the '−' flag does work (it forces fields to be left-aligned). Unlike **−print**, **−printf** does not add a newline at the end of the string. The escapes and directives are:

\a Alarm bell.

\b Backspace.

\c Stop printing from this format immediately and flush the output.

\f Form feed.

\n Newline.

\r Carriage return.

\t Horizontal tab.

\v Vertical tab.

\0 ASCII NUL.

\\ A literal backslash (‘\’).

\NNN The character whose ASCII code is NNN (octal).

A ‘\’ character followed by any other character is treated as an ordinary character, so they both are printed.

%% A literal percent sign.

%a File’s last access time in the format returned by the C ‘ctime’ function.

%A*k* File’s last access time in the format specified by *k*, which is either ‘@’ or a directive for the C ‘strftime’ function. The possible values for *k* are listed below; some of them might not be available on all systems, due to differences in ‘strftime’ between systems.

@ seconds since Jan. 1, 1970, 00:00 GMT, with fractional part.

Time fields:

H hour (00..23)

I hour (01..12)

k hour (0..23)

l hour (1..12)

M minute (00..59)

p locale’s AM or PM

r time, 12-hour (hh:mm:ss [AP]M)

S Second (00.00 .. 61.00). There is a fractional part.

T time, 24-hour (hh:mm:ss.xxxxxxxxxx)

\+ Date and time, separated by ‘+’, for example ‘2004−04−28+22:22:05.0’. This is a GNU extension. The time is given in the current timezone (which may be affected by setting the TZ environment variable). The seconds field includes a fractional part.

X locale’s time representation (H:M:S). The seconds field includes a fractional part.

Z time zone (e.g., EDT), or nothing if no time zone is determinable

Date fields:

a locale’s abbreviated weekday name (Sun..Sat)

A locale’s full weekday name, variable length (Sunday..Saturday)

b locale’s abbreviated month name (Jan..Dec)

B locale's full month name, variable length (January..December)

c locale's date and time (Sat Nov 04 12:02:33 EST 1989). The format is the same as for **ctime**(3) and so to preserve compatibility with that format, there is no fractional part in the seconds field.

d day of month (01..31)

D date (mm/dd/yy)

h same as b

j day of year (001..366)

m month (01..12)

U week number of year with Sunday as first day of week (00..53)

w day of week (0..6)

W week number of year with Monday as first day of week (00..53)

x locale's date representation (mm/dd/yy)

y last two digits of year (00..99)

Y year (1970...)

%b The amount of disk space used for this file in 512-byte blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/512, but it can also be smaller if the file is a sparse file.

%c File's last status change time in the format returned by the C 'ctime' function.

%C*k* File's last status change time in the format specified by *k*, which is the same as for %A.

%d File's depth in the directory tree; 0 means the file is a starting-point.

%D The device number on which the file exists (the st_dev field of struct stat), in decimal.

%f File's name with any leading directories removed (only the last element).

%F Type of the filesystem the file is on; this value can be used for −fstype.

%g File's group name, or numeric group ID if the group has no name.

%G File's numeric group ID.

%h Leading directories of file's name (all but the last element). If the file name contains no slashes (since it is in the current directory) the %h specifier expands to '.'.

%H Starting-point under which file was found.

%i File's inode number (in decimal).

%k The amount of disk space used for this file in 1K blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/1024, but it can also be smaller if the file is a sparse file.

%l Object of symbolic link (empty string if file is not a symbolic link).

%m File's permission bits (in octal). This option uses the 'traditional' numbers which most Unix implementations use, but if your particular implementation uses an unusual ordering of octal permissions bits, you will see a difference between the actual value of the file's mode and the output of %m. Normally you will want to have a leading zero on this number, and to do this, you should use the # flag (as in, for example, '%#m').

%M File's permissions (in symbolic form, as for **ls**). This directive is supported in findutils 4.2.5 and later.

%n Number of hard links to file.

%p File's name.

%P File's name with the name of the starting-point under which it was found removed.

%s File's size in bytes.

%S File's sparseness. This is calculated as (BLOCKSIZE*st_blocks / st_size). The exact value you will get for an ordinary file of a certain length is system-dependent. However, normally sparse files will have values less than 1.0, and files which use indirect blocks may have a value which is greater than 1.0. The value used for BLOCKSIZE is system-dependent, but is usually 512 bytes. If the file size is zero, the value printed is undefined. On systems which lack support for st_blocks, a file's sparseness is assumed to be 1.0.

%t File's last modification time in the format returned by the C 'ctime' function.

%T*k* File's last modification time in the format specified by *k*, which is the same as for %A.

%u File's user name, or numeric user ID if the user has no name.

%U File's numeric user ID.

%y File's type (like in **ls −l**), U=unknown type (shouldn't happen)

%Y File's type (like %y), plus follow symlinks: L=loop, N=nonexistent

%Z (SELinux only) file's security context.

%{ %[%(
Reserved for future use.

A '%' character followed by any other character is discarded, but the other character is printed (don't rely on this, as further format characters may be introduced). A '%' at the end of the format argument causes undefined behaviour since there is no following character. In some locales, it may hide your door keys, while in others it may remove the final page from the novel you are reading.

The %m and %d directives support the **#** , **0** and **+** flags, but the other directives do not, even if they print numbers. Numeric directives that do not support these flags include **G**, **U**, **b**, **D**, **k** and **n**. The '−' format flag is supported and changes the alignment of a field from right-justified (which is the default) to left-justified.

See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

−prune True; if the file is a directory, do not descend into it. If **−depth** is given, false; no effect. Because **−delete** implies **−depth**, you cannot usefully use **−prune** and **−delete** together.

−quit Exit immediately. No child processes will be left running, but no more paths specified on the command line will be processed. For example, **find /tmp/foo /tmp/bar −print −quit** will print only **/tmp/foo**. Any command lines which have been built up with **−execdir ... {} +** will be invoked before **find** exits. The exit status may or may not be zero, depending on whether an error has already occurred.

OPERATORS

Listed in order of decreasing precedence:

(*expr*) Force precedence. Since parentheses are special to the shell, you will normally need to quote them. Many of the examples in this manual page use backslashes for this purpose: '\(...\)' instead of '(...)'.

! *expr* True if *expr* is false. This character will also usually need protection from interpretation by the shell.

−not *expr*
Same as ! *expr*, but not POSIX compliant.

expr1 expr2
Two expressions in a row are taken to be joined with an implied **−a**; *expr2* is not evaluated if *expr1* is false.

expr1 −a *expr2*
Same as *expr1 expr2*.

expr1 −and *expr2*
Same as *expr1 expr2*, but not POSIX compliant.

expr1 −o *expr2*
Or; *expr2* is not evaluated if *expr1* is true.

expr1 −or *expr2*
Same as *expr1* **−o** *expr2*, but not POSIX compliant.

expr1 , *expr2*
List; both *expr1* and *expr2* are always evaluated. The value of *expr1* is discarded; the value of the list is the value of *expr2*. The comma operator can be useful for searching for several different types of thing, but traversing the filesystem hierarchy only once. The **−fprintf** action can be used to list the various matched items into several different output files.

Please note that **−a** when specified implicitly (for example by two tests appearing without an explicit operator between them) or explicitly has higher precedence than **−o**. This means that **find . −name afile −o −name bfile −print** will never print *afile*.

UNUSUAL FILENAMES

Many of the actions of **find** result in the printing of data which is under the control of other users. This includes file names, sizes, modification times and so forth. File names are a potential problem since they can contain any character except '\0' and '/'. Unusual characters in file names can do unexpected and often undesirable things to your terminal (for example, changing the settings of your function keys on some terminals). Unusual characters are handled differently by various actions, as described below.

−print0, −fprint0
Always print the exact filename, unchanged, even if the output is going to a terminal.

−ls, −fls
Unusual characters are always escaped. White space, backslash, and double quote characters are printed using C-style escaping (for example '\f', '\"'). Other unusual characters are printed using an octal escape. Other printable characters (for **−ls** and **−fls** these are the characters between octal 041 and 0176) are printed as-is.

−printf, −fprintf
If the output is not going to a terminal, it is printed as-is. Otherwise, the result depends on which directive is in use. The directives %D, %F, %g, %G, %H, %Y, and %y expand to values which are not under control of files' owners, and so are printed as-is. The directives %a, %b, %c, %d, %i, %k, %m, %M, %n, %s, %t, %u and %U have values which are under the control of files' owners but which cannot be used to send arbitrary data to the terminal, and so these are printed as-is. The directives %f, %h, %l, %p and %P are quoted. This quoting is performed in the same way as for GNU **ls**. This is not the same quoting mechanism as the one used for **−ls** and **−fls**. If you are able to decide what format to use for the output of **find** then it is normally better to use '\0' as a terminator than to use newline, as file names can contain white space and newline characters. The setting of the 'LC_CTYPE' environment variable is used to determine which characters need to be quoted.

−print, −fprint
Quoting is handled in the same way as for **−printf** and **−fprintf**. If you are using **find** in a script or in a situation where the matched files might have arbitrary names, you should consider using **−print0** instead of **−print**.

The **−ok** and **−okdir** actions print the current filename as-is. This may change in a future release.

STANDARDS CONFORMANCE

For closest compliance to the POSIX standard, you should set the POSIXLY_CORRECT environment variable. The following options are specified in the POSIX standard (IEEE Std 1003.1, 2003 Edition):

−H This option is supported.

−L This option is supported.

−name This option is supported, but POSIX conformance depends on the POSIX conformance of the system's **fnmatch**(3) library function. As of findutils-4.2.2, shell metacharacters ('*', '?' or '[]' for example) will match a leading '.', because IEEE PASC interpretation 126 requires this. This is a change from previous versions of findutils.

−type Supported. POSIX specifies 'b', 'c', 'd', 'l', 'p', 'f' and 's'. GNU find also supports 'D', representing a Door, where the OS provides these. Furthermore, GNU find allows multiple types to be specified at once in a comma-separated list.

−ok Supported. Interpretation of the response is according to the 'yes' and 'no' patterns selected by setting the 'LC_MESSAGES' environment variable. When the 'POSIXLY_CORRECT' environment variable is set, these patterns are taken system's definition of a positive (yes) or negative (no) response. See the system's documentation for **nl_langinfo**(3), in particular YESEXPR and NOEXPR. When 'POSIXLY_CORRECT' is not set, the patterns are instead taken from **find**'s own message catalogue.

−newer
Supported. If the file specified is a symbolic link, it is always dereferenced. This is a change from previous behaviour, which used to take the relevant time from the symbolic link; see the HISTORY section below.

−perm Supported. If the POSIXLY_CORRECT environment variable is not set, some mode arguments (for example +a+x) which are not valid in POSIX are supported for backward-compatibility.

Other predicates

The predicates **−atime**, **−ctime**, **−depth**, **−group**, **−links**, **−mtime**, **−nogroup**, **−nouser**, **−print**, **−prune**, **−size**, **−user** and **−xdev** ‘−atime’, ‘−ctime’, ‘−depth’, ‘−group’, ‘−links’, ‘−mtime’, ‘−nogroup’, ‘−nouser’, ‘−perm’, ‘−print’, ‘−prune’, ‘−size’, ‘−user’ and ‘−xdev’, are all supported.

The POSIX standard specifies parentheses ‘(’, ‘)’, negation ‘!’ and the ‘and’ and ‘or’ operators (**−a**, **−o**).

All other options, predicates, expressions and so forth are extensions beyond the POSIX standard. Many of these extensions are not unique to GNU find, however.

The POSIX standard requires that **find** detects loops:

> The **find** utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, find shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

GNU **find** complies with these requirements. The link count of directories which contain entries which are hard links to an ancestor will often be lower than they otherwise should be. This can mean that GNU find will sometimes optimise away the visiting of a subdirectory which is actually a link to an ancestor. Since **find** does not actually enter such a subdirectory, it is allowed to avoid emitting a diagnostic message. Although this behaviour may be somewhat confusing, it is unlikely that anybody actually depends on this behaviour. If the leaf optimisation has been turned off with **−noleaf**, the directory entry will always be examined and the diagnostic message will be issued where it is appropriate. Symbolic links cannot be used to create filesystem cycles as such, but if the **−L** option or the **−follow** option is in use, a diagnostic message is issued when **find** encounters a loop of symbolic links. As with loops containing hard links, the leaf optimisation will often mean that **find** knows that it doesn’t need to call *stat()* or *chdir()* on the symbolic link, so this diagnostic is frequently not necessary.

The **−d** option is supported for compatibility with various BSD systems, but you should use the POSIX-compliant option **−depth** instead.

The POSIXLY_CORRECT environment variable does not affect the behaviour of the **−regex** or **−iregex** tests because those tests aren’t specified in the POSIX standard.

ENVIRONMENT VARIABLES

LANG Provides a default value for the internationalization variables that are unset or null.

LC_ALL

If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE

The POSIX standard specifies that this variable affects the pattern matching to be used for the **−name** option. GNU find uses the **fnmatch**(3) library function, and so support for ‘LC_COLLATE’ depends on the system library. This variable also affects the interpretation of the response to **−ok;** while the ‘LC_MESSAGES’ variable selects the actual pattern used to interpret the response to **−ok**, the interpretation of any bracket expressions in the pattern will be affected by ‘LC_COLLATE’.

LC_CTYPE

This variable affects the treatment of character classes used in regular expressions and also with the **−name** test, if the system’s **fnmatch**(3) library function supports this. This variable also affects the interpretation of any character classes in the regular expressions used to interpret the response to the prompt issued by **−ok**. The ‘LC_CTYPE’ environment variable will also affect which characters are considered to be unprintable when filenames are printed; see the section UNUSUAL FILENAMES.

LC_MESSAGES
Determines the locale to be used for internationalised messages. If the 'POSIXLY_CORRECT' environment variable is set, this also determines the interpretation of the response to the prompt made by the **−ok** action.

NLSPATH
Determines the location of the internationalisation message catalogues.

PATH Affects the directories which are searched to find the executables invoked by **−exec**, **−execdir**, **−ok** and **−okdir**.

POSIXLY_CORRECT
Determines the block size used by **−ls** and **−fls**. If **POSIXLY_CORRECT** is set, blocks are units of 512 bytes. Otherwise they are units of 1024 bytes.

Setting this variable also turns off warning messages (that is, implies **−nowarn**) by default, because POSIX requires that apart from the output for **−ok**, all messages printed on stderr are diagnostics and must result in a non-zero exit status.

When POSIXLY_CORRECT is not set, **−perm** +zzz is treated just like **−perm** /zzz if +zzz is not a valid symbolic mode. When POSIXLY_CORRECT is set, such constructs are treated as an error.

When POSIXLY_CORRECT is set, the response to the prompt made by the **−ok** action is interpreted according to the system's message catalogue, as opposed to according to **find**'s own message translations.

TZ Affects the time zone used for some of the time-related format directives of **−printf** and **−fprintf**.

EXAMPLES

find /tmp −name core −type f −print | xargs /bin/rm −f

Find files named **core** in or below the directory **/tmp** and delete them. Note that this will work incorrectly if there are any filenames containing newlines, single or double quotes, or spaces.

find /tmp −name core −type f −print0 | xargs −0 /bin/rm −f

Find files named **core** in or below the directory **/tmp** and delete them, processing filenames in such a way that file or directory names containing single or double quotes, spaces or newlines are correctly handled. The **−name** test comes before the **−type** test in order to avoid having to call **stat(2)** on every file.

find . −type f −exec file '{}' \;

Runs 'file' on every file in or below the current directory. Notice that the braces are enclosed in single quote marks to protect them from interpretation as shell script punctuation. The semicolon is similarly protected by the use of a backslash, though single quotes could have been used in that case also.

**find / \(−perm −4000 −fprintf /root/suid.txt '%#m %u %p\n' \) , **
\(−size +100M −fprintf /root/big.txt '%−10s %p\n' \)

Traverse the filesystem just once, listing setuid files and directories into **/root/suid.txt** and large files into **/root/big.txt**.

find $HOME −mtime 0

Search for files in your home directory which have been modified in the last twenty-four hours. This

command works this way because the time since each file was last modified is divided by 24 hours and any remainder is discarded. That means that to match **−mtime 0**, a file will have to have a modification in the past which is less than 24 hours ago.

find /sbin /usr/sbin -executable \! -readable −print

Search for files which are executable but not readable.

find . −perm 664

Search for files which have read and write permission for their owner, and group, but which other users can read but not write to. Files which meet these criteria but have other permissions bits set (for example if someone can execute the file) will not be matched.

find . −perm −664

Search for files which have read and write permission for their owner and group, and which other users can read, without regard to the presence of any extra permission bits (for example the executable bit). This will match a file which has mode 0777, for example.

find . −perm /222

Search for files which are writable by somebody (their owner, or their group, or anybody else).

find . −perm /220
find . −perm /u+w,g+w
find . −perm /u=w,g=w

All three of these commands do the same thing, but the first one uses the octal representation of the file mode, and the other two use the symbolic form. These commands all search for files which are writable by either their owner or their group. The files don't have to be writable by both the owner and group to be matched; either will do.

find . −perm −220
find . −perm −g+w,u+w

Both these commands do the same thing; search for files which are writable by both their owner and their group.

find . −perm −444 −perm /222 \! −perm /111
find . −perm −a+r −perm /a+w \! −perm /a+x

These two commands both search for files that are readable for everybody (**−perm −444** or **−perm −a+r**), have at least one write bit set (**−perm /222** or **−perm /a+w**) but are not executable for anybody (**! −perm /111** and **! −perm /a+x** respectively).

cd /source-dir
find . −name .snapshot −prune −o \(\! −name '*~' −print0 \)|
cpio −pmd0 /dest-dir

This command copies the contents of **/source-dir** to **/dest-dir**, but omits files and directories named

.snapshot (and anything in them). It also omits files or directories whose name ends in ~, but not their contents. The construct **−prune −o \(... −print0 \)** is quite common. The idea here is that the expression before **−prune** matches things which are to be pruned. However, the **−prune** action itself returns true, so the following **−o** ensures that the right hand side is evaluated only for those directories which didn't get pruned (the contents of the pruned directories are not even visited, so their contents are irrelevant). The expression on the right hand side of the **−o** is in parentheses only for clarity. It emphasises that the **−print0** action takes place only for things that didn't have **−prune** applied to them. Because the default 'and' condition between tests binds more tightly than **−o**, this is the default anyway, but the parentheses help to show what is going on.

**find repo/ \(-exec test -d '{}'/.svn \; -or **
**-exec test -d {}/.git \; -or -exec test -d {}/CVS \; \) **
-print -prune

Given the following directory of projects and their associated SCM administrative directories, perform an efficient search for the projects' roots:

repo/project1/CVS
repo/gnu/project2/.svn
repo/gnu/project3/.svn
repo/gnu/project3/src/.svn
repo/project4/.git

In this example, **−prune** prevents unnecessary descent into directories that have already been discovered (for example we do not search project3/src because we already found project3/.svn), but ensures sibling directories (project2 and project3) are found.

find /tmp -type f,d,l

Search for files, directories, and symbolic links in the directory **/tmp** passing these types as a comma-separated list (GNU extension), which is otherwise equivalent to the longer, yet more portable:

find /tmp \(-type f -o -type d -o -type l \)

EXIT STATUS

find exits with status 0 if all files are processed successfully, greater than 0 if errors occur. This is deliberately a very broad description, but if the return value is non-zero, you should not rely on the correctness of the results of **find**.

When some error occurs, **find** may stop immediately, without completing all the actions specified. For example, some starting points may not have been examined or some pending program invocations for **−exec ... {} +** or **−execdir ... {} +** may not have been performed.

SEE ALSO

locate(1), **locatedb**(5), **updatedb**(1), **xargs**(1), **chmod**(1), **fnmatch**(3), **regex**(7), **stat**(2), **lstat**(2), **ls**(1), **printf**(3), **strftime**(3), **ctime**(3)

The full documentation for **find** is maintained as a Texinfo manual. If the **info** and **find** programs are properly installed at your site, the command **info find** should give you access to the complete manual.

HISTORY

As of findutils-4.2.2, shell metacharacters (‘*’, ‘?’ or ‘[]’ for example) used in filename patterns will match a leading ‘.’, because IEEE POSIX interpretation 126 requires this.

As of findutils-4.3.3, **−perm /000** now matches all files instead of none.

Nanosecond-resolution timestamps were implemented in findutils-4.3.3.

As of findutils-4.3.11, the **−delete** action sets **find**’s exit status to a nonzero value when it fails. However, **find** will not exit immediately. Previously, **find**’s exit status was unaffected by the failure of **−delete**.

Feature	Added in	Also occurs in
−newerXY	4.3.3	BSD
−D	4.3.1	
−O	4.3.1	
−readable	4.3.0	
−writable	4.3.0	
−executable	4.3.0	
−regextype	4.2.24	
−exec ... +	4.2.12	POSIX
−execdir	4.2.12	BSD
−okdir	4.2.12	
−samefile	4.2.11	
−H	4.2.5	POSIX
−L	4.2.5	POSIX
−P	4.2.5	BSD
−delete	4.2.3	
−quit	4.2.3	
−d	4.2.3	BSD
−wholename	4.2.0	
−iwholename	4.2.0	
−ignore_readdir_race	4.2.0	
−fls	4.0	
−ilname	3.8	
−iname	3.8	
−ipath	3.8	
−iregex	3.8	

The syntax **−perm +MODE** was removed in findutils-4.5.12, in favour of **−perm /MODE**. The **+MODE** syntax had been deprecated since findutils-4.2.21 which was released in 2005.

NON-BUGS

Operator precedence surprises

The command **find . −name afile −o −name bfile −print** will never print *afile* because this is actually equivalent to **find . −name afile −o \(−name bfile −a −print \)**. Remember that the precedence of **−a** is higher than that of **−o** and when there is no operator specified between tests, **−a** is assumed.

“paths must precede expression” error message

$ find . −name *.c −print
find: paths must precede expression
Usage: find [−H] [−L] [−P] [−Olevel] [−D ... [path...] [expression]

This happens because **.c* has been expanded by the shell resulting in **find** actually receiving a command line like this:
find . −name frcode.c locate.c word_io.c −print
That command is of course not going to work. Instead of doing things this way, you should enclose the pattern in quotes or escape the wildcard:
$ find . −name '*.c' −print
$ find . −name *.c −print

BUGS

There are security problems inherent in the behaviour that the POSIX standard specifies for **find**, which therefore cannot be fixed. For example, the **−exec** action is inherently insecure, and **−execdir** should be used instead. Please see **Finding Files** for more information.

The environment variable **LC_COLLATE** has no effect on the **−ok** action.

The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findutils. The reason for this is that you will then be able to track progress in fixing the problem. Other comments about **find**(1) and about the findutils package in general can be sent to the *bug−findutils* mailing list. To join the list, send email to *bug−findutils−request@gnu.org*.

NAME

grep, egrep, fgrep, rgrep − print lines matching a pattern

SYNOPSIS

grep [*OPTIONS*] *PATTERN* [*FILE*...]
grep [*OPTIONS*] [**−e** *PATTERN*]... [**−f** *FILE*]... [*FILE*...]

DESCRIPTION

grep searches the named input *FILE*s for lines containing a match to the given *PATTERN*. If no files are specified, or if the file "**−**" is given, **grep** searches standard input. By default, **grep** prints the matching lines.

In addition, the variant programs **egrep**, **fgrep** and **rgrep** are the same as **grep −E**, **grep −F**, and **grep −r**, respectively. These variants are deprecated, but are provided for backward compatibility.

OPTIONS

Generic Program Information

−−help Output a usage message and exit.

−V, **−−version**
Output the version number of **grep** and exit.

Matcher Selection

−E, **−−extended−regexp**
Interpret *PATTERN* as an extended regular expression (ERE, see below).

−F, **−−fixed−strings**
Interpret *PATTERN* as a list of fixed strings (instead of regular expressions), separated by newlines, any of which is to be matched.

−G, **−−basic−regexp**
Interpret *PATTERN* as a basic regular expression (BRE, see below). This is the default.

−P, **−−perl−regexp**
Interpret the pattern as a Perl-compatible regular expression (PCRE). This is highly experimental and **grep −P** may warn of unimplemented features.

Matching Control

−e *PATTERN*, **−−regexp=***PATTERN*
Use *PATTERN* as the pattern. If this option is used multiple times or is combined with the **−f** (**−−file**) option, search for all patterns given. This option can be used to protect a pattern beginning with "−".

−f *FILE*, **−−file=***FILE*
Obtain patterns from *FILE*, one per line. If this option is used multiple times or is combined with the **−e** (**−−regexp**) option, search for all patterns given. The empty file contains zero patterns, and therefore matches nothing.

−i, **−−ignore−case**
Ignore case distinctions in both the *PATTERN* and the input files.

−v, **−−invert−match**
Invert the sense of matching, to select non-matching lines.

−w, **−−word−regexp**
Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore. This option has no effect if **−x** is also specified.

−x, **−−line−regexp**
Select only those matches that exactly match the whole line. For a regular expression pattern, this is like parenthesizing the pattern and then surrounding it with ^ and **$**.

−y Obsolete synonym for **−i**.

General Output Control

−c, **−−count**
Suppress normal output; instead print a count of matching lines for each input file. With the **−v**, **−−invert−match** option (see below), count non-matching lines.

−−color[=*WHEN*], **−−colour**[=*WHEN*]
Surround the matched (non-empty) strings, matching lines, context lines, file names, line numbers, byte offsets, and separators (for fields and groups of context lines) with escape sequences to display them in color on the terminal. The colors are defined by the environment variable **GREP_COLORS**. The deprecated environment variable **GREP_COLOR** is still supported, but its setting does not have priority. *WHEN* is **never**, **always**, or **auto**.

−L, **−−files−without−match**
Suppress normal output; instead print the name of each input file from which no output would normally have been printed. The scanning will stop on the first match.

−l, **−−files−with−matches**
Suppress normal output; instead print the name of each input file from which output would normally have been printed. The scanning will stop on the first match.

−m *NUM*, **−−max−count=***NUM*
Stop reading a file after *NUM* matching lines. If the input is standard input from a regular file, and *NUM* matching lines are output, **grep** ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When **grep** stops after *NUM* matching lines, it outputs any trailing context lines. When the **−c** or **−−count** option is also used, **grep** does not output a count greater than *NUM*. When the **−v** or **−−invert−match** option is also used, **grep** stops after outputting *NUM* non-matching lines.

−o, **−−only−matching**
Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.

−q, **−−quiet**, **−−silent**
Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected. Also see the **−s** or **−−no−messages** option.

−s, **−−no−messages**
Suppress error messages about nonexistent or unreadable files.

Output Line Prefix Control

−b, **−−byte−offset**
Print the 0-based byte offset within the input file before each line of output. If **−o** (**−−only−matching**) is specified, print the offset of the matching part itself.

−H, **−−with−filename**
Print the file name for each match. This is the default when there is more than one file to search.

−h, **−−no−filename**
Suppress the prefixing of file names on output. This is the default when there is only one file (or only standard input) to search.

−−label=*LABEL*
Display input actually coming from standard input as input coming from file *LABEL*. This is especially useful when implementing tools like **zgrep**, e.g., **gzip -cd foo.gz | grep --label=foo -H something**. See also the **−H** option.

−n, **−−line−number**
Prefix each line of output with the 1-based line number within its input file.

−T, **−−initial−tab**
Make sure that the first character of actual line content lies on a tab stop, so that the alignment of tabs looks normal. This is useful with options that prefix their output to the actual content: **−H**,**−n**, and **−b**. In order to improve the probability that lines from a single file will all start at the same column, this also causes the line number and byte offset (if present) to be printed in a minimum size field width.

−u, **−−unix−byte−offsets**
Report Unix-style byte offsets. This switch causes **grep** to report byte offsets as if the file were a Unix-style text file, i.e., with CR characters stripped off. This will produce results identical to running **grep** on a Unix machine. This option has no effect unless **−b** option is also used; it has no effect on platforms other than MS-DOS and MS-Windows.

−Z, **−−null**
Output a zero byte (the ASCII **NUL** character) instead of the character that normally follows a file name. For example, **grep −lZ** outputs a zero byte after each file name instead of the usual newline. This option makes the output unambiguous, even in the presence of file names containing unusual characters like newlines. This option can be used with commands like **find −print0**, **perl −0**, **sort −z**, and **xargs −0** to process arbitrary file names, even those that contain newline characters.

Context Line Control

−A *NUM*, **−−after−context=***NUM*
Print *NUM* lines of trailing context after matching lines. Places a line containing a group separator (**−−**) between contiguous groups of matches. With the **−o** or **−−only−matching** option, this has no effect and a warning is given.

−B *NUM*, **−−before−context=***NUM*
Print *NUM* lines of leading context before matching lines. Places a line containing a group separator (**−−**) between contiguous groups of matches. With the **−o** or **−−only−matching** option, this has no effect and a warning is given.

−C *NUM*, *−NUM*, **−−context=***NUM*
Print *NUM* lines of output context. Places a line containing a group separator (**−−**) between contiguous groups of matches. With the **−o** or **−−only−matching** option, this has no effect and a warning is given.

File and Directory Selection

−a, **−−text**
Process a binary file as if it were text; this is equivalent to the **−−binary−files=text** option.

−−binary−files=*TYPE*
If a file's data or metadata indicate that the file contains binary data, assume that the file is of type *TYPE*. Non-text bytes indicate binary data; these are either output bytes that are improperly encoded for the current locale, or null input bytes when the **−z** option is not given.

By default, *TYPE* is **binary**, and when **grep** discovers that a file is binary it suppresses any further output, and instead outputs either a one-line message saying that a binary file matches, or no message if there is no match.

If *TYPE* is **without-match**, when **grep** discovers that a file is binary it assumes that the rest of the file does not match; this is equivalent to the **−I** option.

If *TYPE* is **text**, **grep** processes a binary file as if it were text; this is equivalent to the **−a** option.

When *type* is **binary**, **grep** may treat non-text bytes as line terminators even without the **−z** option. This means choosing **binary** versus **text** can affect whether a pattern matches a file. For example, when *type* is **binary** the pattern **q$ might** match **q** immediately followed by a null byte, even though this is not matched when *type* is **text**. Conversely, when *type* is **binary** the pattern **.**

(period) might not match a null byte.

Warning: The **−a** option might output binary garbage, which can have nasty side effects if the output is a terminal and if the terminal driver interprets some of it as commands. On the other hand, when reading files whose text encodings are unknown, it can be helpful to use **−a** or to set **LC_ALL='C'** in the environment, in order to find more matches even if the matches are unsafe for direct display.

−D *ACTION*, **−−devices=***ACTION*
If an input file is a device, FIFO or socket, use *ACTION* to process it. By default, *ACTION* is **read**, which means that devices are read just as if they were ordinary files. If *ACTION* is **skip**, devices are silently skipped.

−d *ACTION*, **−−directories=***ACTION*
If an input file is a directory, use *ACTION* to process it. By default, *ACTION* is **read**, i.e., read directories just as if they were ordinary files. If *ACTION* is **skip**, silently skip directories. If *ACTION* is **recurse**, read all files under each directory, recursively, following symbolic links only if they are on the command line. This is equivalent to the **−r** option.

−−exclude=*GLOB*
Skip any command-line file with a name suffix that matches the pattern *GLOB*, using wildcard matching; a name suffix is either the whole name, or any suffix starting after a / and before a +non-/. When searching recursively, skip any subfile whose base name matches *GLOB*; the base name is the part after the last /. A pattern can use *****, **?**, and **[**...**]** as wildcards, and \ to quote a wildcard or backslash character literally.

−−exclude-from=*FILE*
Skip files whose base name matches any of the file-name globs read from *FILE* (using wildcard matching as described under **−−exclude**).

−−exclude-dir=*GLOB*
Skip any command-line directory with a name suffix that matches the pattern *GLOB*. When searching recursively, skip any subdirectory whose base name matches *GLOB*. Ignore any redundant trailing slashes in *GLOB*.

−I Process a binary file as if it did not contain matching data; this is equivalent to the **−−binary−files=without-match** option.

−−include=*GLOB*
Search only files whose base name matches *GLOB* (using wildcard matching as described under **−−exclude**).

−r, **−−recursive**
Read all files under each directory, recursively, following symbolic links only if they are on the command line. Note that if no file operand is given, grep searches the working directory. This is equivalent to the **−d recurse** option.

−R, **−−dereference−recursive**
Read all files under each directory, recursively. Follow all symbolic links, unlike **−r**.

Other Options

−−line−buffered
Use line buffering on output. This can cause a performance penalty.

−U, **−−binary**
Treat the file(s) as binary. By default, under MS-DOS and MS-Windows, **grep** guesses whether a file is text or binary as described for the **−−binary−files** option. If **grep** decides the file is a text file, it strips the CR characters from the original file contents (to make regular expressions with **^** and **$** work correctly). Specifying **−U** overrules this guesswork, causing all files to be read and passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some regular expressions to fail. This option has no effect on platforms other than MS-DOS and MS-Windows.

−z, **−−null−data**
Treat input and output data as sequences of lines, each terminated by a zero byte (the ASCII NUL character) instead of a newline. Like the **−Z** or **−−null** option, this option can be used with commands like **sort -z** to process arbitrary file names.

REGULAR EXPRESSIONS

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

grep understands three different versions of regular expression syntax: “basic” (BRE), “extended” (ERE) and “perl” (PCRE). In GNU **grep**, there is no difference in available functionality between basic and extended syntaxes. In other implementations, basic regular expressions are less powerful. The following description applies to extended regular expressions; differences for basic regular expressions are summarized afterwards. Perl-compatible regular expressions give additional functionality, and are documented in pcresyntax(3) and pcrepattern(3), but work only if PCRE is available in the system.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any meta-character with special meaning may be quoted by preceding it with a backslash.

The period **.** matches any single character.

Character Classes and Bracket Expressions

A *bracket expression* is a list of characters enclosed by **[** and **]**. It matches any single character in that list; if the first character of the list is the caret ^ then it matches any character *not* in the list. For example, the regular expression **[0123456789]** matches any single digit.

Within a bracket expression, a *range expression* consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters, inclusive, using the locale’s collating sequence and character set. For example, in the default C locale, **[a−d]** is equivalent to **[abcd]**. Many locales sort characters in dictionary order, and in these locales **[a−d]** is typically not equivalent to **[abcd]**; it might be equivalent to **[aBbCcDd]**, for example. To obtain the traditional interpretation of bracket expressions, you can use the C locale by setting the **LC_ALL** environment variable to the value **C**.

Finally, certain named classes of characters are predefined within bracket expressions, as follows. Their names are self explanatory, and they are **[:alnum:]**, **[:alpha:]**, **[:cntrl:]**, **[:digit:]**, **[:graph:]**, **[:lower:]**, **[:print:]**, **[:punct:]**, **[:space:]**, **[:upper:]**, and **[:xdigit:].** For example, **[[:alnum:]]** means the character class of numbers and letters in the current locale. In the C locale and ASCII character set encoding, this is the same as **[0−9A−Za−z]**. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket expression.) Most meta-characters lose their special meaning inside bracket expressions. To include a literal **]** place it first in the list. Similarly, to include a literal ^ place it anywhere but first. Finally, to include a literal − place it last.

Anchoring

The caret ^ and the dollar sign **$** are meta-characters that respectively match the empty string at the beginning and end of a line.

The Backslash Character and Special Expressions

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. The symbol **\w** is a synonym for **[_[:alnum:]]** and **\W** is a synonym for **[^_[:alnum:]]**.

Repetition

A regular expression may be followed by one of several repetition operators:

? The preceding item is optional and matched at most once.
***** The preceding item will be matched zero or more times.
+ The preceding item will be matched one or more times.
{*n*} The preceding item is matched exactly *n* times.
{*n*,} The preceding item is matched *n* or more times.

{,*m*} The preceding item is matched at most *m* times. This is a GNU extension.
{*n*,*m*} The preceding item is matched at least *n* times, but not more than *m* times.

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

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

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

Back References and Subexpressions
The back-reference *n*, where *n* is a single digit, matches the substring previously matched by the *n*th parenthesized subexpression of the regular expression.

Basic vs Extended Regular Expressions
In basic regular expressions the meta-characters **?**, **+**, **{**, **|**, **(**, and **)** lose their special meaning; instead use the backslashed versions **?**, **+**, **{**, **|**, **(**, and **)**.

ENVIRONMENT VARIABLES

The behavior of **grep** is affected by the following environment variables.

The locale for category **LC_***foo* is specified by examining the three environment variables **LC_ALL**, **LC_***foo*, **LANG**, in that order. The first of these variables that is set specifies the locale. For example, if **LC_ALL** is not set, but **LC_MESSAGES** is set to **pt_BR**, then the Brazilian Portuguese locale is used for the **LC_MESSAGES** category. The C locale is used if none of these environment variables are set, if the locale catalog is not installed, or if **grep** was not compiled with national language support (NLS). The shell command **locale −a** lists locales that are currently available.

GREP_OPTIONS
This variable specifies default options to be placed in front of any explicit options. As this causes problems when writing portable scripts, this feature will be removed in a future release of **grep**, and **grep** warns if it is used. Please use an alias or script instead.

GREP_COLOR
This variable specifies the color used to highlight matched (non-empty) text. It is deprecated in favor of **GREP_COLORS**, but still supported. The **mt**, **ms**, and **mc** capabilities of **GREP_COLORS** have priority over it. It can only specify the color used to highlight the matching non-empty text in any matching line (a selected line when the **-v** command-line option is omitted, or a context line when **-v** is specified). The default is **01;31**, which means a bold red foreground text on the terminal's default background.

GREP_COLORS
Specifies the colors and other attributes used to highlight various parts of the output. Its value is a colon-separated list of capabilities that defaults to **ms=01;31:mc=01;31:sl=:cx=:fn=35:ln=32:bn=32:se=36** with the **rv** and **ne** boolean capabilities omitted (i.e., false). Supported capabilities are as follows.

sl= SGR substring for whole selected lines (i.e., matching lines when the **−v** command-line option is omitted, or non-matching lines when **−v** is specified). If however the boolean **rv** capability and the **−v** command-line option are both specified, it applies to context matching lines instead. The default is empty (i.e., the terminal's default color pair).

cx= SGR substring for whole context lines (i.e., non-matching lines when the **−v** command-line option is omitted, or matching lines when **−v** is specified). If however the boolean **rv** capability and the **−v** command-line option are both specified, it applies to selected non-matching lines instead. The default is empty (i.e., the terminal's default color pair).

rv Boolean value that reverses (swaps) the meanings of the **sl=** and **cx=** capabilities when the **−v** command-line option is specified. The default is false (i.e., the capability is omitted).

mt=01;31
SGR substring for matching non-empty text in any matching line (i.e., a selected line when the **−v** command-line option is omitted, or a context line when **−v** is specified). Setting this is equivalent to setting both **ms=** and **mc=** at once to the same value. The default is a bold red text foreground over the current line background.

ms=01;31
SGR substring for matching non-empty text in a selected line. (This is only used when the **−v** command-line option is omitted.) The effect of the **sl=** (or **cx=** if **rv**) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

mc=01;31
SGR substring for matching non-empty text in a context line. (This is only used when the **−v** command-line option is specified.) The effect of the **cx=** (or **sl=** if **rv**) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

fn=35 SGR substring for file names prefixing any content line. The default is a magenta text foreground over the terminal's default background.

ln=32 SGR substring for line numbers prefixing any content line. The default is a green text foreground over the terminal's default background.

bn=32 SGR substring for byte offsets prefixing any content line. The default is a green text foreground over the terminal's default background.

se=36 SGR substring for separators that are inserted between selected line fields (**:**), between context line fields, (**−**), and between groups of adjacent lines when nonzero context is specified (**−−**). The default is a cyan text foreground over the terminal's default background.

ne Boolean value that prevents clearing to the end of line using Erase in Line (EL) to Right (**\33[K**) each time a colorized item ends. This is needed on terminals on which EL is not supported. It is otherwise useful on terminals for which the **back_color_erase** (**bce**) boolean terminfo capability does not apply, when the chosen highlight colors do not affect the background, or when EL is too slow or causes too much flicker. The default is false (i.e., the capability is omitted).

Note that boolean capabilities have no =... part. They are omitted (i.e., false) by default and become true when specified.

See the Select Graphic Rendition (SGR) section in the documentation of the text terminal that is used for permitted values and their meaning as character attributes. These substring values are integers in decimal representation and can be concatenated with semicolons. **grep** takes care of assembling the result into a complete SGR sequence (**\33[**...**m**). Common values to concatenate include **1** for bold, **4** for underline, **5** for blink, **7** for inverse, **39** for default foreground color, **30** to **37** for foreground colors, **90** to **97** for 16-color mode foreground colors, **38;5;0** to **38;5;255** for 88-color and 256-color modes foreground colors, **49** for default background color, **40** to **47** for background colors, **100** to **107** for 16-color mode background colors, and **48;5;0** to **48;5;255** for 88-color and 256-color modes background colors.

LC_ALL, **LC_COLLATE**, **LANG**
These variables specify the locale for the **LC_COLLATE** category, which determines the collating sequence used to interpret range expressions like **[a−z]**.

LC_ALL, **LC_CTYPE**, **LANG**
: These variables specify the locale for the **LC_CTYPE** category, which determines the type of characters, e.g., which characters are whitespace. This category also determines the character encoding, that is, whether text is encoded in UTF-8, ASCII, or some other encoding. In the C or POSIX locale, all characters are encoded as a single byte and every byte is a valid character.

LC_ALL, **LC_MESSAGES**, **LANG**
: These variables specify the locale for the **LC_MESSAGES** category, which determines the language that **grep** uses for messages. The default C locale uses American English messages.

POSIXLY_CORRECT
: If set, **grep** behaves as POSIX requires; otherwise, **grep** behaves more like other GNU programs. POSIX requires that options that follow file names must be treated as file names; by default, such options are permuted to the front of the operand list and are treated as options. Also, POSIX requires that unrecognized options be diagnosed as "illegal", but since they are not really against the law the default is to diagnose them as "invalid". **POSIXLY_CORRECT** also disables ***_N_*_GNU_nonoption_argv_flags_**, described below.

***_N_*_GNU_nonoption_argv_flags_**
: (Here *N* is **grep**'s numeric process ID.) If the *i*th character of this environment variable's value is **1**, do not consider the *i*th operand of **grep** to be an option, even if it appears to be one. A shell can put this variable in the environment for each command it runs, specifying which operands are the results of file name wildcard expansion and therefore should not be treated as options. This behavior is available only with the GNU C library, and only when **POSIXLY_CORRECT** is not set.

EXIT STATUS

Normally the exit status is 0 if a line is selected, 1 if no lines were selected, and 2 if an error occurred. However, if the **-q** or **--quiet** or **--silent** is used and a line is selected, the exit status is 0 even if an error occurred.

COPYRIGHT

BUGS

Reporting Bugs

Email bug reports to the bug-reporting address ⟨bug-grep@gnu.org⟩. An email archive ⟨http://lists.gnu.org/mailman/listinfo/bug-grep⟩ and a bug tracker ⟨http://debbugs.gnu.org/cgi/pkgreport.cgi?package=grep⟩ are available.

Known Bugs

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

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

SEE ALSO

Regular Manual Pages

awk(1), cmp(1), diff(1), find(1), gzip(1), perl(1), sed(1), sort(1), xargs(1), zgrep(1), read(2), pcre(3), pcresyntax(3), pcrepattern(3), terminfo(5), glob(7), regex(7).

POSIX Programmer's Manual Page

grep(1p).

Full Documentation

A complete manual ⟨http://www.gnu.org/software/grep/manual/⟩ is available. If the **info** and **grep** programs are properly installed at your site, the command

info grep

should give you access to the complete manual.

NOTES

This man page is maintained only fitfully; the full documentation is often more up-to-date.

NAME

htop − interactive process viewer

SYNOPSIS

htop [−*dChusv*]

DESCRIPTION

Htop is a free (GPL) ncurses-based process viewer for Linux.

It is similar to top, but allows you to scroll vertically and horizontally, so you can see all the processes running on the system, along with their full command lines, as well as viewing them as a process tree, selecting multiple processes and acting on them all at once.

Tasks related to processes (killing, renicing) can be done without entering their PIDs.

COMMAND-LINE OPTIONS

Mandatory arguments to long options are mandatory for short options too.

−d −−delay=DELAY
Delay between updates, in tenths of seconds

−C −−no-color −−no-colour
Start htop in monochrome mode

−h −−help
Display a help message and exit

−p −−pid=PID,PID...
Show only the given PIDs

−s −−sort−key COLUMN
Sort by this column (use −−sort−key help for a column list)

−u −−user=USERNAME
Show only the processes of a given user

−v −−version
Output version information and exit

INTERACTIVE COMMANDS

The following commands are supported while in htop:

Up, Alt-k
Select (hightlight) the previous process in the process list. Scroll the list if necessary.

Down, Alt-j
Select (hightlight) the next process in the process list. Scroll the list if necessary.

Left, Alt-h
Scroll the process list left.

Right, Alt-l
Scroll the process list right.

PgUp, PgDn
Scroll the process list up or down one window.

Home
Scroll to the top of the process list and select the first process.

End Scroll to the bottom of the process list and select the last process.

Ctrl-A, ^
Scroll left to the beginning of the process entry (i.e. beginning of line).

Ctrl-E, $
Scroll right to the end of the process entry (i.e. end of line).

Space
Tag or untag a process. Commands that can operate on multiple processes, like "kill", will then apply over the list of tagged processes, instead of the currently highlighted one.

U Untag all processes (remove all tags added with the Space key).

s Trace process system calls: if strace(1) is installed, pressing this key will attach it to the currently selected process, presenting a live update of system calls issued by the process.

l Display open files for a process: if lsof(1) is installed, pressing this key will display the list of file descriptors opened by the process.

F1, h, ?
Go to the help screen

F2, S
Go to the setup screen, where you can configure the meters displayed at the top of the screen, set various display options, choose among color schemes, and select which columns are displayed, in which order.

F3, / Incrementally search the command lines of all the displayed processes. The currently selected (highlighted) command will update as you type. While in search mode, pressing F3 will cycle through matching occurrences.

**F4, ** Incremental process filtering: type in part of a process command line and only processes whose names match will be shown. To cancel filtering, enter the Filter option again and press Esc.

F5, t Tree view: organize processes by parenthood, and layout the relations between them as a tree. Toggling the key will switch between tree and your previously selected sort view. Selecting a sort view will exit tree view.

F6 On sorted view, select a field for sorting, also accessible through < and >. The current sort field is indicated by a highlight in the header. On tree view, expand or collapse the current subtree. A "+" indicator in the tree node indicates that it is collapsed.

F7,] Increase the selected process's priority (subtract from 'nice' value). This can only be done by the superuser.

F8, [Decrease the selected process's priority (add to 'nice' value)

F9, k
"Kill" process: sends a signal which is selected in a menu, to one or a group of processes. If processes were tagged, sends the signal to all tagged processes. If none is tagged, sends to the currently selected process.

F10, q
Quit

I Invert the sort order: if sort order is increasing, switch to decreasing, and vice-versa.

+, − When in tree view mode, expand or collapse subtree. When a subtree is collapsed a "+" sign shows to the left of the process name.

a (on multiprocessor machines)
Set CPU affinity: mark which CPUs a process is allowed to use.

u Show only processes owned by a specified user.

M Sort by memory usage (top compatibility key).

P Sort by processor usage (top compatibility key).

T Sort by time (top compatibility key).

F "Follow" process: if the sort order causes the currently selected process to move in the list, make the selection bar follow it. This is useful for monitoring a process: this way, you can keep a process always visible on screen. When a movement key is used, "follow" loses effect.

K Hide kernel threads: prevent the threads belonging the kernel to be displayed in the process list. (This is a toggle key.)

H Hide user threads: on systems that represent them differently than ordinary processes (such as recent NPTL-based systems), this can hide threads from userspace processes in the process list. (This is a toggle key.)

p Show full paths to running programs, where applicable. (This is a toggle key.)

Ctrl-L
Refresh: redraw screen and recalculate values.

Numbers
PID search: type in process ID and the selection highlight will be moved to it.

COLUMNS

The following columns can display data about each process. A value of ’−’ in all the rows indicates that a column is unsupported on your system, or currently unimplemented in htop. The names below are the ones used in the "Available Columns" section of the setup screen. If a different name is shown in htop’s main screen, it is shown below in parenthesis.

Command
The full command line of the process (i.e. program name and arguments).

PID The process ID.

STATE (S)
The state of the process:
S for sleeping (idle)
R for running
D for disk sleep (uninterruptible)
Z for zombie (waiting for parent to read its exit status)
T for traced or suspended (e.g by SIGTSTP)
W for paging

PPID
The parent process ID.

PGRP
The process’s group ID.

SESSION (SESN)
The process’s session ID.

TTY_NR (TTY)
The controlling terminal of the process.

TPGID
The process ID of the foreground process group of the controlling terminal.

MINFLT
The number of page faults happening in the main memory.

CMINFLT
The number of minor faults for the process’s waited-for children (see MINFLT above).

MAJFLT
The number of page faults happening out of the main memory.

CMAJFLT
The number of major faults for the process’s waited-for children (see MAJFLT above).

UTIME (UTIME+)
The user CPU time, which is the amount of time the process has spent executing on the CPU in user mode (i.e. everything but system calls), measured in clock ticks.

STIME (STIME+)
The system CPU time, which is the amount of time the kernel has spent executing system calls on behalf of the process, measured in clock ticks.

CUTIME (CUTIME+)
The children's user CPU time, which is the amount of time the process's waited-for children have spent executing in user mode (see UTIME above).

CSTIME (CSTIME+)
The children's system CPU time, which is the amount of time the kernel has spent executing system calls on behalf of all the process's waited-for children (see STIME above).

PRIORITY (PRI)
The kernel's internal priority for the process, usually just its nice value plus twenty. Different for real-time processes.

NICE (NI)
The nice value of a process, from 19 (low priority) to -20 (high priority). A high value means the process is being nice, letting others have a higher relative priority. The usual OS permission restrictions for adjusting priority apply.

STARTTIME (START)
The time the process was started.

PROCESSOR (CPU)
The ID of the CPU the process last executed on.

M_SIZE (VIRT)
The size of the virtual memory of the process.

M_RESIDENT (RES)
The resident set size (text + data + stack) of the process (i.e. the size of the process's used physical memory).

M_SHARE (SHR)
The size of the process's shared pages.

M_TRS (CODE)
The text resident set size of the process (i.e. the size of the process's executable instructions).

M_DRS (DATA)
The data resident set size (data + stack) of the process (i.e. the size of anything except the process's executable instructions).

M_LRS (LIB)
The library size of the process.

M_DT (DIRTY)
The size of the dirty pages of the process.

ST_UID (UID)
The user ID of the process owner.

PERCENT_CPU (CPU%)
The percentage of the CPU time that the process is currently using.

PERCENT_MEM (MEM%)
The percentage of memory the process is currently using (based on the process's resident memory size, see M_RESIDENT above).

USER
The username of the process owner, or the user ID if the name can’t be determined.

TIME (TIME+)
The time, measured in clock ticks that the process has spent in user and system time (see UTIME, STIME above).

NLWP
The number of threads in the process.

TGID
The thread group ID.

CTID
OpenVZ container ID, a.k.a virtual environment ID.

VPID
OpenVZ process ID.

VXID
VServer process ID.

RCHAR (RD_CHAR)
The number of bytes the process has read.

WCHAR (WR_CHAR)
The number of bytes the process has written.

SYSCR (RD_SYSC)
The number of read(2) syscalls for the process.

SYSCW (WR_SYSC)
The number of write(2) syscalls for the process.

RBYTES (IO_RBYTES)
Bytes of read(2) I/O for the process.

WBYTES (IO_WBYTES)
Bytes of write(2) I/O for the process.

CNCLWB (IO_CANCEL)
Bytes of cancelled write(2) I/O.

IO_READ_RATE (DISK READ)
The I/O rate of read(2) in bytes per second, for the process.

IO_WRITE_RATE (DISK WRITE)
The I/O rate of write(2) in bytes per second, for the process.

IO_RATE (DISK R/W)
The I/O rate, IO_READ_RATE + IO_WRITE_RATE (see above).

CGROUP
Which cgroup the process is in.

OOM
OOM killer score.

IO_PRIORITY (IO)
The I/O scheduling class followed by the priority if the class supports it:
R for Realtime
B for Best-effort
id for Idle

All other flags
Currently unsupported (always displays '-').

CONFIG FILE

By default htop reads its configuration from the XDG-compliant path ~/.config/htop/htoprc -- the configuration file is overwritten by htop's in-program Setup configuration, so it should not be hand-edited. If no user configuration exists htop tries to read the system-wide configuration from /etc/htoprc and as a last resort, falls back to its hard coded defaults.

You may override the location of the configuration file using the $HTOPRC environment variable (so you can have multiple configurations for different machines that share the same home directory, for example).

MEMORY SIZES

Memory sizes in htop are displayed as they are in tools from the GNU Coreutils (when ran with the --human-readable option). This means that sizes are printed in powers of 1024. (e.g., 1023M = 1072693248 Bytes)

The decision to use this convention was made in order to conserve screen space and make memory size representations consistent throughout htop.

SEE ALSO

proc(5), top(1), free(1), ps(1), uptime(1), limits.conf(5)

AUTHORS

htop is developed by Hisham Muhammad <hisham@gobolinux.org>.

This man page was written by Bartosz Fenski <fenio@o2.pl> for the Debian GNU/Linux distribution (but it may be used by others). It was updated by Hisham Muhammad, and later by Vincent Launchbury, who wrote the 'Columns' section.

NOME

ln − cria uma ligação simbólica entre arqvuios

SINOPSE

ln [opções] *origem* **[***destino***]**
ln [opções] *origem... diretório*

Opções POSIX: **[−f]**

Opções GNU (forma reduzida): **[−bdfinsvF] [−S** *backup-suffix***] [−V {numbered,existing,simple}] [−−help] [−−version] [−−]**

DESCRIÇÃO

Existem dois conceitos de ’ligação’no Unix, usualmente nomeadas de ligação forte e ligação fraca. Uma ligação forte é exatamente um nome para o arquivo. (E um arquivo pode ter vários nomes. Aquilo só é apaguado do disco somente quando o último nome é removido. O número de nomes é dado por **ls**(1). Não havia coisa semelhante no nome ’original’: todos os nome tem o mesmo status. Usualmente, mas não necessariamente, todos os nomes do arquivo encontrado no sistema de arquivos contém os mesmo dados.)

Uma ligação fraca (ou ligação simbólica, ou symlink) é um material totalmente diferente: é um pequeno arquivo especial que contém o nome de caminho. Assim, a ligação fraca pode apontar arquivos em diferentes sistemas de arquivos (talvez NFS montadas de diferentes máquinas), e não precisa de ponto para o arquivo existente atualmente. Quando acessado (com a chamada de sistema **open**(2) ou **stat**(2) , uma referência para o symlink é substituída pelo núcleo do sistema operacional com uma referência para o arquivo nomeado no nome de caminho. (De qualquer forma, com **rm**(1) ou **unlink**(2) a ligação é removida, não o arquivo que ela aponta. Estas são chamadas especiais de sistema **lstat**(2) e **readlink**(2) que lêem o status do symlink e para onde aponta o nome do arquivo. Para várias outras chamadas de sistema há um pouco de incerteza e variação entre sistemas operacionais fazem com que as operações atuem no próprio symlink, ou no arquivo apontado.

ln cria ligações entre arquivos. Por padrão, ele cria ligações fortes; com a opção **−s** , ele cria ligações simbólicas (ou ’fracas’).

Se somente um arquivo é fornecido, a ligação daquele arquivo é feita dentro do diretório atual, o qual é, criado um link para aquele arquivo no diretório atual, com o nome igual para (o último componente sobre) o nome daquele arquivo. (Isto é uma extensão GNU.) De outra forma, se o último argumento é o nome de um diretório existente, **ln** deverá criar uma ligação para cada arquivo de *origem* mencionada naquele diretório, como o nome iqual ao (o último componente de) nome daquele arquivo de *origem* . (Veja a descrição da opção **−−no−dereference** abaixo.) De outra forma, se somente dois arquivos são fornecidos, ele cria uma ligação nomeada *destino* para o arquivo de *origem*. Será um erro se o último argumento não for um diretório e mais que dois arquivos forem fornecidos.

Por padrão, **ln** não remove arquivos ou ligações simbólicas existentes. (Assim, é possível o seu uso para propósitos de fechamento: que só terá sucesso se *destino* já não exista.) Mas pode ser forçado a fazer assim com a opção −f.

Em implementações existentes, é possível criar ligações fortes para um diretório, isto pode ser feito somente pelo super-usuário. POSIX proíbe a chamada de sistema **link**(2) e o utilitário **ln** de criar ligações fortes para diretórios (mas não proíbe de criar ligações forte entre sistemas de arquivos diferentes).

OPCÕES POSIX

−f Remove o arquivo de destino existente.

OPÇÕES GNU

−d, −F, −−directory
Permite que o super usuário crie ligações fortes para diretórios.

−f, −−force
Remove o arquivo de destino existente.

−i, −−interactive
Questiona se remove o arquivo de destino existente.

−n, −−no−dereference
Quando é fornecido uma destinação explicita daquele symlink para um diretório, trata o destino como se fosse um arquivo normal.

Quando o destino é no diretório atual (e não um symlink para ele), não há ambigüidade. A ligação é criada naquele diretório. Mas quando o destino especificado é um synlink para um diretório, há dois modos para tratar o pedido do usuário. **ln** pode tratar o destino como um diretório normal e criar uma ligação para ele. Na outra interpretação, o destino pode ser visualizado não como diretório - e sim como uma symlink para ele. Neste caso, **ln** dever apaguar ou criar um backup daquele symlink antes de criar o novo link. O padrão é para se tratar o destino como um symlink para o diretório como se fosse um diretório.

−s, −−symbolic
Cria ligações fracas ao invês de ligações fortes. Esta opção meramente produz uma mensagem de erro nos sistemas que não suportam ligações fracas.

−v, −−verbose
Descreve toda ação ocorrida para cada ligação.

OPÇÕES BACKUP GNU

As versões GNU de programas como **cp**, **mv**, **ln**, **install** e **patch** podem criar backup de sobre os arquivos que foram reescritos, alterados ou destruídos se isto é desejado. Aqueles arquivos de bakcup se desejados são indicado pela opção −b. Como eles deveriam ser nomeados é especificado pela opção de −V. No caso do nome do arquivo posterior é determinado para o nome de um arquivo estendido por um sufixo, este sufixo é especificado pela opção de −S.

−b, −−backup
Cria backups dos arquivos reescritos ou removidos

−S *SUFFIX*, **−−suffix=***SUFFIX*
Anexo *SUFFIX* para cada backup feito. Se esta opção não é especificada, o valor da variável de ambiente **SIMPLE_BACKUP_SUFFIX** é usada. E se **SIMPLE_BACKUP_SUFFIX** não é selecionada, o padrão é ‘~’.

−V *METHOD*, **−−version−control=***METHOD*
Especifica como os arquivos de cópias de segurança serão nomedos. O argumento do *METHOD* pode ser ’numbered’ (ou ’t’), ’existing’ (ou ’nil’), ou ’never’ (ou Se esta opção não é especificada o valor da variável de ambiente **VERSION_CONTROL** é usada. E se **VERSION_CONTROL** não é selecionada, o tipo a cópia de segurança padrão é ’existing’.

Esta opção corresponde a variável do Emacs ’version-control’. Os *MÉTODOS* validos são (são aceitas abreviações únicas):

t, **numbered**
Sempre cria backups numerados.

nil, **existing**
Cria backups numerados dos arquivos que já os tem , simples auxilio dos outros.

never, **simple**
Sempre cria backups simples.

OPÇÕES PADRÃO GNU

−−help Imprime a mensagem de uso na saída padrão e sai.

−−version
Imprime a versão na saída padrão e sai.

−− Encerra a lista de opção.

AMBIENTE

As variáveis LANG, LC_ALL, LC_CTYPE and LC_MESSAGES tem seu significado usual.

OBEDECENDO

POSIX 1003.2. De qualquer forma, POSIX 1003.2 (1996) não cita as ligações fracas. Ligação fracas foram introduzidas pelo BSD, e não ocorrem nas especificações do System V release 3 (e anteriores).

VEJA TAMBÉM

ls(1), **rm**(1), **link**(2), **lstat**(2), **open**(2), **readlink**(2), **stat**(2), **unlink**(2)

NOTAS

Esta página descreve como **ln** é encontrada no pacote Utilitários de Arquivos 4.0; outras versões podem ser um pouco diferente. Envie correções e adições para aeb@cwi.nl. Relatório de problemas no programa para fileutils-bugs@gnu.ai.mit.edu.

TRADUZIDO POR LDP-BR em 21/08/2000.

André L. Fassone Canova <lonelywolf@blv.com.br> (tradução) Ricardo C.O. Freitas <english.quest@best-service.com> (revisão)

NOME

ls, dir, vdir − lista o conteúdo do diretório

SINOPSE

ls [*opções***] [***arquivo...***]**
dir [*arquivo...***]**
vdir [*arquivo...***]**

Opções POSIX: **[−CFRacdilqrtu1]**

Opções GNU (forma reduzida): **[−1abcdfghiklmnopqrstuvwxABCDFGHLNQRSUX] [−w** *coluna***] [−T** *coluna***] [−I** *modelo***] [−−full−time] [−−show−control−chars] [−−block−size=***size***] [−−format={long,verbose,commas,across,vertical,single−column}] [−−sort={none,time,size,extension}] [−−time={atime,access,use,ctime,status}] [−−color[={none,auto,always}]] [−−help] [−−version] [−−]**

DESCRIÇÃO

O programa **ls** lista primeiramente seus argumentos que não sejam *arquivos* de diretórios, e, então, para um argumento diretório, todos os arquivos listáveis incluídos dentro daquele diretório. Se nenhum argumento de opção estiver presente, o argumento padrão '.' (diretório atual) é assumido. A opção −d faz com que diretórios sejam tratados como arquivos. Um arquivo é listável quando ou seu nome não inicia com '.', ou a opção −a for fornecida.

Cada uma das listas de arquivos (que pertence a arquivos regulares, e para cada diretório a lista interna de arquivos) é ordenada separadamente de acordo com a seqüência correspondente, no local atual. Quando a opção −l é fornecida, cada uma das listas é precedida por uma linha de sumário que fornece o tamanho total de todos os arquivos na lista, medidos em grupos de 512 bytes.

A saída é a saída padrão, uma entrada por linha, até que a saída em multiplas colunas seja requesitada pela opção −C. De qualquer forma, para saída em um terminal, em linha simples ou multiplas colunas é indefinido. As opções −1 e −C podem ser usadas para forçar a saída em linha simples ou em multiplas colunas, respectivamente.

OPÇÕES POSIX

−C Lista arquivos em colunas, ordenados verticalmente.

−F Sufixa o nome de cada diretório com '/', cada nome de FIFO com '|', e cada nome de executável com '*'.

−R Lista os diretórios encontrados, recursivamente.

−a Inclue os arquivos com o nome iniciando com '.' na listagem.

−c Usa o status do tempo de alteração ao invés do tempo de modificação para ordenar (com −t) ou listar (com −l).

−d Lista nome de diretórios como arquivo, preferencialmente no lugar de seus conteúdos.

−i Precede a saída para o arquivo pelo número serial do arquivo (número do i-node).

−l Escreve (no formato de coluna simples) o modo do arquivo, o número de ligações para o arquivo, o nome do proprietário, o nome do grupo, o tamanho do arquivo (em bytes), o rótulo de tempo, e o nome do arquivo.

Os tipos de arquivos são os seguintes: **−** para um arquivo comum, **d** para um diretório, **b** para um dispositivo especial de bloco, **c** para um dispositivo especial de caractere, **l** para uma ligação simbólica, **p** para um FIFO, **s** para um socket.

Por padrão, o rótulo de tempo exibido é aquele da última modificação; as opções −c e −u selecionam outros dois rótulos de tempo. Para arquivos de dispositivos especiais o tamanho do campo é geralmente substituído pelos números de dispositivos maior e menor.

−q Mostra caracteres não imprimíveis no nome do arquivo como ponto de interrogação. (Isto é permitido como padrão para a saída em um terminal.)

−r Inverte a ordem do ordenação.

−t Ordena a exibição pelo rótulo de tempo.

−u Use o tempo do último acesso no lugar do tempo de modificação para ordenar (com −t) ou listar (com −l).

−1 Para saída em coluna simples.

DETALHES GNU

Se a saída padrão é um terminal, a saída é em colunas (ordenadas verticalmente).

dir (também instalado como **d**) é equivalente a 'ls −C −b'; isto é, arquivos listados em colunas, ordenados verticalmente. **vdir** (também instalado como **v**) é equivalente a 'ls −l −b'; isto é, arquivos listados no formato longo.

OPÇÕES GNU

−1, −−format=single−column
Lista um arquivo por linha. Isto é o padrão quando a saída padrão não é um terminal.

−a, −−all
Lista todos os arquivos nos diretórios, incluindo todos os arquivos começados com '.'.

−b, −−escape, −−quoting−style=escape
Coloca aspas em nomes de arquivos com caracteres não gráficos usando seqüencia de barra invertida alfabética e octal como usado em C. Esta opção é como a opção **−Q** exceto que os nome de arquivos não são colocados entre duas aspas.

−c, −−time=ctime, −−time=status
Ordena os conteúdos do diretório de acordo com os arquivos de status do tempo de alteração (o 'ctime' no inode). Se a listagem em formato longo é iniciada used (**−l**) mostra o status do tempo de alteração ao invés do tempo de modificação.

−d, −−directory
Lista nomes de diretórios como arquivos, ao invés de seus conteúdos.

−f Não ordena os conteúdos do diretório; lista-os na ordem que estão armazenados no disco. Também ativa **−a** e **−U** e desativa **−l**, **−−color**, **−s**, e **−t** se elas estavam especificadas antes de **−f**.

−g Ignorado; para compatibilidade com o Unix.

−h, −−human−readable
Anexa a letra de tamanho, por exemplo **M** para binários de megabytes ('mebibytes'), para cada tamanho. (Novo no Utilitários de Arquivo 4.0.)

−i, −−inode
Imprime o número inode (também chamado de número serial do arquivo e número índice) de cada arquivo no lado esquerdo de cada nome de arquivo. (Este número identifica unicamente cada arquivo dentro de um sistema de arquivos em particular).

−k, −−kilobytes
Se o tamanho do arquivos vai ser listado, imprime-os em kilobytes.

−l, −−format=long, −−format=verbose
Adicionalmente ao nome de cada arquivo, imprime o tipo de arquivo, permissão, número de ligações fortes, nome do proprietário, nome do grupo, tamanho em bytes, e rótulo de tempo (o tempo de modificação até que outros tempos sejam selecionados). Para arquivos com um tempo maior que 6 meses passados ou com mais de uma hora futura, o rótulo de tempo contém o ano ao invés do dia.

Para cada diretório que é listado, inicia o arquivo com uma linha *'total*de*blocos'* , onde " blocos " é o total de espaço de disco usado por todos arquivos no diretório. Por padrão, blocos de 1024

bytes são usados; se a variável de ambiente **POSIXLY_CORRECT** é selecionada, blocos de 512 bytes são usados (a menos que a opção **−k** seja *fornecida).*Os*blocos*computados*contam*cada*ligação*forte separadamente; isto é discutivelmente uma deficiência.

As permissões listadas são similares às especificações de modo simbólico mas **ls** combina multiplos bits no terceiro caractere de cada conjunto de permissões

s Se o bit setuid ou setgid e o bit correspondente para executáveis estão ambos selecionados.

S Se o bit setuid ou setgid são selecionados e o bit correspondente para executáveis não é selecionado.

t Se o bit contrário e o bit de outros executáveis estão ambos selecionados.

T Se o bit contrário é selecionado e o bit de outros executáveis não é selecionado.

x Se o bit para executáveis é selecionado e nada dos acima citados é aplicado.

− De outra forma.

−m, −−format=commas
Lista os arquivos horizontalmente, com muitos ajustes em cada linha, cada um separado por vírgula e um espaço.

−n, −−numeric−uid−gid
Lista a identificação numérica de usuário e de grupo ao invés dos nomes.

−o Produz listas em formato longo, mas não exibe informações de grupo. Isto é equivalente ao uso de **−−format=long −−no−group**. Esta opção é fornecida para compatibilidades com outras versões do **ls**.

−p, −−file−type, −−indicator−style=file−type
Anexa um caractere para cada nome de arquivo indicando o tipo de arquivo. Isto é como **−F** exceto que executáveis não são marcados. (Na realidade o Utilitários de Arquivo 4.0 trata a opção --file-type como --classify.)

−q, −−hide−control−chars
Imprime pontos de interrogação ao invs de caracteres não gráficos no nome de arquivo. Isto é o padrão.

−r, −−reverse
Ordena os conteúdos do diretório na ordem inversa.

−s, −−size
Imprime o tamanho de cada arquivo em blocos de 1024 bytes no lado esquerdo do nome de arquivo. Se a variável de ambiente **POSIXLY_CORRECT** é selecionada, blocos de 512 bytes são usadas ao invés de, a menos que a opção **−k** seja fornecida.

−t, −−sort=time
Ordena pelo tempo de modificação (o ’mtime’ no inode) ao invés de alfabeticamente , com o nome do arquivo mais recente listado primeiramente.

−u, −−time=atime, −−time=access, −−time=use
Ordena o conteúdo do diretório de acordo com tempo do último acesso do arquivo ao invés do tempo de modificação (o ’atime’ no inode). Se na listagem em formato longo é inicialmente usada, imprime o tempo do último accesso no lugar do tempo de modificação.

−v Ordena o conteúdo do diretório de acordo com a versão do arquivo. Isto leva em conta o fato de que nome de arquivos freqüentemente incluem índices ou números de versão. Funções padrão de ordenamento usualmente não produzem a ordem que o pessoal espera por causa da semelhança encontrada entre bases de caracteres. A versão ordena este problema, e é especialmente útil quando navegando por diretórios que contém muitos arquivos com números de índices/versão em

seus nomes. Por exemplo:

```
> ls -1          > ls -1v
foo.zml-1.gz     foo.zml-1.gz
foo.zml-100.gz    foo.zml-12.gz
foo.zml-12.gz     foo.zml-25.gz
foo.zml-25.gz     foo.zml-100.gz
```

Note também que partes numéricas são iniciadas com zeros e são consideradas como fracionária:

```
> ls -1          > ls -1v
abc-1.007.tgz    abc-1.007.tgz
abc-1.012b.tgz    abc-1.01a.tgz
abc-1.01a.tgz    abc-1.012b.tgz
```

(Novo no Utilitários de Arquivo 4.0.)

−w, −−width *cols*
Assume a tela com largura de *coluna* colunas. O padrão é dado pelo driver de terminal se possível; de outra forma a variável de ambiente **COLUMNS** é usada se estiver selecionada; de outra forma o padrão é 80.

−x, −−format=across, −−format=horizontal
Lista os arquivos em colunas, ordenados horizontalmente.

−A, −−almost−all
Lista todos os arquivos nos diretórios, exceto os ’.’ e ’..’.

−B, −−ignore−backups
Não lista arquivos que terminam com ’~’, a menos que sejam fornecidos na linha de comando.

−C, −−format=vertical
Lista os arquivos em colunas, ordenados verticalmente. Isto é o padrão se a saída padrão é um terminal. É sempre padrão para **dir** e **d**.

−D, −−dired
Com listagem em formato longo (**−l**) , imprime uma linha adicional depois da saída principal:
//DIRED// *BEG1 END1 BEG2 END2 ...*

Os *BEGn* and *ENDn* são inteiros sem sinal que registram o byte de posicionamento do início e do fim de cada um dos nomes de arquivos na saída. Isto facilita para o Emacs achar os nomes, até quando eles contém caracteres não usuais como espaços ou início de linha, sem pesquisa especial.

Se a listagem de diretórios é iniciada de forma recursiva (**−R**), sai uma linha parecida depois de cada sub-diretório:
//SUBDIRETÓRIO// *BEG1 END1 ...*

−F, −−classify, −−indicator−style=classify
Anexa um caractere para cada nome de arquivo indicando o tipo do arquivo. Para arquivos regulares que são executáveis, anexa um ’*’. O tipos de indicadores de arquivos são ’/’ para diretórios, ’@’ para ligações simbólicas, ’|’ para FIFOs,

−G, −−no−group
Inibe a exibição da informação do grupo na listagem de diretório em formato longo.

−H, −−si
Como em **−h**, mas usa a unidade oficial do SI (com potência de 1000 no lugar de 1024, de forma que M significa 1000000 ao invés de 1048576). (Novo no Utilitários de Arquivo 4.0.)

−I, −−ignore=*modelo*
Não lista arquivos com nomes combinando com o *modelo* do interpretador de comandos (não é expressão regular) a menos que eles sejam fornecidos na linha de comando. Como no interpretador de comando, um ’.’ inicial no nome do arquivo não é comparável com um caractere coringa no início do *modelo.*

−L, −−dereference
Lista a informação do arquivo correspondendo as ligações simbólicas referentes no lugar das próprias ligações.

−N, −−literal
Não coloca aspas no nome do arquivo.

−Q, −−quote−name, −−quoting−style=c
Confina o nome do arquivo em aspas duplas e os caracteres não gráficos entre aspas como no C.

−R, −−recursive
Lista o conteúdo de todos diretórios de forma recursiva.

−S, −−sort=size
Ordena o conteúdo do diretório pelo tamanho do arquivo no lugar de ordem alfabética, com os maiores arquivos listados primeiro.

−T, −−tabsize *cols*
Assume que cada marca de tabulação é uma largura de *cols* colunas. O padrão é 8 e pode ser sobrescrita pela variável de ambiente TABSIZE quando POSIXLY_CORRECT não está definido. **ls** usa a tabulação quando possível na saída, para eficiência. Se *cols* é zero, não se usa tabulação.

−U, −−sort=none
Não ordena o conteúdo do diretório; lista-os na ordem que estão armazenados no disco. (A diferença entre **−U** e **−f** é que o anterior não desativa ou ativa opções.) Isto é especialmente útil quando listamos diretórios muitos grandes, desde que não fazendo que qualquer ordenação possa ser notavelmente mais rápida.

−X, −−sort=extension
Ordena o conteúdo do diretório alfabeticamente pela extensão do arquivo (caractere depois do último ’.’); arquivos sem extensão são ordenados primeiramente.

−−block−size=*size*
Imprime o tamanho em blocos de *tamanho* bytes. (Novo no Utilitários de Arquivo 4.0.)

−−color[=*when***]**
Especifica a cor que será usada para distinguir os tipos de arquivo. Cores são especificadas usando a variável de ambiente LS_COLORS. Para informação sobre como selecionar esta variável, veja **dircolors**(1). *quando* pode ser omitido , ou um dos:

none Não usa cores. Isto é o padrão.

auto Somente usa cores se a saída padrão é um terminal.

always Sempre usa cores. Especificando **−−color** e não *when* é equivalente a **−−color=always**.

−−full−time
Lista o tempo completo, em detrimento a abreviação padrão. O formato é como no padrão **date**(1) ; não é possível alterar isto, mas você pode extrair a cadeia de caracteres da data com **cut**(1) e passar o resultado para ’date −d’.

Isto é muito útil por que o tempo na saída inclui os segundos. (O sistema de arquivo do Unix armazena o rótulo de tempo do arquivo somente para os segundos mais próximos, assim esta opção mostra todas as informações existentes). Por exemplo, isto pode ajudar quando você tem um Makefile que não regenera arquivos adequadamente.

−−quoting−style=*word*

Usa o estilo *word* para colocar aspas nos nomes da saída. A *word* pode ser uma das seguintes:

literal Saída de nomes como é. Isto é o comportamento padrão de **ls**.

shell Colocar aspas nos nomes para o interpretador de comandos se eles contém meta caracteres do interpretador de comandos ou que causaria saída ambígua.

shell−always
Coloca aspas nos nomes para o interpretador de comandos, mesmo se eles normalmente não requereriam aspas.

c Coloca aspas nos nomes como nas cadeias de caracteres da linguagem C; isto é igual a opção **−Q**

escape Coloca aspas como o *c* exceto que omite as aspas duplas ao redor; isto é igual a opção **−b**

Um valor padrão para esta opção pode ser especificada com a variável de ambiente QUOTING_STYLE. (Veja **AMBIENTE** abaixo.)

−−show−control−chars
Imprime caracteres não gráficos como no nome do arquivo. Isto é o padrão a menos que a saída seja um terminal e o programa seja **ls**.

OPÇÕES PADRÃO GNU

−−help Imprime a mensagem de uso na saída padrão e sai.

−−version
Imprime a versão na saída padrão e sai.

−− Encerra a lista de opção.

AMBIENTE

A variável POSIXLY_CORRECT determina a escolha da unidade. Se ela não é fixada, então a variável TABSIZE determina o número de caracteres por tabulação. A variável COLUMNS (quando contém a representação de um decimal inteiro) determina a largura da coluna de saída (para usar com a opção −C). O nome do arquivos não devem ser truncados para torná-los adequados à saída em múltiplas colunas.

As variáveis LANG, LC_ALL, LC_CTYPE e LC_MESSAGES têm seus significados usuais. A variável TZ fornece a zona de tempo para a cadeia de caracteres de tempo escrita por **ls**. A variável LS_COLORS é usada para especificar as cores usadas.

A variável QUOTING_STYLE é usada para especificar o valor padrão para a opção **−−quoting−style literal**, embora os autores advertissem que este padrão pode mudar para **shell** em qualquer versão futura de **ls**.

PROBLEMAS

Em sistemas BSD, a opção **−s** relata tamanhos que são a metade dos valores corretos para arquivos que são montados de sistemas HP-UX via NFS. Em sistemas HP-UX, **ls** relata tamanhos que são duas vezes maiores que os valores corretos para arquivos que são montados de sistemas BSD via NFS. Isto é devido a uma falha no HP-UX; e também afeta o programa **ls** do HP-UX.

DE ACORDO COM

POSIX 1003.2

VEJA TAMBÉM

dircolors(1)

NOTAS

Esta página descreve **ls** como é encontrada no pacote Utilitários de Arquivo 4.0; outras versões podem ser um pouco diferentes. Envie correções e adições para aeb@cwi.nl. Relatório de problemas no programa para fileutils-bugs@gnu.ai.mit.edu.

TRADUZIDO POR LDP-BR em 21/08/2000.

André L. Fassone Canova <lonelywolf@blv.com.br> (tradução) Roberto Selbach Teixeira <robteix@zaz.com.br> (revisão)

NAME

man − an interface to the on-line reference manuals

SYNOPSIS

man [**−C** *file*] [**−d**] [**−D**] [**−−warnings** [=*warnings*]] [**−R** *encoding*] [**−L** *locale*] [**−m** *system* [,...]] [**−M** *path*] [**−S** *list*] [**−e** *extension*] [**−i** | **−I**] [**−−regex** | **−−wildcard**] [**−−names−only**] [**−a**] [**−u**] [**−−no−subpages**] [**−P** *pager*] [**−r** *prompt*] [**−7**] [**−E** *encoding*] [**−−no−hyphenation**] [**−−no−justification**] [**−p** *string*] [**−t**] [**−T** [*device*]] [**−H** [*browser*]] [**−X** [*dpi*]] [**−Z**] [[*section*] *page*[. *section*] ...] ...
man −k [*apropos options*] *regexp* ...
man −K [**−w** | **−W**] [**−S** *list*] [**−i** | **−I**] [**−−regex**] [*section*] *term* ...
man −f [*whatis options*] *page* ...
man −l [**−C** *file*] [**−d**] [**−D**] [**−−warnings** [=*warnings*]] [**−R** *encoding*] [**−L** *locale*] [**−P** *pager*] [**−r** *prompt*] [**−7**] [**−E** *encoding*] [**−p** *string*] [**−t**] [**−T** [*device*]] [**−H** [*browser*]] [**−X** [*dpi*]] [**−Z**] *file* ...
man −w | **−W** [**−C** *file*] [**−d**] [**−D**] *page* ...
man −c [**−C** *file*] [**−d**] [**−D**] *page* ...
man [**−?V**]

DESCRIPTION

man is the system's manual pager. Each *page* argument given to **man** is normally the name of a program, utility or function. The *manual page* associated with each of these arguments is then found and displayed. A *section*, if provided, will direct **man** to look only in that *section* of the manual. The default action is to search in all of the available *sections* following a pre-defined order ("1 n l 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7" by default, unless overridden by the **SECTION** directive in */etc/manpath.config*), and to show only the first *page* found, even if *page* exists in several *sections*.

The table below shows the *section* numbers of the manual followed by the types of pages they contain.

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in */dev*)
5 File formats and conventions eg */etc/passwd*
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. **man**(7), **groff**(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

A manual *page* consists of several sections.

Conventional section names include **NAME**, **SYNOPSIS**, **CONFIGURATION**, **DESCRIPTION**, **OPTIONS**, **EXIT STATUS**, **RETURN VALUE**, **ERRORS**, **ENVIRONMENT**, **FILES**, **VERSIONS**, **CONFORMING TO**, **NOTES**, **BUGS**, **EXAMPLE**, **AUTHORS**, and **SEE ALSO**.

The following conventions apply to the **SYNOPSIS** section and can be used as a guide in other sections.

bold text	type exactly as shown.
italic text	replace with appropriate argument.
[**−abc**]	any or all arguments within [ ] are optional.
−a | **−b**	options delimited by | cannot be used together.
argument ...	*argument* is repeatable.
[*expression*] ...	entire *expression* within [ ] is repeatable.

Exact rendering may vary depending on the output device. For instance, man will usually not be able to render italics when running in a terminal, and will typically use underlined or coloured text instead.

The command or function illustration is a pattern that should match all possible invocations. In some cases it is advisable to illustrate several exclusive invocations as is shown in the **SYNOPSIS** section of this manual page.

EXAMPLES

man *ls*
Display the manual page for the *item* (program) *ls*.

man *man.7*
Display the manual page for macro package *man* from section *7*.

man −a *intro*
Display, in succession, all of the available *intro* manual pages contained within the manual. It is possible to quit between successive displays or skip any of them.

man −t *alias | lpr −Pps*
Format the manual page referenced by '*alias*', usually a shell manual page, into the default **troff** or **groff** format and pipe it to the printer named *ps*. The default output for **groff** is usually PostScript. **man −−help** should advise as to which processor is bound to the **−t** option.

man −l −T*dvi ./foo.1x.gz* > *./foo.1x.dvi*
This command will decompress and format the nroff source manual page *./foo.1x.gz* into a **device independent (dvi)** file. The redirection is necessary as the **−T** flag causes output to be directed to **stdout** with no pager. The output could be viewed with a program such as **xdvi** or further processed into PostScript using a program such as **dvips.**

man −k *printf*
Search the short descriptions and manual page names for the keyword *printf* as regular expression. Print out any matches. Equivalent to **apropos** *printf* **.**

man −f *smail*
Lookup the manual pages referenced by *smail* and print out the short descriptions of any found. Equivalent to **whatis** *smail***.**

OVERVIEW

Many options are available to **man** in order to give as much flexibility as possible to the user. Changes can be made to the search path, section order, output processor, and other behaviours and operations detailed below.

If set, various environment variables are interrogated to determine the operation of **man**. It is possible to set the 'catch all' variable $**MANOPT** to any string in command line format with the exception that any spaces used as part of an option's argument must be escaped (preceded by a backslash). **man** will parse $**MANOPT** prior to parsing its own command line. Those options requiring an argument will be overridden by the same options found on the command line. To reset all of the options set in $**MANOPT**, **−D** can be specified as the initial command line option. This will allow man to 'forget' about the options specified in $**MANOPT** although they must still have been valid.

The manual pager utilities packaged as **man-db** make extensive use of **index** database caches. These caches contain information such as where each manual page can be found on the filesystem and what its *whatis* (short one line description of the man page) contains, and allow **man** to run faster than if it had to search the filesystem each time to find the appropriate manual page. If requested using the **−u** option, **man** will ensure that the caches remain consistent, which can obviate the need to manually run software to update traditional *whatis* text databases.

If **man** cannot find a **mandb** initiated **index** database for a particular manual page hierarchy, it will still search for the requested manual pages, although file globbing will be necessary to search within that hierarchy. If **whatis** or **apropos** fails to find an **index** it will try to extract information from a traditional *whatis* database instead.

These utilities support compressed source nroff files having, by default, the extensions of **.Z**, **.z** and **.gz**. It is possible to deal with any compression extension, but this information must be known at compile time. Also, by default, any cat pages produced are compressed using **gzip**. Each 'global' manual page hierarchy such as */usr/share/man* or */usr/X11R6/man* may have any directory as its cat page hierarchy. Traditionally the cat pages are stored under the same hierarchy as the man pages, but for reasons such as those specified in the **File Hierarchy Standard (FHS)**, it may be better to store them elsewhere. For details on how to do this, please read **manpath**(5). For details on why to do this, read the standard.

International support is available with this package. Native language manual pages are accessible (if available on your system) via use of *locale* functions. To activate such support, it is necessary to set either $**LC_MESSAGES**, $**LANG** or another system dependent environment variable to your language locale, usually specified in the **POSIX 1003.1** based format:

<*language*>[**_**<*territory*>[**.**<*character-set*>[**,**<*version*>]]]

If the desired page is available in your *locale*, it will be displayed in lieu of the standard (usually American English) page.

Support for international message catalogues is also featured in this package and can be activated in the same way, again if available. If you find that the manual pages and message catalogues supplied with this package are not available in your native language and you would like to supply them, please contact the maintainer who will be coordinating such activity.

For information regarding other features and extensions available with this manual pager, please read the documents supplied with the package.

DEFAULTS

man will search for the desired manual pages within the *index* database caches. If the **−u** option is given, a cache consistency check is performed to ensure the databases accurately reflect the filesystem. If this option is always given, it is not generally necessary to run **mandb** after the caches are initially created, unless a cache becomes corrupt. However, the cache consistency check can be slow on systems with many manual pages installed, so it is not performed by default, and system administrators may wish to run **mandb** every week or so to keep the database caches fresh. To forestall problems caused by outdated caches, **man** will fall back to file globbing if a cache lookup fails, just as it would if no cache was present.

Once a manual page has been located, a check is performed to find out if a relative preformatted 'cat' file already exists and is newer than the nroff file. If it does and is, this preformatted file is (usually) decompressed and then displayed, via use of a pager. The pager can be specified in a number of ways, or else will fall back to a default is used (see option **−P** for details). If no cat is found or is older than the nroff file, the nroff is filtered through various programs and is shown immediately.

If a cat file can be produced (a relative cat directory exists and has appropriate permissions), **man** will compress and store the cat file in the background.

The filters are deciphered by a number of means. Firstly, the command line option **−p** or the environment variable $**MANROFFSEQ** is interrogated. If **−p** was not used and the environment variable was not set, the initial line of the nroff file is parsed for a preprocessor string. To contain a valid preprocessor string, the first line must resemble

'\" <**string**>

where **string** can be any combination of letters described by option **−p** below.

If none of the above methods provide any filter information, a default set is used.

A formatting pipeline is formed from the filters and the primary formatter (**nroff** or [**tg**]**roff** with **−t**) and executed. Alternatively, if an executable program *mandb_nfmt* (or *mandb_tfmt* with **−t**) exists in the man tree root, it is executed instead. It gets passed the manual source file, the preprocessor string, and optionally the device specified with **−T** or **−E** as arguments.

OPTIONS

Non argument options that are duplicated either on the command line, in $**MANOPT**, or both, are not harmful. For options that require an argument, each duplication will override the previous argument value.

General options

−C *file*, **−−config−file=** *file*
: Use this user configuration file rather than the default of *~/.manpath*.

−d, **−−debug**
: Print debugging information.

−D, **−−default**
: This option is normally issued as the very first option and resets **man's** behaviour to its default. Its use is to reset those options that may have been set in $**MANOPT**. Any options that follow **−D** will have their usual effect.

−−warnings[=*warnings*]
: Enable warnings from *groff*. This may be used to perform sanity checks on the source text of manual pages. *warnings* is a comma-separated list of warning names; if it is not supplied, the default is "mac". See the "Warnings" node in **info groff** for a list of available warning names.

Main modes of operation

−f, **−−whatis**
: Equivalent to **whatis**. Display a short description from the manual page, if available. See **whatis**(1) for details.

−k, **−−apropos**
: Equivalent to **apropos**. Search the short manual page descriptions for keywords and display any matches. See **apropos**(1) for details.

−K, **−−global−apropos**
: Search for text in all manual pages. This is a brute-force search, and is likely to take some time; if you can, you should specify a section to reduce the number of pages that need to be searched. Search terms may be simple strings (the default), or regular expressions if the **−−regex** option is used.

  Note that this searches the *sources* of the manual pages, not the rendered text, and so may include false positives due to things like comments in source files. Searching the rendered text would be much slower.

−l, **−−local−file**
: Activate 'local' mode. Format and display local manual files instead of searching through the system's manual collection. Each manual page argument will be interpreted as an nroff source file in the correct format. No cat file is produced. If '−' is listed as one of the arguments, input will be taken from stdin. When this option is not used, and man fails to find the page required, before displaying the error message, it attempts to act as if this option was supplied, using the name as a filename and looking for an exact match.

−w, **−−where**, **−−path**, **−−location**
: Don't actually display the manual pages, but do print the location(s) of the source nroff files that would be formatted.

−W, **−−where−cat**, **−−location−cat**
: Don't actually display the manual pages, but do print the location(s) of the cat files that would be displayed. If −w and −W are both specified, print both separated by a space.

−c, **−−catman**
This option is not for general use and should only be used by the **catman** program.

−R *encoding*, **−−recode**=*encoding*
Instead of formatting the manual page in the usual way, output its source converted to the specified *encoding*. If you already know the encoding of the source file, you can also use **manconv**(1) directly. However, this option allows you to convert several manual pages to a single encoding without having to explicitly state the encoding of each, provided that they were already installed in a structure similar to a manual page hierarchy.

Finding manual pages

−L *locale*, **−−locale**=*locale*
man will normally determine your current locale by a call to the C function **setlocale**(3) which interrogates various environment variables, possibly including $**LC_MESSAGES** and $**LANG**. To temporarily override the determined value, use this option to supply a *locale* string directly to **man**. Note that it will not take effect until the search for pages actually begins. Output such as the help message will always be displayed in the initially determined locale.

−m *system*[,...], **−−systems**=*system*[,...]
If this system has access to other operating system's manual pages, they can be accessed using this option. To search for a manual page from NewOS's manual page collection, use the option **−m NewOS**.

The *system* specified can be a combination of comma delimited operating system names. To include a search of the native operating system's manual pages, include the system name **man** in the argument string. This option will override the $**SYSTEM** environment variable.

−M *path*, **−−manpath**=*path*
Specify an alternate manpath to use. By default, **man** uses **manpath** derived code to determine the path to search. This option overrides the $**MANPATH** environment variable and causes option **−m** to be ignored.

A path specified as a manpath must be the root of a manual page hierarchy structured into sections as described in the man-db manual (under "The manual page system"). To view manual pages outside such hierarchies, see the **−l** option.

−S *list*, **−s** *list*, **−−sections**=*list*
List is a colon- or comma-separated list of 'order specific' manual sections to search. This option overrides the $**MANSECT** environment variable. (The **−s** spelling is for compatibility with System V.)

−e *sub-extension*, **−−extension**=*sub-extension*
Some systems incorporate large packages of manual pages, such as those that accompany the **Tcl** package, into the main manual page hierarchy. To get around the problem of having two manual pages with the same name such as **exit**(3), the **Tcl** pages were usually all assigned to section **l**. As this is unfortunate, it is now possible to put the pages in the correct section, and to assign a specific 'extension' to them, in this case, **exit**(3tcl). Under normal operation, **man** will display **exit**(3) in preference to **exit**(3tcl). To negotiate this situation and to avoid having to know which section the page you require resides in, it is now possible to give **man** a *sub-extension* string indicating which package the page must belong to. Using the above example, supplying the option **−e tcl** to **man** will restrict the search to pages having an extension of ***tcl**.

−i, **−−ignore−case**
Ignore case when searching for manual pages. This is the default.

−I, **−−match−case**
Search for manual pages case-sensitively.

−−regex
Show all pages with any part of either their names or their descriptions matching each *page* argument as a regular expression, as with **apropos**(1). Since there is usually no reasonable way to pick a "best" page when searching for a regular expression, this option implies **−a**.

−−wildcard
Show all pages with any part of either their names or their descriptions matching each *page* argument using shell-style wildcards, as with **apropos**(1) **−−wildcard**. The *page* argument must match the entire name or description, or match on word boundaries in the description. Since there is usually no reasonable way to pick a "best" page when searching for a wildcard, this option implies **−a**.

−−names−only
If the **−−regex** or **−−wildcard** option is used, match only page names, not page descriptions, as with **whatis**(1). Otherwise, no effect.

−a, **−−all**
By default, **man** will exit after displaying the most suitable manual page it finds. Using this option forces **man** to display all the manual pages with names that match the search criteria.

−u, **−−update**
This option causes **man** to perform an ‘inode level’ consistency check on its database caches to ensure that they are an accurate representation of the filesystem. It will only have a useful effect if **man** is installed with the setuid bit set.

−−no−subpages
By default, **man** will try to interpret pairs of manual page names given on the command line as equivalent to a single manual page name containing a hyphen or an underscore. This supports the common pattern of programs that implement a number of subcommands, allowing them to provide manual pages for each that can be accessed using similar syntax as would be used to invoke the subcommands themselves. For example:

```
$ man −aw git diff
/usr/share/man/man1/git−diff.1.gz
```

To disable this behaviour, use the **−−no−subpages** option.

```
$ man −aw −−no−subpages git diff
/usr/share/man/man1/git.1.gz
/usr/share/man/man3/Git.3pm.gz
/usr/share/man/man1/diff.1.gz
```

Controlling formatted output

−P *pager*, **−−pager=***pager*
Specify which output pager to use. By default, **man** uses **pager**. This option overrides the $**MANPAGER** environment variable, which in turn overrides the $**PAGER** environment variable. It is not used in conjunction with **−f** or **−k**.

The value may be a simple command name or a command with arguments, and may use shell quoting (backslashes, single quotes, or double quotes). It may not use pipes to connect multiple commands; if you need that, use a wrapper script, which may take the file to display either as an argument or on standard input.

−r *prompt*, **−−prompt=***prompt*
If a recent version of **less** is used as the pager, **man** will attempt to set its prompt and some sensible options. The default prompt looks like

Manual page *name***(**_sec_**) line** *x*

where *name* denotes the manual page name, *sec* denotes the section it was found under and *x* the current line number. This is achieved by using the $**LESS** environment variable.

Supplying **−r** with a string will override this default. The string may contain the text **$MAN_PN** which will be expanded to the name of the current manual page and its section name surrounded by '(' and ')'. The string used to produce the default could be expressed as

\ Manual\ page\ \$MAN_PN\ ?ltline\ %lt?L/%L.:
byte\ %bB?s/%s..?\ (END):?pB\ %pB\\%..
(press h for help or q to quit)

It is broken into three lines here for the sake of readability only. For its meaning see the **less**(1) manual page. The prompt string is first evaluated by the shell. All double quotes, back-quotes and backslashes in the prompt must be escaped by a preceding backslash. The prompt string may end in an escaped $ which may be followed by further options for less. By default **man** sets the **−ix8** options.

The $**MANLESS** environment variable described below may be used to set a default prompt string if none is supplied on the command line.

−7, **−−ascii**
When viewing a pure *ascii*(7) manual page on a 7 bit terminal or terminal emulator, some characters may not display correctly when using the *latin1*(7) device description with **GNU nroff**. This option allows pure *ascii* manual pages to be displayed in *ascii* with the *latin1* device. It will not translate any *latin1* text. The following table shows the translations performed: some parts of it may only be displayed properly when using **GNU nroff**'s *latin1*(7) device.

Description	Octal	latin1	ascii
continuation hyphen	255	-	-
bullet (middle dot)	267	•	o
acute accent	264	´	'
multiplication sign	327	×	x

If the *latin1* column displays correctly, your terminal may be set up for *latin1* characters and this option is not necessary. If the *latin1* and *ascii* columns are identical, you are reading this page using this option or **man** did not format this page using the *latin1* device description. If the *latin1* column is missing or corrupt, you may need to view manual pages with this option.

This option is ignored when using options **−t**, **−H**, **−T**, or **−Z** and may be useless for **nroff** other than **GNU's**.

−E *encoding*, **−−encoding**=*encoding*
Generate output for a character encoding other than the default. For backward compatibility, *encoding* may be an **nroff** device such as **ascii**, **latin1**, or **utf8** as well as a true character encoding such as **UTF−8**.

−−no−hyphenation, **−−nh**
Normally, **nroff** will automatically hyphenate text at line breaks even in words that do not contain hyphens, if it is necessary to do so to lay out words on a line without excessive spacing. This option disables automatic hyphenation, so words will only be hyphenated if they already contain hyphens.

If you are writing a manual page and simply want to prevent **nroff** from hyphenating a word at an inappropriate point, do not use this option, but consult the **nroff** documentation instead; for instance, you can put "\%" inside a word to indicate that it may be hyphenated at that point, or put "\%" at the start of a word to prevent it from being hyphenated.

−−no−justification, **−−nj**
Normally, **nroff** will automatically justify text to both margins. This option disables full justification, leaving justified only to the left margin, sometimes called "ragged-right" text.

If you are writing a manual page and simply want to prevent **nroff** from justifying certain paragraphs, do not use this option, but consult the **nroff** documentation instead; for instance, you can use the ".na", ".nf", ".fi", and ".ad" requests to temporarily disable adjusting and filling.

−p *string*, **−−preprocessor=***string*
Specify the sequence of preprocessors to run before **nroff** or **troff**/**groff**. Not all installations will have a full set of preprocessors. Some of the preprocessors and the letters used to designate them are: **eqn** (**e**), **grap** (**g**), **pic** (**p**), **tbl** (**t**), **vgrind** (**v**), **refer** (**r**). This option overrides the $**MANROFFSEQ** environment variable. **zsoelim** is always run as the very first preprocessor.

−t, **−−troff**
Use *groff -mandoc* to format the manual page to stdout. This option is not required in conjunction with **−H**, **−T**, or **−Z**.

−T[*device*], **−−troff−device**[=*device*]
This option is used to change **groff** (or possibly **troff's**) output to be suitable for a device other than the default. It implies **−t**. Examples (provided with Groff-1.17) include **dvi**, **latin1**, **ps**, **utf8**, **X75** and **X100**.

−H[*browser*], **−−html**[=*browser*]
This option will cause **groff** to produce HTML output, and will display that output in a web browser. The choice of browser is determined by the optional *browser* argument if one is provided, by the $**BROWSER** environment variable, or by a compile-time default if that is unset (usually **lynx**). This option implies **−t**, and will only work with **GNU troff**.

−X[*dpi*], **−−gxditview**[=*dpi*]
This option displays the output of **groff** in a graphical window using the **gxditview** program. The *dpi* (dots per inch) may be 75, 75-12, 100, or 100-12, defaulting to 75; the -12 variants use a 12-point base font. This option implies **−T** with the X75, X75-12, X100, or X100-12 device respectively.

−Z, **−−ditroff**
groff will run **troff** and then use an appropriate post-processor to produce output suitable for the chosen device. If *groff -mandoc* is **groff**, this option is passed to **groff** and will suppress the use of a post-processor. It implies **−t**.

Getting help

−?, **−−help**
Print a help message and exit.

−−usage
Print a short usage message and exit.

−V, **−−version**
Display version information.

EXIT STATUS

0 Successful program execution.

1 Usage, syntax or configuration file error.

2 Operational error.

3 A child process returned a non-zero exit status.

16 At least one of the pages/files/keywords didn't exist or wasn't matched.

ENVIRONMENT

MANPATH

If $**MANPATH** is set, its value is used as the path to search for manual pages.

MANROFFOPT

The contents of $**MANROFFOPT** are added to the command line every time **man** invokes the formatter (**nroff**, **troff**, or **groff**).

MANROFFSEQ

If $**MANROFFSEQ** is set, its value is used to determine the set of preprocessors to pass each manual page through. The default preprocessor list is system dependent.

MANSECT

If $**MANSECT** is set, its value is a colon-delimited list of sections and it is used to determine which manual sections to search and in what order. The default is "1 n l 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7", unless overridden by the **SECTION** directive in */etc/manpath.config*.

MANPAGER, **PAGER**

If $**MANPAGER** or $**PAGER** is set ($**MANPAGER** is used in preference), its value is used as the name of the program used to display the manual page. By default, **pager** is used.

The value may be a simple command name or a command with arguments, and may use shell quoting (backslashes, single quotes, or double quotes). It may not use pipes to connect multiple commands; if you need that, use a wrapper script, which may take the file to display either as an argument or on standard input.

MANLESS

If $**MANLESS** is set, its value will be used as the default prompt string for the **less** pager, as if it had been passed using the **−r** option (so any occurrences of the text **$MAN_PN** will be expanded in the same way). For example, if you want to set the prompt string unconditionally to "my prompt string", set $**MANLESS** to '**−Psmy prompt string**'. Using the **−r** option overrides this environment variable.

BROWSER

If $**BROWSER** is set, its value is a colon-delimited list of commands, each of which in turn is used to try to start a web browser for **man −−html**. In each command, *%s* is replaced by a filename containing the HTML output from **groff**, *%%* is replaced by a single percent sign (%), and *%c* is replaced by a colon (:).

SYSTEM

If $**SYSTEM** is set, it will have the same effect as if it had been specified as the argument to the **−m** option.

MANOPT

If $**MANOPT** is set, it will be parsed prior to **man's** command line and is expected to be in a similar format. As all of the other **man** specific environment variables can be expressed as command line options, and are thus candidates for being included in $**MANOPT** it is expected that they will become obsolete. N.B. All spaces that should be interpreted as part of an option's argument must be escaped.

MANWIDTH

If $**MANWIDTH** is set, its value is used as the line length for which manual pages should be formatted. If it is not set, manual pages will be formatted with a line length appropriate to the current terminal (using the value of $**COLUMNS**, an **ioctl**(2) if available, or falling back to 80 characters if neither is available). Cat pages will only be saved when the default formatting can be used, that is when the terminal line length is between 66 and 80 characters.

MAN_KEEP_FORMATTING

Normally, when output is not being directed to a terminal (such as to a file or a pipe), formatting characters are discarded to make it easier to read the result without special tools. However, if

$**MAN_KEEP_FORMATTING** is set to any non-empty value, these formatting characters are retained. This may be useful for wrappers around **man** that can interpret formatting characters.

MAN_KEEP_STDERR

Normally, when output is being directed to a terminal (usually to a pager), any error output from the command used to produce formatted versions of manual pages is discarded to avoid interfering with the pager's display. Programs such as **groff** often produce relatively minor error messages about typographical problems such as poor alignment, which are unsightly and generally confusing when displayed along with the manual page. However, some users want to see them anyway, so, if $**MAN_KEEP_STDERR** is set to any non-empty value, error output will be displayed as usual.

LANG, **LC_MESSAGES**

Depending on system and implementation, either or both of $**LANG** and $**LC_MESSAGES** will be interrogated for the current message locale. **man** will display its messages in that locale (if available). See **setlocale**(3) for precise details.

FILES

/etc/manpath.config

man-db configuration file.

/usr/share/man

A global manual page hierarchy.

/usr/share/man/index.(bt|db|dir|pag)

A traditional global *index* database cache.

/var/cache/man/index.(bt|db|dir|pag)

An FHS compliant global *index* database cache.

SEE ALSO

apropos(1), **groff**(1), **less**(1), **manpath**(1), **nroff**(1), **troff**(1), **whatis**(1), **zsoelim**(1), **setlocale**(3), **manpath**(5), **ascii**(7), **latin1**(7), **man**(7), **catman**(8), **mandb**(8), the man-db package manual, **FSSTND**

HISTORY

1990, 1991 – Originally written by John W. Eaton (jwe@che.utexas.edu).

Dec 23 1992: Rik Faith (faith@cs.unc.edu) applied bug fixes supplied by Willem Kasdorp (wkasdo@nikhefk.nikef.nl).

30th April 1994 – 23rd February 2000: Wilf. (G.Wilford@ee.surrey.ac.uk) has been developing and maintaining this package with the help of a few dedicated people.

30th October 1996 – 30th March 2001: Fabrizio Polacco <fpolacco@debian.org> maintained and enhanced this package for the Debian project, with the help of all the community.

31st March 2001 – present day: Colin Watson <cjwatson@debian.org> is now developing and maintaining man-db.

NOME

mkdir − cria diretórios

SINOPSE

mkdir [*opções***]** *diretório...*

Opções POSIX: **[−p] [−m** *modo***]**

Opções GNU (forma reduzida): **[−p] [−m** *modo***] [−−verbose] [−−help] [−−version] [−−]**

DESCRIÇÃO

mkdir cria diretório com os nomes especificados.

Por padrão, o modo de criação dos diretórios é 0777 ("a+rwx") menos os bits selecionados na umask.

OPÇÕES

−m *modo*, **−−mode=***modo*

Seleciona o modo de criação dos diretórios para *modo*, que pode ser simbólico como em **chmod**(1) e então usa o modo padrão como ponto de partida.

−p, −−parents

Cria qualquer diretório pai faltante para cada argumento *diretório* para diretórios pais é fixo pela umask modificada por "u+wx". Ignora argumentos que correspondem a diretórios existentes. (Assim, se um diretório /a existe, então "mkdir /a" vai provocar um erro, mas "mkdir −p /a" não.)

−−verbose

Imprime uma mensagem para cada diretório criado. Isto é muito útil com **−−parents**.

OPÇÕES PADRÃO GNU

−−help Imprime a mensagem de uso na saída padrão e sai.

−−version

Imprime a versão na saída padrão e sai.

−− Encerra a lista de opção.

AMBIENTE

As variáveis LANG, LC_ALL, LC_CTYPE and LC_MESSAGES tem seu significado usual.

DE ACORDO COM

POSIX 1003.2

NOTAS

Esta página descreve **mkdir** como é encontrado no pacote Utilitários de Arquivos 4.0; outras versões podem ser um pouco diferentes. Envie correções e adições para aeb@cwi.nl. Relatório de problemas no programa para fileutils-bugs@gnu.ai.mit.edu.

TRADUZIDO POR LDP-BR em 21/08/2000.

André L. Fassone Canova <lonelywolf@blv.com.br> (tradução) Ricardo C.O. Freitas <english.quest@best-service.com> (revisão)

NAME

nano − Nano's ANOther editor, an enhanced free Pico clone

SYNOPSIS

nano [*options*] [[+*line*[,*column*]] *file*]...

DESCRIPTION

nano is a small, free and friendly editor which aims to replace Pico, the default editor included in the non-free Pine package. On top of copying Pico's look and feel, **nano** also implements some missing (or disabled by default) features in Pico, such as "search and replace" and "go to line and column number".

EDITING

Entering text and moving around in a file is straightforward: typing the letters and using the normal cursor movement keys. Commands are entered by using the Control (^) and the Alt or Meta (M−) keys. Typing **^K** deletes the current line and puts it in the cutbuffer. Consecutive **^K**s will put all deleted lines together in the cutbuffer. Any cursor movement or executing any other command will cause the next **^K** to overwrite the cutbuffer. A **^U** will paste the current contents of the cutbuffer at the current cursor position.

When a more precise piece of text needs to be cut or copied, one can mark its start with **^6**, move the cursor to its end (the marked text will be highlighted), and then use **^K** to cut it, or **M−6** to copy it to the cutbuffer. One can also save the marked text to a file with **^O**, or spell check it with **^T**.

Since nano-2.7.0, text can also be selected by holding Shift and moving the cursor with the arrow keys. Holding down the Alt key too will increase the stride.

The two lines at the bottom of the screen show the most important commands; the built-in help (**^G**) lists all the available ones. The default key bindings can be changed via the .nanorc file -- see **nanorc**(5).

OPTIONS

+line*,*column

Places the cursor on line number *line* and at column number *column* (at least one of which must be specified) on startup, instead of the default line 1, column 1.

−A, **−−smarthome**

Make the Home key smarter. When Home is pressed anywhere but at the very beginning of non-whitespace characters on a line, the cursor will jump to that beginning (either forwards or backwards). If the cursor is already at that position, it will jump to the true beginning of the line.

−B, **−−backup**

When saving a file, back up the previous version of it, using the current filename suffixed with a tilde (~).

−C *directory*, **−−backupdir=***directory*

Make and keep not just one backup file, but make and keep a uniquely numbered one every time a file is saved -- when backups are enabled. The uniquely numbered files are stored in the specified *directory*.

−D, **−−boldtext**

Use bold text instead of reverse video text.

−E, **−−tabstospaces**

Convert typed tabs to spaces.

−F, **−−multibuffer**

Enable multiple file buffers (if support for them has been compiled in).

−G, **−−locking**

Enable vim-style file locking when editing files.

−H, **−−historylog**
Log search and replace strings to *~/.nano/search_history*, so they can be retrieved in later sessions.

−I, **−−ignorercfiles**
Don’t look at the system’s **nanorc** nor at **~/.nanorc**.

−K, **−−rebindkeypad**
Interpret the numeric keypad keys so that they all work properly. You should only need to use this option if they don’t, as mouse support won’t work properly with this option enabled.

−L, **−−nonewlines**
Don’t add newlines to the ends of files.

−N, **−−noconvert**
Disable automatic conversion of files from DOS/Mac format.

−O, **−−morespace**
Use the blank line below the titlebar as extra editing space.

−P, **−−positionlog**
For the 200 most recent files, log the last position of the cursor, and place it at that position again upon reopening such a file. (The old form of this option, **−−poslog**, is deprecated.)

−Q "*characters*", **−−quotestr=**"*characters*"
Set the quoting string for justifying. The default is **"^([\t]*[#:>\|}])+"** if extended regular expression support is available, or **"> "** otherwise. Note that **\t** stands for a Tab.

−R, **−−restricted**
Restricted mode: don’t read or write to any file not specified on the command line; don’t read any *nanorc* files nor history files; don’t allow suspending nor spell checking; don’t allow a file to be appended to, prepended to, or saved under a different name if it already has one; and don’t use backup files. This restricted mode is also accessible by invoking **nano** with any name beginning with ’r’ (e.g. "rnano").

−S, **−−smooth**
Enable smooth scrolling. Text will scroll line-by-line, instead of the usual chunk-by-chunk behavior.

−T *number*, **−−tabsize=***number*
Set the size (width) of a tab to *number* columns. The value of *number* must be greater than 0. The default value is 8.

−U, **−−quickblank**
Do quick statusbar blanking. Statusbar messages will disappear after 1 keystroke instead of 25. Note that **−c** overrides this.

−V, **−−version**
Show the current version number and exit.

−W, **−−wordbounds**
Detect word boundaries differently by treating punctuation characters as part of a word.

−X "*characters*", **−−wordchars=**"*characters*"
Specify which other characters (besides the normal alphanumeric ones) should be considered as part of a word. This overrides option **−W** (**−−wordbounds**).

−Y *name*, **−−syntax=***name*
Specify the name of the syntax highlighting to use from among the ones defined in the *nanorc* files.

−c, **−−constantshow**
Constantly show the cursor position. Note that this overrides **−U**.

−d, **−−rebinddelete**
Interpret the Delete key differently so that both Backspace and Delete work properly. You should only need to use this option if Backspace acts like Delete on your system.

−g, **−−showcursor**
Make the cursor visible in the file browser, putting it on the highlighted item. Useful for braille users.

−h, **−−help**
Show a summary of the available command-line options and exit.

−i, **−−autoindent**
Indent new lines to the previous line's indentation. Useful when editing source code.

−k, **−−cut**
Make the 'Cut Text' command (normally **^K**) cut from the current cursor position to the end of the line, instead of cutting the entire line.

−l, **−−linenumbers**
Display line numbers to the left of the text area.

−m, **−−mouse**
Enable mouse support, if available for your system. When enabled, mouse clicks can be used to place the cursor, set the mark (with a double click), and execute shortcuts. The mouse will work in the X Window System, and on the console when gpm is running. Text can still be selected through dragging by holding down the Shift key.

−n, **−−noread**
Treat any name given on the command line as a new file. This allows **nano** to write to named pipes: it will start with a blank buffer, and will write to the pipe when the user saves the "file". This way **nano** can be used as an editor in combination with for instance **gpg** without having to write sensitive data to disk first.

−o *directory*, **−−operatingdir=***directory*
Set the operating directory. This makes **nano** set up something similar to a chroot.

−p, **−−preserve**
Preserve the XON and XOFF sequences (^Q and ^S) so they will be caught by the terminal.

−q, **−−quiet**
Do not report errors in the *nanorc* files nor ask them to be acknowledged by pressing Enter at startup.

−r *number*, **−−fill=***number*
Hard-wrap lines at column *number*. If this value is 0 or less, wrapping will occur at the width of the screen less *number* columns, allowing the wrap point to vary along with the width of the screen if the screen is resized. The default value is −8. This option conflicts with **−w** -- the last one given takes effect.

−s *program*, **−−speller=***program*
Use this alternative spell checker command.

−t, **−−tempfile**
Save a changed buffer without prompting (when exiting with **^X**).

−u, **−−unix**
Save a file by default in Unix format. This overrides nano's default behavior of saving a file in the format that it had. (This option has no effect when you also use **−−noconvert**.)

−v, **−−view**
Just view the file and disallow editing: read-only mode.

−w, **−−nowrap**
Disable the hard-wrapping of long lines. This option conflicts with **−r** -- the last one given takes effect.

−x, **−−nohelp**
Don’t show the two help lines at the bottom of the screen.

−z, **−−suspend**
Enable the suspend ability.

−$, **−−softwrap**
Enable ’soft wrapping’. This will make **nano** attempt to display the entire contents of any line, even if it is longer than the screen width, by continuing it over multiple screen lines. Since ’$’ normally refers to a variable in the Unix shell, you should specify this option last when using other options (e.g. ’nano −wS$’) or pass it separately (e.g. ’nano −wS −$’).

−a, **−b**, **−e**, **−f**, **−j**
Ignored, for compatibility with Pico.

TOGGLES

Several of the above options can be switched on and off also while **nano** is running. For example, **M−L** toggles the hard-wrapping of long lines, **M−$** toggles soft-wrapping, **M−#** toggles line numbers, **M−M** toggles the mouse, **M−I** auto-indentation, and **M−X** the help lines. See at the end of the **^G** help text for a complete list.

INITIALIZATION FILE

nano will read initialization files in the following order: the system’s **nanorc** (if it exists), and then the user’s **~/.nanorc** (if it exists). Please see **nanorc**(5) for more information on the possible contents of those files.

NOTES

If no alternative spell checker command is specified on the command line nor in one of the *nanorc* files, **nano** will check the **SPELL** environment variable for one.

In some cases **nano** will try to dump the buffer into an emergency file. This will happen mainly if **nano** receives a SIGHUP or SIGTERM or runs out of memory. It will write the buffer into a file named *nano.save* if the buffer didn’t have a name already, or will add a ".save" suffix to the current filename. If an emergency file with that name already exists in the current directory, it will add ".save" plus a number (e.g. ".save.1") to the current filename in order to make it unique. In multibuffer mode, **nano** will write all the open buffers to their respective emergency files.

BUGS

Justifications (**^J**) and reindentations (**M−{** and **M−}**) are not yet covered by the general undo system. So after a justification that is not immediately undone, or after any reindentation, earlier edits cannot be undone any more. The workaround is, of course, to exit without saving.

Please report any other bugs that you encounter via:
https://savannah.gnu.org/bugs/?group=nano.

HOMEPAGE

https://nano-editor.org/

SEE ALSO

nanorc(5)

/usr/share/doc/nano/ (or equivalent on your system)

AUTHOR

Chris Allegretta and others (see the files *AUTHORS* and *THANKS* for details). This manual page was originally written by Jordi Mallach for the Debian system (but may be used by others).

NAME

nmap − Ferramenta de exploração de Rede e Rastreio de Segurança / Portas

SYNOPSIS

nmap [*Tipo de Rastreio(Scan)...*] [*Opções*] {*Especificação do Alvo*}

DESCRIçãO

O Nmap (“Network Mapper”) é uma ferramenta em código aberto para exploração de rede e auditoria de segurança. Foi desenhada para rastrear(Scan) rapidamente redes amplas contudo funciona bem com um único anfitrião(host). Nmap usa pacotes IP em estado bruto(raw) sobre novas formas para determinar que anfitriões(hosts) estão disponíveis na rede, que serviços (nome e versão da aplicação) esses anfitriões(hosts) estão disponibilizando, que sistemas operativos (e versões de SO) estão em uso, que tipo de filtros de pacotes/firewalls estão em uso e dezenas de outras características. Enquanto o Nmap é frequentemente usado para auditorias de segurança, muitos sistemas e administradores de redes consideram−no útil para as tarefas de rotina como o inventário da rede, gestão de actualizações de serviços e monitorizar o uptime de anfitriões ou serviços.

A saída do Nmap é uma lista de alvos rastreados(scanned) com informações adicionais de cada um dependendo das opções utilizadas. Uma informação chave é a “tabela de portas interessantes”. Essa tabela lista o número da porta e o protocolo, o nome do serviço e o estado. O estado pode ser aberto (open), filtrado (filtered), fechado (closed), ou não−filtrado (unfilterd). Aberto (open) significa que uma aplicação na máquina−alvo está escutando as conexões/pacotes nessa porta. Filtrado (filtered) significa que o firewall, filtro ou outro obstáculo de rede está bloqueando a porta de forma que o Nmap não consegue dizer se ela está aberta (open) ou fechada (closed). Portas fechadas (closed)não possuem uma aplicação escutando nelas embora possam abrir a qualquer instante. Portas são classificadas como não filtradas (unfiltered)quando elas respondem às sondagens do Nmap mas o Nmap não consegue determinar se as portas estão abertas ou fechadas. O Nmap reporta as combinações aberta|filtrada (open|filtered)e fechada|filtrada (closed|filtered)quando não consegue determinar qual dos dois estados descrevem melhor a porta. A tabela de portas também pode incluir detalhes de versão de software quando a detecção de versão for solicitada. Quando um rastreio(scan) do protocolo IP é solicitado (**−sO**) o Nmap fornece informações dos protocolos IP suportados ao invés de portas que estejam abertas.

Além da tabela de portas interessantes o Nmap pode fornecer informações adicionais sobre os alvos, incluíndo nomes de DNS reverso, suposições de sistema operativo, tipos de dispositivos e endereços MAC.

Um rastreio(scan) típico do Nmap é mostrado em Example 1, “Uma amostra de rastreio(scan) do Nmap”. Os únicos argumentos que o Nmap utiliza nesse exemplo são **−A** para permitir a detecção de SO e a versão **−T4** para execução mais rápida e os nomes de anfitrião(hostnames) de dois alvos.

Example 1. Uma amostra de rastreio(scan) do Nmap

```
# nmap −A −T4 scanme.nmap.org playground

Starting nmap ( http://www.insecure.org/nmap/ )
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1663 ports scanned but not shown below are in state: filtered)
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 3.9p1 (protocol 1.99)
53/tcp  open   domain
70/tcp  closed gopher
80/tcp  open   http    Apache httpd 2.0.52 ((Fedora))
113/tcp closed auth
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.7 − 2.6.11, Linux 2.6.0 − 2.6.11
Uptime 33.908 days (since Thu Jul 21 03:38:03 2005)

Interesting ports on playground.nmap.org (192.168.0.40):
```

(The 1659 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE VERSION
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios−ssn
389/tcp open ldap?
445/tcp open microsoft−ds Microsoft Windows XP microsoft−ds
1002/tcp open windows−icfw?
1025/tcp open msrpc Microsoft Windows RPC
1720/tcp open H.323/Q.931 CompTek AquaGateKeeper
5800/tcp open vnc−http RealVNC 4.0 (Resolution 400x250; VNC TCP port: 5900)
5900/tcp open vnc VNC (protocol 3.8)
MAC Address: 00:A0:CC:63:85:4B (Lite−on Communications)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows XP Pro RC1+ through final release
Service Info: OSs: Windows, Windows XP

Nmap finished: 2 IP addresses (2 hosts up) scanned in 88.392 seconds

A versão mais nova do Nmap pode ser obtida em **http://www.insecure.org/nmap/**. A versão mais nova da página man está disponível em **http://www.insecure.org/nmap/man/**.

SUMáRIO DAS OPçõES

Este sumário de opções é mostrado quando o Nmap é executado sem argumentos e a última versão está sempre disponível em **http://www.insecure.org/nmap/data/nmap.usage.txt**. Ele ajuda as pessoas a lembrar−se das opções mais comuns mas não substitui a documentação mais técnica do restante deste manual. Algumas opções mais obscuras nem estão aqui incluídas.

Synopsis: nmap [Tipo(s) de Rastreio(Scan)] [Opções] {especificação do alvo}
ESPECIFICAÇÂO DO ALVO:
Pode−se usar nomes de anfitriões(hostnames), Endereços IP, redes, etc.
Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0−255.0−255.1−254
−iL <inputfilename>: Entrada(Input) de listas de anfitriões(hosts)/redes
−iR <num hosts>: Escolher alvos aleatoriamente
−−exclude <host1[,host2][,host3],...>: Excluir anfitriões(hosts)/redes
−−excludefile <exclude_file>: Lista de exclusões de um ficheiro
DESCOBERTA DE ANFITRIÕES(HOSTS):
−sL: List Scan − lista simplesmente os alvos para efectuar o rastreio(scan)
−sP: Ping Scan − apenas determnina se o anfitrião está online
−P0: Considera todos os anfitriões como online −− salta a descoberta de anfitriões
−PS/PA/PU [portlist]: rastreio de descoberta TCP SYN/ACK ou UDP para determinadas portas
−PE/PP/PM: Rastreio(scan) de descoberta ICMP echo, timestamp, and netmask request
−n/−R: Nunca resolver/Resolver sempre nomes de DNS [default: resolver algumas vezes]
TÉCNICAS DE SCAN:
−sS/sT/sA/sW/sM: Rastreios(Scans) TCP SYN/Connect()/ACK/Window/Maimon
−sN/sF/sX: Rastreios(Scans) TCP Null, FIN, and Xmas
−−scanflags <flags>: Customizar as TCP scan flags
−sI <anfitrião(host) zombie[:probeport]>: Idlescan
−sO: Rastreio(Scan) de protocolo IP
−b <ftp relay host>: FTP bounce scan
ESPECIFICAÇÃO DO PORTO E ORDEM DE RASTREIO:
−p <port ranges>: Apenas efectuar o rastreio(scan) de portas específicas
Ex: −p22; −p1−65535; −p U:53,111,137,T:21−25,80,139,8080
−F: Rápido − Efectua o rastreio(Scan) apenas das portas especificadas no ficheiro nmap−services
−r: Efectuar o rastreio(Scan) das portas consecutivas e não aleatoriamente
DETECÇÃO DO SERVIÇO/VERSÃO:

−sV: Rastrear(scan) portas abertas para determinar a informação sobre o serviço/versão
−−version−light: Limitar aos rastreios mais prováveis para identificação mais rápida
−−version−all: Experimentar todos os rastreios para detectar a versão
−−version−trace: Mostrar detalhadamente a actividade do rastreio(scan)
da versão (para debugging)
DETECÇÃO DO SO:
−O: Permite a detecção do SO
−−osscan−limit: Limitar a detecção de SO aos alvos promissores
−−osscan−guess: Efectuar o rastreio do SO de forma mais agressiva
TIMING AND PERFORMANCE:
−T[0−6]: Ajustar o tempo do modelo(template) (maior é mais rápido)
−−min−hostgroup/max−hostgroup <msec>: Tamanho dos grupos de rastreio(scan)
de anfitrião(host) paralelo
−−min−parallelism/max−parallelism <msec>: Rastreio paralelismo
−−min_rtt_timeout/max−rtt−timeout/initial−rtt−timeout <msec>: Ajustar o
tempo de retorno do rastreio.
−−host−timeout <msec>: Desistir de um alvo após este tempo
−−scan−delay/−−max_scan−delay <msec>: Ajustar esperas entre rastreios
FIREWALL/IDS EVASÃO E DISFARÇE(SPOOFING):
−f; −−mtu <val>: fragmentar pacotes (opcional com dado MTU)
−D <decoy1,decoy2[,ME],...>: Disfarça um rastreio(scan) com iscos
−S <IP_Address>: Disfarçar(Spoof) endereço de origem
−e <iface>: Usar um interface especifico
−g/−−source−port <portnum>: Usar um determinado numero de porta
−−data−length <num>: Acrescentar dados aleatorios aos pacotes enviados
−−ttl <val>: Ajustar o campo IP TTL tempo−de−vida
−−spoof−mac <mac address, prefix, or vendor name>: Disfarçar(Spoof) o endereço MAC
SAIDA(OUTPUT):
−oN/−oX/−oS/−oG <file>: Retorna os resultados do rastreio(scan) em XML normal, s|<rIpt kIddi3,
e formatados respectivamente para o ficheiro especificado
−oA <basename>: Saida(Output) nos três formatos principais
−v: Aumenta o nivel de informação apresentada(verbosity) (usar 2x para aumentar o efeito)
−d[level]: Ajusta o nivel de debugging (Áté 9 é significativo)
−−packet−trace: Mostra todos os pacotes enviados e recebidos
−−iflist: Mostra os interfaces do anfitrião e rotas (para debugging)
−−append−output: Acrescenta, em vez de destruir/substituir, ficheiros de resultados
−−resume <filename>: Continuar um rastreio cancelado(aborted)
−−stylesheet <path/URL>: A XSL stylesheet para transformar retorno(output) XML para HTML
−−no_stylesheet: Impedir que o Nmap de associar a XSL stylesheet com retorno(output) XML
OUTROS(MISC):
−6: Permitir rastreio(scanning) IPv6
−A: Permitir detecção do SO e versão
−−datadir <dirname>: Especifica a localização do ficheiro de dados personalizado do Nmap
−−send−eth/−−send−ip: Enviar pacotes utilizando "raw ethernet frames" ou pacotes IP
−−privileged: Assume que o utilizador possui os previlégios necessários
−V: Mostra a versão
−h: Mostra esta página de sumário de ajuda
EXEMPLOS:
nmap −v −A scanme.nmap.org
nmap −v −sP 192.168.0.0/16 10.0.0.0/8
nmap −v −iR 10000 −P0 −p 80

ESPECIFICAçãO DE ALVO

Tudo na linha de comando do Nmap que não for uma opção (ou argumento de uma opção) é tratado como uma especificação de um anfitrião (host)−alvo. O caso mais simples é especificar um endereço IP como alvo ou um nome de anfitrião(hostname) para ser rastreado(scaned).

Algumas vezes pode querer efectuar o rastreio(scan) de uma rede inteira de anfitriões(hosts) adjacentes. Para isso o Nmap suporta o estilo de endereçamento CIDR. Pode acrescentar /*númerodebits* em um endereço ou hostname e o Nmap irá efectuar o rastreio(scan) de cada endereço IP para o qual os primeiros *númerosdebits* sejam o mesmo que o IP de referência ou o hostname dado. Por exemplo, 192.168.10.0/24 escanearia os 256 hosts entre 192.168.10.0 (binário: 11000000 10101000 00001010 00000000) e 192.168.10.255 (binário: 11000000 10101000 00001010 11111111), inclusive. 192.168.10.40/24 faria exatamente a mesma coisa. Dado que o afitrião(host) scanme.nmap.org está no endereço IP 205.217.153.62, a especificação scanme.nmap.org/16 efectuaria o rastreio(scan) dos 65.536 endereços IP entre 205.217.0.0 e 205.217.255.255. O menor valor permitido é /1, que equivale ao rastreio(scan) de metada da Internet. O maior valor é 32, que faz o rastreio(scan) de apenas o anfitrião(host) nomeado ou endereço IP porque todos os bits de endereçamento estão fixos.

A notação CIDR é curta mas nem sempre flexível o suficiente. Por exemplo, pode querer fazer o rastreio(scan) de 192.168.0.0/16 mas desejar saltar todos os IPs terminados em .0 ou .255 porque eles são normalmente endereços de broadcast. O Nmap suporta isso através de endereçamento por faixa de octeto. Ao invés de especificar um endereço IP normal, pode especificar uma lista de números separada por vírgulas ou faixa de números para cada octeto. Por exemplo, 192.168.0−255.1−254 irá saltar todos os endereços na faixa que terminarem com .0 e/ou .255. Faixas não precisam ser limitadas ao octeto final: o especificador 0−255.0−255.13.37 irá executar um rastreio(scan) em toda a Internet buscando os endereços IP terminados em 13.37. Esse tipo de amostragem ampla pode ser útil em levantamentos e pesquisas de toda a Internet.

Endereços IPv6 podem apenas ser especificados utilizando o endereço IP ou hostname IPv6 completamente qualificado. Faixas CIDR e octetos não são suportados para o IPv6 porque eles raramente são úteis.

O Nmap aceita múltiplas especificações de anfitrião(host) na linha de comando, e elas não precisam ser do mesmo tipo. O comando **nmap scanme.nmap.org 192.168.0.0/8 10.0.0,1,3−7.0−255** executa o que se espera dele.

Embora os alvos sejam normalmente especificados na linha de comando, as seguintes opções também estão disponíveis para controlar a seleção de alvos:

−iL <arquivodeentrada> (Entrada a partir de uma lista)
Lê a especificação de alvos à partir de um *arquivodeentrada*. Passar uma lista enorme de anfitriões(hosts) na linha de comando é muito ruim, ainda que seja normalmente desejável. Por exemplo, o seu servidor DHCP pode exportar uma lista de 10.000 endereços correntes em uso que deseja efectuar o rastreio(scan). Ou talvez deseje o rastreio(scan) dr todos os endereços IP *excepto* aqueles usados para localizar anfitriões(hosts) que usam endereços IP estáticos não−autorizados. Simplesmente gere uma lista de anfitriões(hosts) a efectuar o rastreio(scan) e passe o nome do arquivo para o Nmap como um argumento à opção **−iL**. As entradas podem estar em qualquer um dos formatos aceites pelo Nmap na linha de comando (endereço IP, hostname, CIDR, IPv6, ou faixas de octetos). Cada entrada deve ser separada por um ou mais espaços em branco, tabulações ou newlines. Você pode especificar um hífen (−) como nome de arquivo se quiser que o Nmap leia os nomes de anfitrião(hostsnames) da entrada padrão (standard input) ao invés de um arquivo.

−iR <número de afitriões(hosts)> (Escolhe alvos aleatórios)
Para levantamentos na Internet toda e outras pesquisas, pode querer escolher alvos de forma aleatória. O argumento *número de anfitriões(hosts)* diz ao Nmap quantos IPs ele deverá gerar. IPs indesejáveis, tais como aqueles de certas redes privadas, multicast e faixas de endereços não−alocadas são automaticamente anuladas. O argumento 0 (zero) pode ser especificado caso deseje um rastreio(scan) sem fim. Tenha em mente que alguns administradores de rede não gostam de rastreios(scans) não−autorizados de suas redes e podem apresentar queixa Use esta opção por sua conta e risco! Se estiver realmente aborrecido em uma tarde chuvosa, tente o comando **nmap −sS −PS80 −iR 0 −p 80** para localizar servidores web aleatórios para navegar.

−−exclude <host1[,host2][,host3],...> (Exclui anfitriões(hosts)/redes)
Especifica uma lista de alvos, separados por vírgula, a serem excluídos do rastreio(scan) mesmo que façam parte da faixa de rede especificada. A lista que fornece utiliza a sintaxe normal do Nmap, portanto ela pode incluir nomes de afitrião(hosts), blocos de rede CIDR, faixas de octetos, etc. Isso pode ser útil quando a rede que deseja efectuar o rastreio(scan) inclui servidores de missão crítica intocáveis, sistemas que reconhecidamente reagem mal a rastreio(scan) de portas ou sub−redes administradas por outras pessoas.

−−excludefile <arquivo_exclusão> (Exclui a lista do arquivo)
Oferece a mesma funcionalidade que a opção **−−exclude**, excepto que os alvos a excluir são fornecidos em um *exclude_file* , delimitados por newline, espaço em branco ou tabulação, ao invés de na linha de comando.

DESCOBERTA DE HOSTS

Um dos primeiros passos em qualquer missão de reconhecimento de uma rede é reduzir um conjunto (às vezes enorme) de faixas de endereços IP, em uma lista de anfitriões(hosts) activos e interessantes. Efectuar o rastreio(scan) de cada porta de cada endereço IP é lento e normalmente desnecessário. É claro que o que torna um anfitrião(host) interessante depende muito do propósito do rastreio(scan). Administradores de rede podem estar apenas interessados em hosts que executam um determinado serviço, enquanto os auditores de segurança podem se importar com cada dispositivo que possuir um endereço IP. Um administrador pode se sentir à vontade em usar o ping ICMP para localizar os anfitriões(hosts) na rede interna, enquanto um profissional externo de análise de vulnerabilidades (penetration tester) pode utilizar um conjunto diversificado de dezenas de sondagens numa tentativa de enganar as restrições da firewall.

As necessidades para o descobrimento de anfitrião(host) são muito diversas e, por isso, o Nmap oferece uma ampla variedade de opções para customizar as técnicas utilizadas. A descoberta de anfitrião(host) às vezes é chamada de rastreo ping(ping scan), mas ela vai muito além dos simples pacotes ICMP de echo request associados com a popular ferramenta conhecida como ping. Os usuários podem saltar a etapa do ping inteiramente com uma lista de rastreio(scan) (**−sL**) ou desactivado o ping (**−P0**), ou enfrentar a rede com combinações arbitrárias de sondagens multi−portas TCP SYN/ACK, UDP, e ICMP. O objetivo dessas sondagens é solicitar respostas que mostrem que um endereço IP está realmente activo (é utilizado por um afitrião(host) ou dispositivo de rede). Em muitas redes, apenas uma pequena percentagem dos endereços IP está activa em um dado momento. Isso é particularmente comum com o espaço de endereçamento privado ao abrigo do RFC1918 como, por exemplo, 10.0.0.0/8. Essa rede tem 16 milhões de IPs, mas eu já a vi sendo utilizado em empresas com menos de mil máquinas. A descoberta de anfitriões(hosts) pode encontrar essas máquinas escassamente alocadas em um mar de endereços IP.

Se nenhuma opção de descoberta de hosts for dada, o Nmap envia um pacote TCP ACK destinado a porta 80 e uma procura ICMP Echo Request a cada máquina−alvo. Uma exceção a isso é que um rastreio(scan) ARP é utilizado para cada alvo localizado na rede ethernet local. Para usuários Unix sem privilégios de shell, um pacote SYN é enviado ao invés do ack utilizando a chamada de sistema **connect()**. Esses valores default equivalem às opções **−PA −PE**. Esta descoberta de anfitrião(host) frequentemente é suficiente para o rastreio(scan) de redes locais, mas um conjunto de sondagens mais abrangentes é recomendado para auditoria de segurança.

As opções **−P*** (que seleccionam tipos de ping) podem ser combinadas. Você pode aumentar as chances de penetrar numa firewall enviando muitos tipos de sondagens, utilizando diferentes portas/flags TCP e códigos ICMP. Note também que a descoberta por ARP (**−PR**) é feita por default contra alvos na rede ethernet local mesmo que especifique outras opções **−P*** , porque é quase sempre mais rápida e mais eficiente.

As seguintes opções controlam a descoberta de anfitriões(hosts).

−sL (Listagem de rastreio(scan))
A listagem de rastreio(scan) é uma forma degenerada de descoberta de anfitriões(hosts) que simplesmente lista cada anfitrião(host) da rede especificada, sem enviar nenhum pacote aos hosts−alvos. Por default o Nmap fará a resolução de DNS dos anfitriões(hosts) para descobrir seus nomes. Ainda é surpreendente a quantidade de informações úteis que simples nomes de hosts podem dar. Por exemplo, fw.chi.playboy.com é o firewall do escritório de Chicago da Playboy Enterprises.

Nmap também reporta o número total de endereços IP ao final. A listagem de rastreio(scan) é um bom teste de sanidade para assegurar que está com a lista correta de endereços IP dos seus alvos. Se os anfitriões(hosts) mostrarem nomes de domínios que não reconhece, vale a pena investigar melhor para evitar o rastreio(scan) da rede da empresa errada.

Uma vez que a idéia é apenas mostrar uma lista dos hosts−alvos, as opções de funcionalidade de nível mais alto tais como o rastreio(scan) de portas, detecção de SO, ou rastreio(scan) utilizando ping, não podem ser combinadas com esta opção. Se deseja desactivar o rastreio(scan) utilizando ping enquanto executa funções de nível elevado, leia a opção **−P0**.

−sP (Rastreio(scan) usando Ping)
Esta opção diz ao Nmap para *somente* executar um rastreio(scan) usando o ping (descoberta de anfitriões(hosts)), e então mostrar os hosts disponíveis que responderam ao scan. Nenhum teste adicional (tais como o rastreio(scan) de portas e deteção de SO) é executado. Isto é um pouco mais intrusivo que a listagem de rastreio(scan), e pode ser usado para os mesmos propósitos. Permite um reconhecimento leve de uma rede−alvo sem chamar muita atenção. Saber quantos hosts estão ativos é mais valioso para invasores que a lista fornecida pela listagem de rastreio(scan) com cada endereço IP e seu nome de anfitrião(host).

Administradores de sistemas frequentemente acham esta opção valiosa. Ela pode ser facilmente utilizada para contar o número de máquinas disponíveis em uma rede ou monitorar a disponibilidade dos servidores. Isto é normalmente chamado de varredura com ping (ping sweep), e é mais confiável do que fazer um ping num endereço de broadcast, pois muitos anfitriões(rastreio(scan)hosts) não respondem a pesquisas com broadcast.

A opção **−sP** envia um ICMP echo request e um pacote TCP para a porta 80 por default. Quando executada por um usuário sem privilégios, um pacote SYN é enviado (usando uma chamada **connect()**) para a porta 80 no alvo. Quando um usuário privilegiado tenta rastrear(scan) alvos na rede ethernet local, requisições ARP (**−PR**) são utilizadas, a menos que **−−send−ip** tenha sido especificado. A opção **−sP** pode ser combinada com qualquer um dos tipos de sondagens de descobrimento (as opções **−P*** , excluindo **−P0**) para maior flexibilidade. Se qualquer uma dessas opções de tipos de sondagens e número de porta for utilizada, as sondagens default (ACK e echo request) são sobrepostas. Quando firewalls restritivos estão posicionados entre o host de origem que executa o Nmap e a rede−alvo, utilizar essas técnica avançadas é recomendado. Do contrário, hosts podem ser perdidos quando o firewall ignorar as sondagens ou as respostas delas.

−P0 (Sem ping)
Esta opção salta completamente a fase de descoberta do Nmap. Normalmente o Nmap utiliza este estágio para determinar as máquinas activas para o rastreio(scan) mais agressivo. Por default, o Nmap apenas executa sondagens agressivas tais como o rastreio(scan) de portas, detecção de versões, ou detecções do SO contra afitriões(hosts) que foram verificados como activos. Desactivar a descoberta de anfitriões(hosts) com **−P0** faz com que o Nmap teste as funções de rastreio(scan) solicitadas contra *todos* os endereços IP alvos especificados. Portanto se um espaço de endereçamento alvo do tamanho de uma classe B (/16) for especificado na linha de comando, todos os 65.536 endereços IP serão alvo do rastreio(scan). O segundo caracter da opção **−P0** é um zero e não a letra O. A descoberta de anfitriões(hosts) apropriada é desconsiderada como na listagem de rastreio(scan), mas ao invés de parar e mostrar a lista de alvos, o Nmap continua a executar as funções solicitadas como se cada alvo IP estivesse activo.

−PS [listadeportas] (Ping usando TCP SYN)
Esta opção envia um pacote TCP vazio com a flag SYN marcada. A porta de destino default é a 80 (configurada em tempo de compilação pela variável DEFAULT_TCP_PROBE_PORT no nmap.h), mas uma porta alternativa pode ser especificada como um parâmetro. Até uma lista de portas separadas por vírgula pode ser especificada (p.ex. **−PS22,23,25,80,113,1050,35000**), nesse caso as sondagens serão tentadas contra cada porta em paralelo.

A flag SYN sugere aos sistemas remotos que está tentando estabelecer uma comunicação. Normalmente a porta de destino estará fechada e um pacote RST (reset) será enviado de volta. Se a porta estiver aberta, o alvo irá dar o segundo passo do cumprimento−de−três−vias (3−way−handshake) do TCP respondendo com um pacote TCP SYN/ACK TCP. A máquina executando o Nmap então derruba a conexão recém−criada respondendo com um RST ao invés de enviar um pacote ACK que iria completar o cumprimento−de−três−vias e estabelecer uma conexão completa. O pacote RST é enviado pelo kernel da máquina que está executando o Nmap em resposta ao SYN/ACK inesperado, e não pelo próprio Nmap.

O Nmap não se importa se a porta está aberta ou fechada. Tanto a resposta RST ou SYN/ACK discutidas anteriormente dizem ao Nmap se o hosts está disponível e responsivo.

Em máquinas UNIX apenas o usuário privilegiado root é capaz, normalmente, de enviar e receber pacotes TCP em estado bruto(raw packets). Para usuários não privilegiados um contorno é automaticamente empregado em concordância com a chamada de sistema connect() iniciada contra cada porta−alvo. Isso tem o efeito de enviar um pacote SYN ao anfitrião(host) alvo em uma tentativa de estabelecer uma conexão. Se o connect() retornar com sucesso rápido ou com uma falha ECONNREFUSED, a pilha TCP subjacente deve ter recebido um SYN/ACK ou RST e o anfitrião(host) é marcado como disponível. Se a tentativa de conexão for abandonada até que um timeout ocorra, o host é marcado como indisponível. Esse contorno também é usado para conexões IPv6, pois o suporte a construção de pacotes IPv6 em estado bruto(raw) ainda não está disponível no Nmap.

−PA [listadeportas] (Ping usando TCP ACK)

O ping usando TCP ACK é muito similar ao recém−discutido ping usando SYN. A diferença como poderia imaginar, é que a flag TCP ACK é marcada ou invés da flag SYN. O pacote ACK finge reconhecer dados de uma conexão TCP estabelecida, quando nenhuma conexão existe de facto. Então os anfitriões(hosts) remotos deveriam sempre responder com pacotes RST revelando sua existência no processo.

A opção **−PA** utiliza a mesma porta default que a sondagem SYM (80) e pode também obter uma lista de portas destino no mesmo formato. Se um usuário privilegiado tenta isto, ou se um alvo IPv6 é especificado, o contorno connect() discutido anteriormente é utilizado. Esse contorno é imperfeito pois o connect() está realmente enviando um pacote SYN ao invés de um ACK.

O motivo para oferecer ambas as sondagens ping, que utilizam SYN e ACK, é maximizar as chances de passar por firewalls. Muitos administradores configuram routers e outros firewalls simples para bloquear a entrada de pacotes SYN excepto aqueles destinados a serviços públicos como o site web da empresa ou servidor de correio electrónico. Isso evita as demais conexões entradas na organização, permitindo aos usuários fazer conexões desobstruidas à Internet. Essa aproximação não−orientada à conexão (non−stateful ou stateless) consome poucos recursos no firewall/router e é amplamente suportada por filtros de hardware e software. O firewall de software Netfilter/iptables do Linux oferece a conveniência da opção **−−syn** para implementar essa abordagem stateless. Quando regras stateless do firewall como essas são implementadas, sondagens de ping usando SYN (**−PS**) muito provavelmente serão bloqueadas quando forem enviadas à portas fechadas. Nesses casos, a sondagem ACK se destaca pois ela simplesmente passa por essas regras.

Outro tipo comum de firewall utiliza regras orientadas a conexão que descartam pacotes inesperados. Esta característica era encontrada inicialmente apenas em firewalls de alto−nível, embora tenha se tornado mais comum com o passar dos anos. O sistema Netfilter/iptables do Linux suporta esta característica através da opção **−−state**, que categoriza os pacotes baseados no estado da conexão. Uma sondagem SYN tem maiores chances de funcionar contra um sistema assim, pois pacotes ACK inesperados são normalmente reconhecidos como falsos e descartados. Uma solução para esse dilema é enviar ambas as sondagens SYN e ACK especificando **−PS** e **−PA**.

−PU [listadeportas] (Ping usando UDP)

Outra opção de descoberta de anfitriões(hosts) é o ping usando UDP, que envia um pacote UDP vazio (a menos que **−−data−length** seja especificado) para as portas informadas. A listadeportas tem o mesmo formato que os discutidos anteriormente nas opções **−PS** e **−PA**. Se nenhuma porta for especificada, o default é 31338. Esse default pode ser configurado em tempo de compilação alterando DEFAULT_UDP_PROBE_PORT no nmap.h. Uma porta alta não comum é utilizada como default porque enviar para portas abertas normalmente é indesejado para este tipo particular de rastreio(scan).

Ao bater contra uma porta fechada na máquina−alvo, a sondagem UDP deve criar um pacote ICMP de porta inalcançável como resposta. Isso diz ao Nmap que a máquina está activa e disponível. Muitos outros tipos de erros ICMP, tais como anfitrião(host)/rede inalcançável ou TTL excedido são indicativos de um anfitrião(host) inactivo ou inalcançável. A falta de resposta também é interpretada dessa forma. Se uma porta aberta é alcançada, a maioria dos serviços simplesmente ignoram o pacote vazio e falham em retornar qualquer resposta. É por isso que a porta de sondagem default é 31338, que pouco provavelmente estará em uso. Uns poucos serviços, tal como o chargen, irá responder a um pacote UDP vazio, e com isso revelará ao Nmap que a máquina está disponível.

A principal vantagem deste tipo de scan é que ele passa por firewalls e filtros que apenas examinam o TCP. Por exemplo, uma vez eu tive um router broadband sem−fios Linksys BEFW11S4. A interface externa desse dispositivo filtrava todas as portas TCP por default, mas as sondagens UDP ainda causavam mensagens de porta inalcançável, denunciando assim o dispositivo.

−PE; **−PP**; **−PM** (Tipos de Ping do ICMP)
Além dos tipos incomuns de descoberta de anfitriões(hosts) TCP e UDP discutidos anteriormente, o Nmap pode enviar os pacotes−padrão que normalmente são enviados pelo popular programa ping. O Nmap envia um pacote ICMP do tipo 8 (echo request) ao endereço IP alvo, esperando como resposta um tipo 0 (Echo Reply) do anfitrião(host) disponível. Infelizmente para muitos exploradores de rede, muitos anfitriões(hosts) e firewalls actualmente bloqueiam esses pacotes, ao invés de responder como é requerido pela **RFC 1122**[1]. Por essa razão, rastreios(scans) puramente ICMP são raramente confiáveis o suficiente contra alvos desconhecidos na Internet. Mas para administradores de sistemas monitorando uma rede interna eles podem ser uma abordagem prática e eficiente. Utilize a opção **−PE** para activar esse comportamento echo request.

Embora o echo request seja a pesquisa padrão de um ping ICMP, o Nmap não pára aqui. A padronização do ICMP (**RFC 792**[2]) também especifica timestamp request, information request, e pacotes address mask request como códigos 13, 15, e 17, respectivamente. Apesar do propósito ostensivo dessas pesquisas seja obter informações tais como a máscara do endereço e hora corrente, eles podem ser facilmente utilizados para descoberta de anfitriões(hosts). Um sistema que responda está activo e disponível. O Nmap não implementa actualmente os pacotes de requisição de informações, pois eles não são amplamente suportados. A RFC 1122 insiste que “um anfitrião(host) NÃO DEVERIA implementar essas mensagens”. Pesquisas de marcação de hora (Timestamp) e máscara de endereço podem ser enviadas com as opções **−PP** e **−PM** , respectivamente. Uma resposta timestamp reply (código ICMP 14) ou uma resposta address mask reply (código 18) revela que o host está disponível. Essas duas pesquisas podem ser valiosas quando os administradores bloqueiam pacotes echo request especificamente e esquecem que outras pesquisas ICMP podem ser usadas com o mesmo propósito.

−PR (Ping usando ARP)
Um dos cenários de uso mais comuns do Nmap é o rastreio(scan) da LAN ethernet. Na maioria das LANs, especialmente aquelas que utilizam a faixa de endereçamento privado ao abrigo do RFC1918, a vasta maioria dos endereços IP nuca são utilizados. Quando o Nmap tenta enviar um pacote IP em estado bruto(raw), tal como um ICMP echo request, o sistema operativo deve determinar o endereço físico de destino (ARP) correspondente ao IP−alvo de forma que ele possa endereçar adequadamente o frame ethernet. Isso normalmente é lento e problemático, pois os sistemas operativos não foram escritos com a expectativa de que precisariam fazer milhões de requisições ARP contra anfitriões(hosts) indisponíveis em um curto período de tempo.

O rastreio(scan) ARP encarrega o Nmap e seus algorítmos optimizados de fazer as requisições ARP. E se ele conseguir uma resposta de volta, o Nmap não precisa de se preocupar com os pacotes ping baseados em IP, uma vez que ele já sabe que o anfitrião(host) está activo. Isso torna o rastreio(sca) ARP muito mais rápido e mais confiável que os rastreios(scans) baseados em IP. Portanto isso é feito por default quando se faz o rastreio(scan) de anfitriões(hosts) ethernet que o Nmap detecta estarem posicionados em uma rede ethernet local. Mesmo se tipos diferentes de ping (tais como **−PI** ou **−PS**) sejam especificados, o Nmap usa o ARP em vez, para cada um dos alvos que estiverem na mesma LAN. Se não quiser de forma alguma fazer um ratreio(scan) ARP, especifique **−−send−ip**.

−n (Não faça resolução DNS)
Diz ao Nmap para *nunca* fazer uma resolução DNS nos endereços IP activos que ele encontrar. Uma vez que o DNS é normalmente lento, isso acelera as coisas.

−R (resolução DNS para todos os alvos)
Diz ao Nmap para fazer *sempre* uma resolução DNS reversa nos endereços IP−alvos. Normalmente isto apenas é executado quando uma máquina está activa.

FUNDAMENTOS DO RASTREIO(SCAN) DE PORTAS

Embora o Nmap tenha crescido em funcionalidades ao longo dos anos, ele começou como um eficiente scanner de portas e essa permanece a sua função principal. O simples comando **nmap** *alvo* faz o rastreio(scan) a mais de 1660 portas TCP no anfitrião(host) *alvo*. Embora muitos scanner de portas tenham tradicionalmente agrupado todas as portas nos estados aberto ou fechado, o Nmap é muito mais granular. Ele divide as portas em seis estados: aberto(open), fechado(closed),filtrado(filtered), não−filtrado(unfiltered), aberto(open)|filtrado(filtered), ou fechado(closed)|filtrado(filtered).

Esses estados não são propriedades intrínsecas da porta mas descrevem como o Nmap as vê. Por exemplo, um rastreio(scan) do Nmap da mesma rede como alvo pode mostrar a porta 135/tcp como aberta, enquanto um rastreio(scan) ao mesmo tempo com as mesmas opções a partir da Internet poderia mostrar essa porta como filtrada.

Os seis estados de porta reconhecidos pelo Nmap

aberto (open)
Uma aplicação está activamente aceitando conexões TCP ou pacotes UDP nesta porta. Encontrar esse estado é frequentemente o objectivo principal de um rastreio(scan) de portas. Pessoas conscientes sobre a segurança sabem que cada porta aberta é um convite para um ataque. Invasores e profissionais de avaliação de segurança querem explorar as portas abertas, enquanto os administradores tentam fechar ou proteger com firewalls sem bloquear usuários legítimos. Portas abertas são também interessantes para rastreios(scans) não−relacionados à segurança pois mostram os serviços disponíveis para utilização na rede.

fechado (closed)
Uma porta fechada está acessível (ela recebe e responde a pacotes de sondagens do Nmap), mas não há nenhuma aplicação ouvindo nela. Elas podem ser úteis para mostrar que um anfitrião(host) está activo em um determinado endereço IP (descoberta de hosts, ou rastreio(scan) usando ping), e como parte de uma deteção de SO. Pelo facto de portas fechadas serem alcançáveis, pode valer a pena o rastreio(scan) mais tarde no caso de alguma delas abrir. Os administradores deveriam considerar o bloqueio dessas portas com um firewall. Então elas apareceriam no estado filtrado, discutido a seguir.

filtrado(filtered)
O Nmap não consegue determinar se a porta está aberta porque uma filtragem de pacotes impede que as sondagens alcancem a porta. A filtragem poderia ser de um dispositivo firewall dedicado, regras de router, ou um software de firewall baseado em anfitrião(host). Essas portas frustram os atacantes pois elas fornecem poucas informações. Às vêzes elas respondem com mensagens de erro ICMP tais como as do tipo 3 código 13 (destino inalcançável: comunicação proibida administrativamente), mas os filtros que simplesmente descartam pacotes sem responder são bem mais comuns. Isso força o Nmap a tentar diversas vezes só para o caso de a sondagem ter sido descartada por congestionamento da rede ao invés de filtragem. Isso reduz a velocidade do rastreio(scan) dramaticamente.

não−filtrado(unfiltered)

O estado não−filtrado significa que uma porta está acessível, mas que o Nmap é incapaz de determinar se ela está aberta ou fechada. Apenas o rastreio(scan) ACK, que é usado para mapear conjuntos de regras de firewall classifica portas com este estado. O rastreio(scan) de portas não−filtradas com outros tipos de scan, tal como scan Window, scan Syn, ou scan FIN, podem ajudar a responder se a porta está aberta.

open|filtered
O Nmap coloca portas neste estado quando é incapaz de determinar se uma porta está aberta ou filtrada. Isso acontece para tipos de rastreio(scan) onde as portas abertas não dão nenhuma resposta. A falta de resposta poderia também significar que um filtro de pacotes descartou a sondagem ou qualquer resposta que ela tenha provocado. Portanto o não sabe com certeza se a porta está aberta ou se está sendo filtrada. Os rastreios(scans) UDP, IP Protocol, FIN, Null, e Xmas classificam portas desta forma.

closed|filtered
Este estado é usado quando o Nmap é incapaz de determinar se uma porta está fechada ou filtrada. É apenas usado para o rastreio(scan) IPID Idle scan.

TéCNICAS DE RASTREIO(SCAN) DE PORTAS

Como um novato executando uma reparação automóvel posso perder horas tentando usar minhas ferramentas rudimentares (martelo, fita adesiva, grifo, etc.) nas tarefas. Quando eu falho miseravelmente e reboco minha lata−velha para um mecânico de verdade ele invariavelmente pesca aqui e ali em um enorme baú de ferramentas até pegar a coisa perfeita que torna a tarefa numa brincadeira. A arte de rastrear(scaning) portas é similar. Os peritos entendem as dezenas de técnicas de rastreio(scan) e escolhem as que são apropriadas (ou uma combinação) para uma dada tarefa. Usuários inexperientes e script kiddies, por outro lado, tentam resolver todos os problemas com o scan SYN default. Uma vez que o Nmap é gratuito a única barreira para a mestria em rastreio(scaning) de portas é o conhecimento. Isso certamente é melhor que no mundo automóvel onde pode ser necessário uma grande habilidade para determinar que precisa de um compressor de molas e então tem que pagar milhares de euros por um.

A maioria dos tipos de rastreio(scan) está disponível apenas para usuários privilegiados. Isso acontece porque eles enviam e recebem pacotes em estado bruto(raw), o que requer acesso de root em sistemas Unix. Utilizar a conta de administrador no Windows é recomendado, embora o Nmap às vêzes funcione com usuários sem privilégios nessa plataforma quando o WinPcap foi carregado no SO. Requerer privilégio de root era uma séria limitação quando o Nmap foi lançado em 1997, pois muitos usuários apenas tinham acesso a contas de shell compartilhadas. Agora o mundo é diferente. Computadores estão mais baratos, muito mais pessoas tem acesso directo e permanente à Internet e computadores desktop Unix (incluindo Linux e MAC OS X) são comuns. Uma versão para o Windows do Nmap se encontra actualmente disponível permitindo que se use em muito mais computadores desktop. Por todas essas razões os usuários têm menos necessidade de executar o Nmap a partir de contas de shell compartilhadas e limitadas. Isso é muito bom pois as opções privilegiadas tornam o Nmap muito mais poderoso e flexível.

Embora o Nmap tente produzir resultados precisos tenha em mente que todas as deduções são baseadas em pacotes devolvidos pelas máquinas−alvo (ou firewalls na frente delas). Tais anfitriões(hosts) podem não ser confiáveis e enviar respostas com o propósito de confundir ou enganar o Nmap. Muito mais comum são os anfitriões(hosts) não−de−acordo−com−a−rfc que não respondem como deveriam às sondagens do Nmap. As sondagens FIN, Null e Xmas são particularmente suscetíveis a esse problema. Tais questões são específicas de determinados tipos de scan e portanto são discutidos nas entradas individuais de cada um dos tipos.

Esta seção documenta as dezenas de técnicas de rastreio(scan) de portas suportadas pelo Nmap. Apenas um método pode ser utilizado de cada vezm excepto que um scan UDP (**−sU**) pode ser combinado com qualquer um dos tipos de scan TCP. Como uma ajuda para a memória as opções dos tipos de rastreio(scan) de portas estão no formato **−s***C*, onde *C* é um caracter proeminente no nome do rastreio(scan), normalmente o primeiro. A única excepção a essa regra é para o rastreio(scan) denominado FTP bounce (**−b**). Por default o Nmap executa um rastreio(scan) SYN, embora ele substitua por um rastreio(scan) Connect() se o usuário não tiver os privilégios adequados para enviar pacotes em estado bruto(raw) (requer acesso de root no UNIX) ou se alvos IPv6 forem especificados. Dos rastreios(scans) listados nesta secção os usuários não privilegiados podem apenas executar os rastreios(scans) connect() e ftp bounce.

−sS (rastreio(scan) TCP SYN)
O rastreio(scan) SYN é a opção de rastreio(scan) default e a mais popular por boas razões. Pode ser executada rapidamente fazendo o rastreio(scan) a milhares de portas por segundo em uma rede rápida, não bloqueada por firewalls intrusivos. O rastreio(scan) SYN é relativamente não−obstrusivo e camuflado, uma vez que ele nunca completa uma conexão TCP. Ele também trabalha contra qualquer pilha TCP padronizada ao invés de depender de factores específicos de plataformas como os rastreios(scans) Fin/Null/Xmas, Maimon e Idle fazem. Ele também permite uma diferenciação limpa e confiável entre os estados aberto (open), fechado (closed), e filtrado (filtered).

Esta técnica é freqüentemente chamada de rastreio(scan) de porta entreaberta (half−open scanning), porque não abre uma conexão TCP completamente. Você envia um pacote SYN, como se fosse abrir uma conexão real e então espera uma resposta. Um SYN/ACK indica que a porta está ouvindo (aberta) enquanto um RST (reset) é indicativo de uma não−ouvinte. Se nenhuma resposta é recebida após diversas retransmissões a porta é marcada como filtrada. A porta também é marcada como filtrada se um erro ICMP de inalcançável é recebido (tipo 3, código 1,2, 3, 9, 10, ou 13).

−sT (rastreio(scan) TCP connect())
O rastreio(scan) TCP Connect() é o rastreio(scan) default do TCP quando o rastreio(scan) SYN não é uma opção. Esse é o caso quando o usuário não tem privilégios para criar pacotes em estado bruto(raw) ou rastrear redes IPv6. Ao invés de criar pacotes em estado bruto(raw) como a maioria dos outros tipos de rastreio(scan) fazem, o Nmap pede ao sistema operativo para estabelecer uma conexão com a máquina e porta alvos enviando uma chamada de sistema connect(). Essa é a mesma chamada de alto nível que os navegadores da web, clientes P2P, e a maioria das outras aplicações para rede utilizam para estabelecer uma conexão. É parte do interface de programação conhecida como API de Sockets de Berkeley. Ao invés de ler as respostas em pacotes em estado bruto(raw) directamente dos fios, o Nmap utiliza esta API para obter informações do estado de cada tentativa de conexão.

Quando um rastreio(scan) SYN está disponível é normalmente a melhor escolha. O Nmap tem menos controle sobre a chamada de alto nível connect() do que sobre os pacotes em estado bruto(raw) tornando−o menos eficiente. A chamada de sistema completa as conexões nas portas−alvo abertas ao invés de executar o reset de porta entreaberta que o rastreio(scan) SYN faz. Isso não só leva mais tempo e requer mais pacotes para obter a mesma informação mas também torna mais provável que as máquinas−alvo registrem a conexão. Um sistema IDS decente irá detectar qualquer um deles, mas a maioria das máquinas não tem esse tipo de sistema de alarme. Muitos serviços na maioria dos sistema Unix irão acrescentar uma nota na syslog e às vêzes uma mensagem de erro obscura, quando o Nmap se conecta e então fecha a conexão sem enviar nenhum dado. Serviços verdadeiramente patéticos irão travar quando isso acontecer embora isso seja incomum. Um administrador que vê um punhado de tentativas de conexão nos registros vindos de um único sistema deveria saber que foi rastreado(scanned) com connect.

−sU (rastreios(scans) UDP)
Embora os serviços mais populares na Internet operem sobre o protocolo TCP, os serviços **UDP**[3] são amplamente difundidos. O DNS, o SNMP e o DHCP (registrados nas portas 53, 161/162, e 67/68) são três dos mais comuns. Pelo facto do rastreio(scan) UDP ser normalmente mais lento e mais difícil que o TCP alguns auditores de segurança ignoram essas portas. Isso é um erro pois serviços UDP passíveis de exploração são bastante comuns e invasores certamente não ignoram o protocolo inteiro. Felizmente o Nmap pode ajudar a inventariar as portas UDP.

O rastreio(scan) UDP é activado com a opção **−sU**. Ele pode ser combinado com um tipo de rastreio(scan) TCP como o rastreio(scan) SYN (**−sS**) para averiguar ambos protocolos na mesma execução.

O SYN UDP funciona enviando um cabeçalho UDP vazio (sem dados) para cada porta pretendida. Se um erro ICMP de porta inalcançável (tipo 3, código 3) é retornado a porta está fechada. Outros erros do tipo inalcançável (tipo 3, códigos 1, 2, 9, 10, ou 13) marcam a porta como filtrada. Ocasionalmente um serviço irá responder com um pacote UDP provando que está aberta. Se nenhuma resposa é

recebida após as retransmissões a porta é classificada como aberta|filtrada. Isso significa que a porta poderia estar aberta ou talvez que filtros de pacotes estejam bloqueando a comunicação. Rastreios(scans) de versões (**−sV**) podem ser utilizados para ajudar a diferenciar as portas verdadeiramente abertas das que estão filtradas.

Um grande desafio com o rastreio(scan) UDP é fazê−lo rapidamente. Portas abertas e filtradas raramente enviam alguma resposta, deixando o Nmap esgotar o tempo (time out) e então efectuar retransmissões para o caso de a sondagem ou a resposta ter sido perdida. Portas fechadas são normalmente um problema ainda maior. Elas costumam enviar de volta um erro ICMP de porta inalcançável. Mas, ao contrário dos pacotes RST enviados pelas portas TCP fechadas em resposta a um rastreio(scan) SYN ou Connect, muitos anfitriões(hosts) limitam a taxa de mensagens ICMP de porta inalcançável por default. O Linux e o Solaris são particularmente rigorosos quanto a isso. Por exemplo, o kernel 2.4.20 do Linux limita a quantidade de mensagens de destino inalcançável a até uma por segundo (no net/ipv4/icmp.c).

O Nmap detecta a limitação de taxa e diminui o ritmo de acordo para evitar inundar a rede com pacotes inúteis que a máquina−alvo irá descartar. Infelizmente, um limite como o do Linux de um pacote por segundo faz com que um rastreio(scan) de 65.536 portas leve mais de 18 horas. Idéias para acelerar o rastreio(scan) UDP incluem rastrear(scan) mais anfitriões(hosts) em paralelo, fazer um rastreio(scan) rápido apenas das portas mais comuns primeiro, rastrear(scan) por detrás de um firewall e utilizar **−−host−timeout** para saltar os anfitriões(hosts) lentos.

−sN; **−sF**; **−sX** (rastreios(scans) TCP Null, FIN, e Xmas)
Estes três tipos de rastreio(scan) (até mais são possíveis com a opção **−−scanflags** descrita na próxima secção) exploram uma brecha subtil na **RFC do TCP**[4] para diferenciarem entre portas abertas e fechadas. A página 65 diz que “se a porta [destino] estiver FECHADA .... um segmento de entrada que não contenha um RST irá causar o envio de um RST como resposta.” Então a página seguinte discute os pacotes enviados a portas abertas sem os bits SYN, RST ou ACK marcados, afirmando que: “é pouco provável que chegue aqui, mas se chegar, descarte o segmento e volte.”

Quando se rastreia(scan) sistemas padronizados com o texto desta RFC, qualquer pacote que não contenha os bits SYN, RST ou ACK irá resultar em um RST como resposta se a porta estiver fechada e nenhuma resposta se a porta estiver aberta. Contanto que nenhum desses três bits esteja incluídos qualquer combinação dos outros três (FIN, PSH e URG) é válida. O Nmap explora isso com três tipos de rastreio(scan):

rastreio(scan) Null (**−sN**)
Não marca nenhum bit (o cabeçalho de flag do tcp é 0)

rastreio(scan) FIN (**−sF**)
Marca apenas o bit FIN do TCP.

rastreio(scan) Xmas(**−sX**)
Marca as flags FIN, PSH e URG, iluminando o pacote como uma árvore de Natal.

Estes três tipos de rastreio(scan) são exatamente os mesmos em termos de comportamento exceto pelas flags TCP marcadas no pacotes de sondagem. Se um pacote RST for recebido a porta é considerada fechada e nenhuma resposta significa que está aberta|filtrada. A porta é marcada como filtrada se um erro ICMP do tipo inalcançável (tipo 3, código 1, 2, 3, 9, 10, ou 13) for recebido.

A vantagem principal destes tipos de rastreio(scan) é que eles podem bisbilhotar através de alguns firewalls não−orientados à conexão e de routers que filtram pacotes. Outra vantagem é que esses tipos de rastreio(scan) são um pouco mais camuflados do que o rastreio(scan) SYN. Mas não conte com isso −− a maioria dos produtos IDS modernos podem ser configurados para detectá−los. O maior problema é que nem todos os sistemas seguem a RFC 793 ao pé−da−letra. Diversos sistemas enviam respostas RST para as sondagens independentemente do facto da porta estar aberta ou não. Isso faz com que todas as portas sejam classificadas como fechadas. A maioria dos sistemas operativos que fazem isso

são Microsoft Windows, muitos dispositivos Cisco, BSDI e o IBM OS/400. Esse rastreio(scan) funciona realmente contra a maioria dos sistemas baseados em Unix. Outro ponto negativo desses rastreios(scans) é que eles não conseguem diferenciar portas abertas de alguns tipos de portas filtradas deixando com a resposta abera|filtrada.

−sA (rastreio(scan) TCP ACK)
Este rastreio(scan) é diferente dos outros discutidos até agora pelo facto de que ele nunca determina se uma porta está aberta (ou mesmo aberta|filtrada). Ele é utilizado para mapear conjuntos de regras do firewall determinando se eles são orientados à conexão ou não e quais portas estão filtradas.

O pacote de sondagem do rastreio(scan) ACK tem apenas a flag ACK marcada (a menos que use **−−scanflags**). Quando se rastreia(scan) sistemas não−filtrados as portas abertas e fechadas irão devolver um pacote RST. O Nmap então coloca nelas o rótulo não−filtradas (unfiltered) significando que elas estão alcançáveis pelo pacote ACK, mas se elas estão abertas ou fechadas é indeterminado. Portas que não respondem ou que devolvem certas mensagens de erro ICMP (tipo 3, código 1, 2, 3, 9, 10, ou 13), são rotuladas como filtradas.

−sW (rastreio(scan) da Janela TCP)
Rastreio(scan) da Janela é exactamente o mesmo que o rastreio(scan) ACK excepto que ele explora um detalhe da implementação de certos sistemas de forma a diferenciar as portas abertas das fechadas ao invés de sempre mostrar não−filtrada quando um RST é devolvido. Ele faz isso examinando o campo Janela TCP (TCP Window) do pacote RST devolvido. Em alguns sistemas as portas abertas usam um valor positivo de tamanho de janela (mesmo para pacotes RST) enquanto que as portas fechadas têm um valor igual a zero. Então, ao invés de mostrar sempre uma porta como não−filtrada quando se recebe um RST de volta, o rastreio(scan) da Janela mostra a porta como aberta ou fechada se o valor da Janela TCP no reset for positivo ou zero, respectivamente.

Este rastreio(scan) se baseia em um detalhe de implementação de uma minoria de sistemas na Internet, portanto não se pode confiar sempre nele. Sistemas que não suportam isso irão normalmente devolver todas as portas como fechadas. É claro que é possível que a máquina realmente não tenha nenhuma porta aberta. Se a maioria das portas rastreadas(scaned) estiver fechada mas uns poucos números de portas comuns (tais como 22, 25, 53) estão filtrados, o sistema muito provavelmente está vulnerável. De vez em quando os sistemas irão mostrar exatamente o comportamento oposto. Se o seu rastreio(scan) mostrar 1000 portas abertas e 3 fechadas ou filtradas, então essas três podem muito bem ser as verdadeiramente abertas.

−sM (rastreio(scan) TCP Maimon)
O rastreio(scan) Maimon recebeu o nome de seu descobridor, Uriel Maimon. Ele descreveu a técnica na Phrack Magazine, edição 49 (Novembro de 1996). O Nmap, que incluiu essa técnica, foi lançado duas edições mais tarde. A técnica é exatamente a mesma que os rastreios(scans) Null, FIN e Xmas, exceto que a sondagem é FIN/ACK. De acordo com a RFC 793 (TCP) um pacote RST deveria ser gerado em resposta a tal sondagem se a porta estiver aberta ou fechada. Entretanto, Uriel notou que muitos sistemas derivados do BSD simplesmente descartavam o pacote se a porta estivesse aberta.

−−scanflags (rastreio(scan) TCP Customizado)
Usuários verdadeiramente avançados do Nmap não precisam se limitar aos tipos de rastreios(scans) enlatados oferecidos. A opção **−−scanflags** permite que desenhe seu próprio rastreio(scan) permitindo a especificação de flags TCP arbitrárias. Deixe sua imaginação correr solta enquanto dribla sistemas de detecção de intrusão cujos fabricantes apenas olharam rapidamente a página man do Nmap adicionando regras específicas!

O argumento do **−−scanflags** pode ser um valor numérico da marca (flag) como o 9 (PSH e FIN), mas usar nomes simbólicos é mais fácil. Apenas esprema alguma combinação de URG, ACK, PSH, RST, SYN, e FIN. Por exemplo, **−−scanflags URGACKPSHRSTSYNFIN** marca tudo, embora não seja muito útil para rastreio(scan). A ordem em que essas marcas são especificadas é irrelevante.

Além de especificar as marcas desejadas pode especificar um tipo de rastreio(scan) TCP (como o **−sA**

ou **−sF**). Esse tipo−base diz ao Nmap como interpretar as respostas. Por exemplo, um rastreio(scan) SYN considera nenhuma−resposta como uma indicação de porta filtrada enquanto que um rastreio(scan) FIN trata a mesma como aberta|filtrada. O Nmap irá se comportar da mesma forma que o tipo de rastreio(scan)−base escolhido, excepto que ele irá usar as marcas TCP que especificar. Se não escolher um tipo−base, o rastreio(scan) SYN é utilizado.

−sI <hostzumbi[:portadesondagem]> (rastreio(scan) Idle)
Este método avançado de rastreio(scan) permite um rastreio(scan) TCP realmente cego das portas do alvo (significando que nenhum pacote é enviado para o alvo do seu endereço IP real). Ao invés disso um ataque canal−lateral (side−channel) explora a previsível geração de sequência de ID, consequência da fragmentação do IP no anfitrião(host) zumbi, para juntar informações sobre as portas abertas no alvo. Sistemas IDS irão mostrar o rastreio(scan) como se viessem da máquina zumbi que especificou (que deve estar activa e obedecer a alguns critérios). Este tipo fascinante de rastreio(scan) é complexo demais para se descrever completamente aqui neste guia de referência, então eu escrevi e postei um trabalho informal com detalhes completos em **https://nmap.org/book/idlescan.html**.

Além de ser extraordinariamente camuflado (devido à sua natureza cega), este tipo de rastreio(scan) permite mapear relações de confiança baseadas em IP entre máquinas. A listagem de portas mostra as portas abertas *da perspectiva do anfitrião(host) zumbi*. Portanto pode tentar rastrear(scan) um alvo usando vários zumbis que acha que podem ser confiáveis (via regras de router/filtro de pacotes).

Você pode adicionar os dois−pontos seguindo do número da porta ao nome do anfitrião(host) zumbi, se quiser sondar uma porta em particular no zumbi verificando as mudanças de IPID. Do contrário o Nmap irá utilizar a porta que ele normalmente usa por default para pings tcp (80).

−sO (Rastreios(Scans) do protocolo IP)
Scans do Protocolo IP permitem que determine quais protocolos IP (TCP, ICMP, IGMP, etc.) são suportados pelas máquina−alvo. Isso não é tecnicamente um rastreio(scan) de portas, pois ele varia os números do protocolo IP ao invés dos números de portas TCP e UDP. Ainda assim, ele utiliza a opção **−p** para seleccionar os números de protocolos a rastrear(scan), mostra os resultados dentro do formato normal da tabela de portas e até usa o mesmo mecanismo de rastreio(scan) dos métodos de descoberta de portas. Portanto ele é parecido o suficiente com um rastreio(scan) de portas e por isso pertence a este lugar.

Além de ser útil de certa forma, o rastreio(scan) de protocolo mostra o poder do software de código aberto. Embora a idéia fundamental seja bastante simples, eu não tinha pensado em adicioná−la e nem havia recebido nenhuma solicitação para essa funcionalidade. Então, no verão de 2000, Gerhard Rieger concebeu a idéia, escreveu uma excelente alteração (patch) implementando−a e enviou−a para a lista de discussão nmap−hackers. Eu incorporei a alteração na árvore do Nmap e lancei uma nova versão no dia seguinte. Poucos produtos de software comercial tem usuários entusiasmados o suficiente para desenhar e contribuir com melhorias!

O rastreio(scan) de protocolo funciona de uma forma similar a um rastreio(scan) UDP. Ao invés de ficar repetindo alternando o campo de número de porta de um pacote UDP, ele envia cabeçalhos de pacote IP e faz a repetição alternando o campo de protocolo IP de 8 bits. Os cabeçalhos normalmente estão vazios, sem conter dados, nem mesmo o cabeçalho apropriado do suposto protocolo. As três excepções são o TCP, o UDP e o ICMP. Um cabeçalho de protocolo apropriado para estes é incluído, uma vez que alguns sistemas não os enviarão caso não tenham e porque o Nmap tem as funções para criá−los ao invés de observar as mensagens de erro ICMP de porta inalcançável, o rastreio(scan) de protocolo fica de olho nas mensagens ICMP de *protocolo* inalcançável. Se o Nmap recebe qualquer resposta de qualquer protocolo do anfitrião(host)−alvo, o Nmap marca esse protocolo como aberto. Um erro ICMP de protocolo não−alcançável (tipo 3, código 2) faz com que o protocolo seja marcado como fechado. Outros erros ICMP do tipo inalcançável (tipo 3, código 1, 3, 9, 10, ou 13) fazem com que o protocolo seja marcado como filtrado (embora eles provem, ao mesmo tempo, que o ICMP está aberto). Se nenhuma resposta for recebida após as retransmissões, o protocolo é marcado como aberto|filtrado.

−b <anfitrião(host) para relay de ftp> (Rastreio(Scan) de FTP bounce)
Uma característica interessante do protocolo FTP (**RFC 959**[5]) é o suporte a conexões denominadas proxy ftp. Isso permite que um usuário conecte−se a um servidor FTP e então solicite que arquivos sejam enviados a um terceiro servidor. Tal característica é sujeita a abusos em diversos níveis, por isso a maioria dos servidores parou de suportá−la. Um dos abusos permitidos é fazer com que o servidor FTP efectue o rastreio(scan) das portas de outros anfitriões(hosts). Simplesmente solicite que o servidor FTP envie um arquivo para cada porta interessante do anfitrião(host)−alvo. A mensagem de erro irá descrever se a porta está aberta ou não. Esta é uma boa forma de passar por cima de firewalls porque os servidores FTP de empresas normalmente são posicionados onde tem mais acesso a outros anfitriões(hosts) internos que os velhos servidores da Internet teriam. O Nmap suporta o rastreio(scan) de ftp bounce com a opção **−b**. Ela recebe um argumento no formato *nomedousuário*:*senha@servidor*:*porta*. *Servidor* é o nome ou endereço IP de um servidor FTP vulnerável. Assim como em uma URL normal, pode omitir *nomedousuário*:*senha*, neste caso as credenciais de login anónimo (usuário: anonymous senha:−wwwuser@) serão usados. O número da porta (e os dois−pontos) podem ser omitidos, e então a porta FTP default (21) no *servidor* será utilizada.

Esta vulnerabilidade espalhou−se em 1997 quando o Nmap foi lançado mas foi corrigida amplamente. Servidores vulneráveis ainda estão por aí, então pode valer a pena tentar se tudo o mais falhar. Se passar por cima de um firewall é o seu objetivo, faça o rastreio(scan) da rede−alvo procurando por uma porta 21 aberta (ou mesmo por qualquer serviço FTP se rastrear(scan) todas as portas com a detecção de versão), então tente um rastreio(scan) bounce usando−as. O Nmap irá dizer se o anfitrião(host) é vulnerável ou não. Se estiver apenas tentando encobrir suas pegadas, não precisa (e, na verdade, não deveria) limitar−se a anfitriões(hosts) na rede−alvo. Antes de sair rastreando endereços aleatórios na Internet procurando por servidores FTP, considere que os administradores de sistemas podem não apreciar o seu abuso nos servidores deles.

ESPECIFICAçãO DE PORTAS E ORDEM DE SCAN

Somado a todos os métodos de rastreio(scan) discutidos anteriormente, o Nmap oferece opções para especificar quais portas são rastreadas(scaned) e se a ordem de rastreio(scan) é aleatória ou sequencial. Por default, o Nmap rastreia(scan) todas as portas até, e incluindo, 1024, bem como portas com numeração alta listadas no arquivo nmap−services para o(s) protocolo(s) rastreados(scaned).

−p <faixa de portas> (Rastreia apenas as portas especificadas)
Esta opção especifica quais as portas que deseja rastrear(scan) e prevalece sobre o default. Números de portas individuais são OK, bem como as faixas separadas por um hífen (p.ex.: 1−1023). Os valores iniciais e/ou finais da faixa podem ser omitidos, o que faz com que o Nmap use 1 e 65535 respectivamente. Portanto pode especificar **−p−** para rastrear(scan) as portas de 1 até 65535. Escanear a porta zero é permitido se especificar explicitamente. Para o rastreio(scan) do protocolo IP (**−sO**), esta opção especifica os números dos protocolos que deseja rastrear(scan) (0−255).

Quando rastrear(scan) ambas as portas TCP e UDP, pode especificar um protocolo em particular precedendo os números de portas com T: ou U:. O qualificador dura até que especifique um novo qualificador. Por exemplo, o argumento **−p U:53,111,137,T:21−25,80,139,8080** faria o rastreio(scan) das portas UDP 53, 111 e 137, bem como as portas TCP listadas. Note que para rastrear(scan) ambas as portas UDP & TCP, tem que especificar **−sU** e pelo menos um tipo de rastreio(scan) TCP (tal como **−sS**, **−sF** ou **−sT**). Se nenhum qualificador de protocolo for informado, os números de portas serão acrescentados à todas as listas de protocolos.

−F (rastreio(scan) Rápido (portas limitadas))
Especifica que deseja apenas rastrear(scan) as portas listadas no arquivo nmap−services que vem com o nmap (ou o arquivo de protocolos para o **−sO**). Isto é muito mais rápido do que rastrear(scan) todas as 65535 portas de um anfitrião(host). Pelo facto desta lista conter tantas portas TCP (mais de 1200), a diferença de velocidade de um rastreio(scan) TCP default (cerca de 1650 portas) não é dramática. A diferença pode ser enorme se especificar seu próprio minúsculo arquivo nmap−services usando a opção **−−datadir**.

−r (Não usa as portas de forma aleatória)
Por default o Nmap usa a ordem das portas a serem rastreadas de forma aleatória (excepto aquelas portas normalmente acessíveis que são movidas próximas ao início por motivos de eficiência). Essa técnica de busca aleatória normalmente é desejável mas pode especificar **−r** para um rastreio(scan) de portas sequencial.

DETECçãO DE SERVIçO E VERSãO

Aponte o Nmap para uma máquina remota e ele poderá lhe dizer que as portas 25/tcp, 80/tcp e 53/udp estão abertas. Utilizar o banco de dados nmap−services com cerca de 2.200 serviços bastante conhecidos do Nmap iria relatar que aquelas portas provavelmente correspondem a um servidor de correio eletrônico (SMTP), a um servidor de páginas web (HTTP) e a um servidor de nomes (DNS) respectivamente. Essa pesquisa normalmente é precisa −− a grande maioria de daemons escutando na porta TCP 25 é de facto de servidores de correio eletrónico. Entretanto não deveria apostar a sua segurança nesta informação! As pessoas podem e executam serviços em portas estranhas.

Mesmo que o Nmap esteja certo e o servidor hipotético acima esteja executando os serviços SMTP, HTTP e DNS, isso não é informação o bastante. Quando fizer uma avaliação de vulnerabilidades (ou mesmo um simples inventário da rede) de sua empresa ou clientes, realmente deseja saber qual o programa−servidor de correio eletrónico ou de nomes e as versões que estão rodando. Ter um número de versão exacto ajuda substancialmente na determinação de quais explorações (exploits) o servidor está vulnerável. A detecção de versão ajuda a obter esta informação.

Depois que as portas TCP e/ou UDP forem descobertas usando qualquer um dos outros métodos de rastreio(scan), a detecção de versão interroga essas portas para determinar mais informações sobre o que realmente sendo executado nessas portas. O banco de dados nmap−service−probes do Nmap contém sondagens para pesquisar diversos serviços e expressões de acerto (match expressions) para reconhecer e destrinchar as respostas. O Nmap tenta determinar os protocolos de serviços (p.ex.: ftp, ssh, telnet, http), o nome da aplicação (p.ex.: ISC Bind, Apache httpd, Solaris telnetd), o número da versão, o nome do anfitrião(host), tipo de dispositivo (p.ex.: impressora, router), a família do SO (p.ex.: Windows, Linux) e às vezes detalhes diversos do tipo, se um servidor X está aberto para conexões, a versão do protocolo SSH ou o nome do usuário do KaZaA. É claro que a maioria dos serviços não fornece todas essas informações. Se o Nmap foi compilado com o suporte ao OpenSSL ele irá se conectar aos servidores SSL para deduzir qual o serviço que está escutando por trás da camada criptografada. Quando os serviços RPC são descobertos, o "amolador" de RPC (RPC grinder) do Nmap (**−sR**) é automaticamente utilizado para determinar o nome do programa RPC e o número da versão. Algumas portas UDP são deixadas no estado aberta|filtrada depois que rastreio(scan) de porta UDP não consegue determinar se a porta está aberta ou filtrada. A detecção de versão irá tentar provocar uma resposta dessas portas (do mesmo jeito que faz com as portas abertas) e alterar o estado para aberta se conseguir. Portas TCP do tipo aberta|filtrada são tratadas da mesma forma. Note que a opção **−A** do Nmap habilita a detecção de versão entre outras coisas. Um trabalho documentando o funcionamento, uso e customização da detecção de versão está disponível em **http://www.insecure.org/nmap/versionscan.html**.

Quando o Nmap recebe uma resposta de um serviço mas não consegue encontrá−la em seu banco de dados, ele mostra uma identificação (fingerprint) especial e uma URL para que envie informações se souber com certeza o que está rodando nessa porta. Por favor considere dispor de alguns minutos para mandar essa informação de forma que sua descoberta possa beneficiar a todos. Graças a esses envios o Nmap tem cerca de 3.000 padrões de acerto para mais de 350 protocolos, tais como o smtp, ftp, http, etc.

A detecção de versão é habilitada e controlada com as seguintes opções:

−sV (detecção de versão)
Habilita a detecção de versão, conforme discutido acima. Alternativamente pode usar a opção **−A** para habilitar tanto a detecção de SO como a detecção de versão.

−−allports (Não exclui nenhuma porta da detecção de versão)
Por default a detecção de versão do Nmap salta a porta TCP 9100 por causa de algumas impressoras que imprimem qualquer coisa que seja enviada para essa porta, levando a dezenas de páginas com requisições HTTP, requisições de sessões SSL binárias, etc. Esse comportamento pode ser alterado modificando−se ou removendo a directiva Exclude no nmap−service−probes ou pode especificar

−−**allports** para rastrear(scan) todas as portas independente de qualquer directiva Exclude.

−−**version−intensity <intensidade>** (Estabelece a intensidade do rastreio(scan) de versão)
Quando está executando um rastreio(scan) de versão (−**sV**) o nmap envia uma série de sondagens, cada qual com um valor atribuído de raridade, entre 1 e 9. As sondagens com números baixos são efectivas contra uma ampla variedade de serviços comuns, enquanto as com números altos são raramente úteis. O nível de intensidade especifica quais sondagens devem ser utilizadas. Quando mais alto o número, maiores as chances de o serviço ser corretamente identificado. Entretanto rastreios(scans) de alta intensidade levam mais tempo. A intensidade deve estar entre 0 e 9. O default é 7. Quando uma sondagem é registrada na porta−alvo através da directiva nmap−service−probesports, essa sondagem é tentada independentemente do nível de intensidade. Isso assegura que as sondagens DNS sempre serão tentadas contra qualquer porta 53 aberta e a sondagem SSL será realizada contra a 443, etc.

−−**version−light** (Habilita o modo leve (light))
Esse é um apelido conveniente para −−**version−intensity 2**. Esse modo leve torna o rastreio(scan) de versão muito mais rápido, mas é ligeiramente menos provável que identifique os serviços.

−−**version−all** (Tenta simplesmente todas as sondagens)
Um apelido para −−**version−intensity 9**, assegurando que todas as sondagens sejam tentadas contra cada porta.

−−**version−trace** (Monitora as atividades do rastreio(scan) de versão)
Isto faz com que o Nmap mostre informações de depuração extensivas sobre o que o rastreio(scan) de versão está fazendo. É um sub−conjunto do que obteria com −−**packet−trace**.

−**sR** (Scan RPC)
Este método trabalha em conjunto com os vários métodos de rastreio(scan) de portas do Nmap. Ele pega todas as portas TCP/UDP descobertas no estado aberta e inunda−as com comandos NULL do programa SunRPC em uma tentativa de determinar se elas são portas RPC e se forem, quais programas e números de versão elas mostram. Dessa forma pode obter efectivamente a mesma informação que o **rpcinfo −p** mesmo se o portmapper do alvo estiver atrás de um firewall (ou protegido por TCP wrappers). Chamarizes não funcionam ainda com o rastreio(scan) RPC. Isso é habilitado automaticamente como parte do rastreio(scan) de versão (−**sV**) se o solicitar. Como a detecção de versão inclui isso e é muito mais abrangente, o −**sR** raramente é necessário.

DETECçãO DE SO

Uma das características mais conhecidas do Nmap é a detecção remota de SO utilizando a identificação da pilha (stack fingerprinting) do TCP/IP. O Nmap envia uma série de pacotes TCP e UDP ao anfitrião(host) remoto e examina praticamente todos os bits das respostas. Após executar dezenas de testes como a amostragem TCP ISN, suporte e ordenamento das opções do TCP, amostragem IPID e a observação do tamanho inicial da janela, o Nmap compara os resultados com o banco de dados nmap−os−fingerprints com mais de 1500 identificações de SO conhecidas e mostra os detalhes do SO se houver uma correspondência. Cada identificação inclui uma descrição textual livre do SO e uma classificação que fornece o nome do fabricante (p.ex.: Sun), SO base (p.ex.: Solaris), geração do SO (p.ex.: 10) e tipo de dispositivo (genérico, router, switch, consola de jogo, etc.).

Se o Nmap não conseguir identificar o SO da máquina e as condições forem favoráveis (p.ex.: pelo menos uma porta aberta e uma porta fechada foram encontradas), o Nmap irá fornecer uma URL onde poderá enviar a identificação se souber (com certeza) o SO em execução na máquina. Fazendo isso, contribui para o pool de sistemas operacionais conhecidos pelo Nmap e, com isso, ele será mais preciso para todos.

A detecção de SO habilita diversos outros testes que usam as informações coletadas durante o processo. Um deles é a medição de uptime, que utiliza a opção timestamp do TCP (RFC 1323) para supor quando uma máquina foi reiniciada pela última vez. Isso apenas é mostrado para as máquinas que fornecem essa informação. Outro é a Classificação de Previsibilidade da Seqüencia do TCP. Ele mede aproximadamente o grau de dificuldade de se estabelecer uma conexão TCP forjada contra um anfitrião(host) remoto. É útil para se explorar relações de confiança baseadas no IP de origem (rlogin, filtros de firewall, etc.) ou para ocultar a origem de um ataque. Esse tipo de enganação (spoofing) raramente é executada hoje em dia, mas

muitas máquinas ainda estão vulneráveis a ele. O número de dificuldade real é baseado em amostragens estatísticas e pode variar. Normalmente é melhor usar a classificação em inglês, do tipo “worthy challenge” (um desafio que vale a pena) ou “trivial joke” (uma piada, muito fácil). Isso só é mostrado na saída normal do modo verbose (**−v**). Quando o modo verbose é habilitado juntamente com o **−O**, a Geração de Seqüencia IPID também é mostrada. A maioria das máquinas é classificada como “incremental” , o que significa que elas incrementam o campo ID no cabeçalho IP para cada pacote que envia. Isso torna−as vulnerável a diversos ataques avançados de levantamento e forjamento de informações.

Um trabalho documentando o funcionamento, utilização e customização da datecção de versão está disponível em mais de uma dezena de línguas em **http://www.insecure.org/nmap/osdetect/**.

A deteção de SO é habilitada e controlada com as seguintes opções:

−O (Habilita a detecção de SO)
Habilita a deteção de SO como discutido acima. Alternativamente pode usar **−A** para habilitar tanto a detecção de SO quanto a detecção de versão.

−−osscan−limit (Limitar a detecção de SO a alvos promissores)
A detecção de SO é bem mais eficiente se ao menos uma porta TCP aberta e uma fechada for encontrada. Escolha esta opção e o Nmap não irá nem tentar a detecção de SO contra anfitriões(hosts) que não correspondam a este critério. Isso pode economizar um tempo considerável, particularmente em rastreios(scans) **−P0** contra muitos anfitriões(hosts). Isso só importa quando a detecção de SO é solicitada através de **−O** ou **−A**.

−−osscan−guess; **−−fuzzy** (Resultados de tentativas de detecção de SO)
Quano o Nmap não é capaz de detectar uma correspondência exacta de SO, às vêzes ele oferece possibilidades aproximada. A correspondência tem que ser muito próxima para o Nmap fazer isso por default. Qualquer uma dessas opções (equivalentes) tornam as tentativas do Nmap mais agressivas.

TEMPORIZAçãO (TIMING) E DESEMPENHO

Uma das minhas mais altas prioridades no desenvolvimento do Nmap tem sido o desempenho. Um rastreio(scan) default (**nmap** *hostname*) de um anfitrião(host) em minha rede local leva apenas um quinto de segundo. Isso mal dá tempo de piscar o olho, mas esse tempo conforme está rastreando dezenas ou centenas de milhares de anfitriões(hosts). Além disso, certos tipos de rastreio(scan) como o rastreio(scan) UDP ou a detecção de versão, aumentam o tempo de rastreio(scan) substancialmente. Da mesma forma algumas configurações de firewall fazem o mesmo, particularmente quando limitam a taxa de resposta. Embora o Nmap se utilize de paralelismo e muitos outros algoritmos avançados para acelerar esses rastreios(scans) o usuário tem o controle final sobre como o Nmap executa. Usuários avançados elaboram comandos do Nmap cuidadosamente para obter apenas as informações que importam, sempre se preocupando com as restrições de tempo.

Técnicas para melhorar os tempos de rastreio(scan) incluem omitir testes não−críticos e atualizar até a versão mais recente do Nmap (melhorias de desempenho são feitas freqüentemente). Otimizar os parâmetros de tempo também podem fazer uma grande diferença. Essas opções estão listadas abaixo.

−−min−hostgroup <milissegundos>; **−−max−hostgroup <milissegundos>** (Ajuste dos tamanhos dos grupos de rastreio(scan) paralelos)
O Nmap tem a habilidade de fazer um rastreio(scan) de portas ou de versões em múltiplos anfitriões(hosts) em paralelo. O Nmap faz isso dividindo a faixa de endereços IP−alvo em grupos e então rastreando um grupo de cada vez. No geral grupos maiores são mais eficientes. A contrapartida é que os resultados dos anfitriões(hosts) não pode ser fornecido até que o grupo inteiro tenha terminado. Portanto se o Nmap começou com um tamanho de grupo igual a 50, o usuário não receberia nenhum relatório (exceto pelas atualizações mostradas no modo verbose) até que os primeiros 50 anfitriões(hosts) tivessem completado.

Por default, o Nmap assume um compromisso para resolver esse conflito. Ele começa com um tamanho de grupo pequeno, igual a cinco, para que os primeiros resultados venham rápido e então aumenta o tamanho até que chegue em 1024. O número default exacto depende das opções fornecidas. Por questões de eficiência o Nmap usa tamanhos de grupo maiores para o UDP ou para

rastreios(scans) TCP com poucas portas.

Quando o tamanho de grupo máximo é especificado com **−−max−hostgroup**, o Nmap nunca irá exceder esse tamanho. Especifique um tamanho mínimo com **−−min−hostgroup** e o Nmap irá tentar manter o tamanho dos grupos acima desse nível. O Nmap pode ter que usar tamanhos menores do que especificou, se não houverem anfitriões(hosts)−alvo suficientes restando em uma dada interface para completar o mínimo especificado. Ambos podem ser configurados para manter o tamanho do grupo dentro de uma faixa específica, embora isso raramente seja desejado.

O uso primário destas opções é especificar um tamanho de grupo mínimo grande de forma que o rastreio(scan) completo seja executado mais rapidamente. Uma escolha comum é 256 para rastrear(scan) uma rede em blocos de tamanho Classe C. Para um rastreio(scan) com muitas portas exceder esse número não irá ajudar muito. Para rastreios(scans) com poucos números de portas um tamanho de grupo de anfitriões(hosts) de 2048 ou mais pode ser útil.

−−min−parallelism <milissegundos>; **−−max−parallelism <milissegundos>** (Ajuste da paralelização das sondagens)

Estas opções controlam o número total de sondagens que podem estar pendentes para um grupo de anfitriões(hosts). Elas são usadas para o rastreio(scan) de portas e para a descoberta de anfitriões(hosts). Por default o Nmap calcula um paralelismo ideal e constantemente actualizado baseado no desempenho da rede. Se os pacotes estiverem sendo descartados o Nmap reduz o ritmo e liberta menos sondagens pendentes. O número de sondagens ideal aumenta vagarosamente conforme a rede se mostre mais confiável. Estas opções estabelecem limites mínimo e máximo nessa variável. Por default o paralelismo ideal pode cair até 1 se a rede se mostrar não−confiável e subir até diversas centenas em condições perfeitas.

O uso mais comum é estabelecer **−−min−parallelism** em um número maior que um para melhorar a velocidade dos rastreios(scans) de anfitriões(hosts) ou redes com desempenho ruim. Esta é uma opção arriscada para se ficar brincando pois configurar um valor alto demais pode afetar a precisão. Configurar isso também reduz a habilidade do Nmap de controlar o paralelismo dinamicamente baseado nas condições da rede. Um valor igual a dez pode ser razoável, embora eu só ajuste esse valor como última alternativa.

A opção **−−max−parallelism** às vêzes é configurada para evitar que o Nmap envie aos anfitriões(hosts) mais do que uma sondagem de cada vez. Isso pode ser útil em conjunto com **−−scan−delay** (discutido mais tarde), embora esta última normalmente sirva bem ao propósito por si só.

−−min_rtt_timeout <milissegundos>, **−−max−rtt−timeout <milissegundos>**, **−−initial−rtt−timeout <milissegundos>** (Ajuste de tempo de expiração (timeouts) das sondagens)

O Nmap mantém um valor de tempo de expiração (timeout) de execução para determinar quanto tempo ele deve esperar por uma resposta de uma sondagem antes de desistir ou retransmitir essa sondagem. Isso é calculado com base nos tempos de resposta de sondagens anteriores. Se a lentidão da rede se mostra significativa e variável esse tempo de expiração pode subir para vários segundos. Ele também começa com um nível conservador (alto) e pode ficar desse jeito por um tempo enquanto o Nmap rastreia(scan) anfitriões(hosts) não−responsivos.

Estas opções recebem um valor em milissegundos. Especificar um **−−max−rtt−timeout** e **−−initial−rtt−timeout** mais baixos que o default pode reduzir o tempo de rastreio(scan) significativamente. Isso é particularmente verdade para rastreios(scans) sem ping (**−P0**) e para aqueles contra redes bastante filtradas. Mas não se torne muito agressivo. O rastreio(scan) pode acabar levando mais tempo se especificar um valor tão baixo que muitas sondagens irão expirar o tempo e serem retransmitidas enquanto a resposta ainda está em trânsito.

Se todos os anfitriões(hosts) estão em uma rede local, 100 milissegundos é um valor de **−−max−rtt−timeout** razoavelmente agressivo. Se houver roteamento envolvido faça um ping de um

anfitrião(host) da rede primeiro com o utilitário ICMP ping ou com um formatador de pacotes customizados como o hping2, que pode passar por um firewall mais facilmente. Descubra o tempo máximo de round trip em dez pacotes mais ou menos. Coloque o dobro desse valor em **−−initial−rtt−timeout** e o triplo ou quádruplo para o **−−max−rtt−timeout**. Normalmente eu não configuro o rtt máximo abaixo de 100ms, não importa quais os tempos de ping. Eu também não excedo o valor 1000ms.

−−min_rtt_timeout é uma opção raramente utilizada que poderia ser útil quando uma rede é tão não−confiável que mesmo o default do Nmap é muito agressivo. Considerando que o Nmap apenas reduz o tempo de expiração para um valor mínimo quando a rede parece ser confiável, esta necessidade não é comum e deveria ser reportada à lista de discussão nmap−dev como um bug.

−−host−timeout <milissegundos> (Desiste em anfitriões(hosts)−alvo lentos)
Alguns anfitriões(hosts) simplesmente levam tempo *demais* para serem rastreados. Isso pode ser causado por um hardware ou software de rede com fraco desempenho ou pouco confiável, limitação na taxa dos pacotes ou por um firewall restritivo. Os poucos anfitriões(hosts) mais lentos de todos os anfitriões(hosts) escaneados podem acabar sendo responsáveis pela maior parte do tempo total gasto com o rastreio(scan). Às vêzes é melhor cortar fora o prejuízo e saltar esses anfitriões(hosts) logo no início. Isso pode ser feito especificando **−−host−timeout** com o número de milissegundos que tolera esperar. Eu normalmente especifico 1800000 para ter certeza de que o Nmap não irá gastar mais do que meia hora em um único anfitrião(host). Note que o Nmap pode estar escaneando outros anfitriões(hosts) ao mesmo tempo em que essa meia hora desse único anfitrião(host) está correndo, então não é uma perda de tempo total. Um anfitriões(hosts) que expira o tempo é saltado. Nenhum resultado de tabela de portas, detecção de SO ou detecção de versão é mostrado para esse anfitrião(host).

−−scan−delay <milissegundos>; **−−max_scan−delay <milissegundos>** (Ajusta o atraso entre sondagens)
Esta opção faz com que o Nmap aguarde um determinado número de milissegundos entre cada sondagem enviada a um dado anfitrião(host). Isto é particularmente útil no caso de limitação de taxas de transferência. Máquinas Solaris (entre muitas outras) irão normalmente responder a pacotes de sondagens de rastreios(scans) UDP com apenas uma mensagem ICMP por segundo. Qualquer número maior que isso, enviado pelo Nmap, será um desperdício. Um **−−scan−delay** de 1000 irá manter uma taxa de transferência baixa. O Nmap tenta detectar a limitação de taxa e ajusta o atraso no rastreio(scan) de acordo, mas não dói especificar explicitamente se já sabe qual a taxa que funciona melhor.

Outro uso do **−−scan−delay** é para evitar os sistemas de prevenção e deteção de intrusão (IDS/IPS) baseados em limites.

−T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> (Estabelece um padrão de temporização)
Embora os controles de temporização de ajuste fino discutidos nas seções anteriores sejam poderosos e efectivos, algumas pessoas consideram−nos confusos. Escolher os valores apropriados pode às vêzes levar mais tempo do que o próprio rastreio(scan) que está tentando optimizar. Por isso o Nmap oferece uma aproximação mais simples com seis padrões de temporização. Você pode especificá−los com a opção **−T** e os números (0 − 5) ou os nomes. Os nomes de padrões são paranóico (paranoid, 0), furtivo (sneaky, 1), educado (polite, 2), normal (3), agressivo (agressive, 4) e insano (insane, 5). Os dois primeiros são para evitar um IDS. O modo educado (ou polido), diminui o ritmo de rastreio(scan) para usar menos banda e recursos da máquina alvo. O modo normal é o default e, portanto, **−T3** não faz nada. O modo agressivo acelera os rastreios(scans) assumindo que está em uma rede razoavelmente rápida e confiável. Finalmente, o modo insano assume que está em uma rede extraordinariamente rápida ou está disposto a sacrificar alguma precisão pela velocidade.

Estes padrões permitem que o usuário especifique o quão agressivo desejam ser, ao mesmo tempo que deixam o Nmap escolher os valores de temporização exactos. Os padrões também fazem ajustes pequenos na velocidade onde ainda não existem opções para controle de ajuste fino. Por exemplo, **−T4** proibe que o atraso dinâmico de rastreio(scan) exceda 10ms para portas TCP e **−T5** corta esse valor

para 5 milissegundos. Padrões podem ser utilizados em conjunto com controles de ajuste fino desde que o padrão seja especificado primeiramente. Do contrário os valores default para os padrões irão se sobrepor aos valores que especificar. Eu recomendo usar **−T4** quando rastrear(scan) redes razoavelmente modernas e confiáveis. Mantenha essa opção (no começo da linha de comando) mesmo que adicione controles de ajuste fino, de forma que possa se beneficiar com as pequenas otimizações extras que ela habilita.

Se tiver uma conexão ethernet ou de banda−larga decente, eu recomendaria sempre utilizar **−T4**. Algumas pessoas adoram o **−T5** embora seja agressivo demais para o meu gosto. As pessoas às vêzes especificam **−T2** porque acham que diminui a probabilidade de travar os anfitriões(hosts) ou porque elas consideram−se educadas em geral. Normalmente elas não percebem o quão lento o **−T Polite** realmente é. Esses rastreios(scans) podem levar dez vêzes mais tempo que um rastreio(scan) default. Travamento de máquinas e problemas com a banda são raros com as opções de temporização default (**−T3**) e portanto, eu normalmente as recomendo para escaneadores precavidos. Omitir a detecção de versão é bem mais eficaz do que ficar brincando com os valores de temporização para reduzir esses problemas.

Embora o **−T0** e o **−T1** possam ser usados para evitar alertas no IDS, eles irão leva muito mais tempo para rastrear(scan) milhares de máquinas ou portas. Para um rastreio(scan) tão amplo prefira estabelecer os valores exatos de temporização que precisa ao invés de depender dos valores "engessados" de **−T0** e **−T1**.

Os principais efeitos de **T0** é serializar o rastreio(scan) de forma que apenas uma porta é rastreada de cada vez e então aguardar cinco minutos entre o envio de cada sondagem. **T1** e **T2** são similares mas aguardam apenas 15 segundos e 0,4 segundos, respectivamente, entre as sondagens. **T3** é o comportamento default do Nmap, que inclui o paralelismo. **T4** faz o mesmo que **−−max−rtt−timeout 1250 −−initial−rtt−timeout 500** e estabelece o atraso máximo de rastreio(scan) TCP em 10 milissegundos. **T5** faz o mesmo que **−−max−rtt−timeout 300 −−min_rtt_timeout 50 −−initial−rtt−timeout 250 −−host−timeout 900000** e estabelece o atraso máximo de rastreio(scan) TCP em 5ms.

EVITANDO E ENGANANDO O FIREWALL/IDS

Muitos pioneiros da Internet vislumbraram uma rede mundial aberta com um espaço de endereçamento IP universal que permitisse conexões virtuais entre quaisquer dois nós. Isso permite que os anfitriões(hosts) actuem como verdadeiros semelhantes, servindo e obtendo informações uns dos outros. As pessoas poderiam aceder a seus computadores domésticos do trabalho, mudando os ajustes do controle de climatização ou abrindo as portas para convidados. Essa visão de conectividade universal foi sufocada pela falta de espaço de endereçamento e preocupações com a segurança. No início dos anos 1990 as empresas começaram a instalar firewalls para o propósito claro de reduzir a conectividade. Rede enormes foram isoladas da Internet−sem−fronteiras por proxies de aplicativos, tradução de endereçamento de rede (network address translation) e filtros de pacotes. O fluxo irrestrito de informações deu a vez à regulamentação acirrada de canais de comunicação autorizados e ao conteúdo que neles trafegam.

As obstruções de rede como o firewall podem tornar o mapeamente de uma rede extremamente difícil. E isso não vai se tornar mais fácil, pois sufocar as sondagens casuais é freqüentemente o objetivo principal de se instalar esses dispositivos. Apesar disso o Nmap oferece muitas ferramentas para ajudar a entender essas redes complexas e para verificar que os filtros estão funcionando como esperado. Ele até suporta mecanismos para passar por cima de defesas mal implementadas. Um dos melhores métodos para se entender a postura de segurança de uma rede é tentar derrubá−la. Pense com a mente de uma pessoa que quer atacá−lo e aplique técnicas desta seção contra a sua rede. Lance um rastreio(scan) FTP bounce, um rastreio(scan) idle, um ataque de fragmentação ou tente "tunelar" (criar um túnel) através de um de seus próprios proxies.

Além de restringir a atividade de rede as empresas estão monitorando o tráfego cada vez mais com sistemas de detecção de intrusão (IDS). Todos os principais IDS vêm com regras designadas para detectar rastreios(scans) feitos com o Nmap porque os rastreios(scans) são, às vêzes, precursores de ataques. Muitos

desses produtos foram recentemente metamorfoseados em sistemas de *prevenção* de intrusão (IPS) que bloqueiam o tráfego considerado malicioso de forma activa. Infelizmente para administradores de rede e vendedores de IDS, detectar confiavelmente as más intenções através da análise de dados de pacotes é um problema difícil. Atacantes com paciência, habilidade e a ajuda de certas opções do Nmap podem normalmente passar por um IDS sem serem detectados. Enquanto isso, os administradores devem lidar com um alto número de resultados do tipo falso−positivo onde actividades inocentes são diagnosticadas erradamente e recebem alertas ou são bloqueadas.

De vez em quando as pessoas sugerem que o Nmap não deveria oferecer opções que permitam evitar as regras de firewalls ou passar desapercebidos por IDSs. Elas argumentam que essas características são tão sujeitas à má−utilização por atacantes quanto são utilizadas por administradores para aumentar a segurança. O problema com esta lógica é que esses métodos ainda assim seriam utilizados pelos atacantes que encontrariam outras ferramentas ou então acrescentariam essa funcionalidade no Nmap. Enquanto isso os administradores achariam muito mais difícil executar suas tarefas. Instalar apenas servidores FTP modernos e corrigidos é uma defesa muito melhor do que tentar evitar a distribuição de ferramentas que implementem o ataque FTP bounce.

Não existe uma carta mágica (ou opção do Nmap) para detectar e subverter firewalls e sistemas IDS. É necessário habilidade e experiência. Um tutorial está além do objectivo deste guia de referência que apenas lista as opções relevantes e descreve suas funções.

−f (fragmenta os pacotes); **−−mtu** (usando a MTU especificada)

A opção **−f** faz com que o rastreio(scan) solicitado (incluindo rastreios(scans) usando ping) utilize pequenos pacotes IP fragmentados. A idéia é dividir o cabeçalho TCP em diversos pacotes para tornar mais difícil para os filtros de pacotes, os sistemas de detecção de intrusão e outros aborrecimentos, detectar o que está fazendo. Tenha cuidado com isto! Alguns programas tem problemas para lidar com estes pequenos pacotes. O sniffer da velha−guarda chamado Sniffit sofria uma falha de segmentação assim que recebia o primeiro fragmento. Especifique esta opção uma vez e o Nmap dividirá os pacotes em 8 bytes ou menos após o cabeçalho IP. Portanto, um cabeçalho TCP de 20 bytes seria dividido em 3 pacotes. Dois com oito bytes do cabeçalho TCP e um com os quatro restantes. É claro que cada fragmento também tem um cabeçalho IP. Especifique **−f** novamente para usar 16 bytes por fragmento (reduzindo o número de fragmentos). Ou então, pode especificar o seu próprio tamanho de quebra com a opção **−−mtu**. Não especifique também o **−f** se usar o **−−mtu**. A quebra deve ser um múltiplo de 8. Embora os pacotes fragmentados não passem por filtros de pacotes e firewalls que enfilerem todos os fragmentos IP, tal como a opção CONFIG_IP_ALWAYS_DEFRAG do kernel do Linux faz, algumas redes não aguentam o impacto no desempenho que isso causa deixando a opção desabilitada. Outros não conseguem habilitar isso porque os fragmentos podem seguir por rotas diferentes na rede. Alguns sistemas de origem desfragmentam pacotes de saída no kernel. O Linux e o módulo de reastreamento de conexão do iptables é um exemplo desse tipo. Faça um rastreio(scan) enquanto executa um sniffer como o Ethereal para ter a certeza de que pacotes enviados estão fragmentados. Se o SO do seu anfitrião(host) estiver causando problemas tente a opção **−−send−eth** para passar por cima da camada IP e enviar frames ethernet em estado bruto(raw).

−D <chamariz1 [,chamariz2][,ME],...> (Disfarça um rastreio(scan) usando chamarizes)

Faz com que um rastreio(scan) com chamarizes seja executado, o que parece ao anfitrião(host) remoto que, o(s) anfitrião(host)(s) que especificou como chamarizes também estejam rastreando a rede−alvo. Com isso, o IDS poderá reportar 5 a 10 rastreios(scans) de portas de endereços IP únicos, mas não saberá qual IP estava realmente efectuado o rastreio(scan) e qual era um chamariz inocente. Embora isso possa ser desvendado através de rastreamento de caminho de router, descarte de respostas (response−dropping) e outros mecanismos activos, normalmente é uma técnica eficaz para esconder o seu endereço IP.

Separe cada anfitrião(host)−chamariz com vírgulas e pode opcionalmente usar ME como um dos chamarizes para representar a posição do seu endereço IP real. Se colocar ME na 6a. posição ou acima, alguns detectores de rastreio(scan) de portas comuns (como o excelente scanlogd da Solar Designer) pouco provavelmente irão mostrar o seu endereço IP. Se não utilizar o ME o nmap irá colocá−lo em uma posição aleatória.

Observe que os anfitriões(hosts) que utilizar como chamarizes devem estar activos ou poderá acidentamente inundar com SYN os seus alvos. Também será bastante fácil determinar qual é o anfitrião(host) que está a efectuar o rastreio(scan) se houver apenas um anfitrião(host) realmente activo na rede. Você pode preferir usar endereços IP ao invés de nomes (de forma que as redes chamarizes não vejam em seus logs dos servidores de nomes).

Chamarizes são utilizados tanto no rastreio(scan) com ping inicial (usando ICMP, SYN, ACK ou qualquer outro) como também durante a fase real de rastreio(scan) de portas. Chamarizes também são usados durante a detecção de SO remoto (**−O**). Chamarizes não funcionam com a detecção de versão ou com o rastreio(scan) TCP connect().

Vale a pena observar que usar chamarizes demais pode deixar seu rastreio(scan) lento e potencialmente até torná−lo menos preciso. Outra coisa, alguns provedores ISP irão filtrar os seus pacotes disfarçados mas muitos não restringem pacotes IP disfarçados.

−S <Endereço_IP> (Disfarça o endereço de origem)
Em algumas circunstâncias o Nmap pode não conseguir determinar o seu endereço de origem (o Nmap irá dizer se for esse o caso). Nesta situação use o **−S** com o endereço IP da interface que deseja utilizar para enviar os pacotes.

Outro uso possível para esta flag é para disfarçar o rastreio(scan) e fazer com que os alvos achem que *alguma outra pessoa* está fazendo o rastreio(scan). Imagine uma empresa que está constantemente sofrendo rastreios(scans) de portas de um concorrente! A opção **−e** normalmente seria requerida para este tipo de uso e **−P0** seria recomendável.

−e <interface> (Usa a interface especificada)
Diz ao Nmap qual interface deve ser utilizada para enviar e receber pacotes. O Nmap deveria ser capaz de detectar isto automaticamente mas ele informará se não conseguir.

−−source−port <númerodaporta>; −g <númerodaporta> (Disfarça o número de porta de origem)
Um erro de configuração surpreendentemente comum é confiar no tráfego com base apenas no número da porta de origem. É fácil entender como isso acontece. Um administrador configura um firewall novinho em folha só para ser inundado com queixas de usuários ingratos cujas aplicações param de funcionar. Em particular, o DNS pode parar de funcionar porque as respostas DNS UDP de servidores externos não conseguem mais entrar na rede. O FTP é outro exemplo comum. Em tranferências FTP activas o servidor remoto tenta estabelecer uma conexão de volta com o cliente para poder transferir o arquivo solicitado.

Soluções seguras para esses problemas existem frequentemente na forma de proxies no nível da aplicação ou módulos de firewall para análise de protocolo. Infelizmente também há soluções mais fáceis e inseguras. Observando que as respostas DNS chegam pela porta 53 e o FTP activo pela porta 20 muitos administradores caem na armadilha de apenas permitir tráfego vindo dessas portas. Eles normalmente assumem que nenhum atacante irá notar e explorar essas brechas no firewall. Em outros casos os administradores consideram isso uma medida provisória de curto prazo até que eles possam implementar uma solução mais segura. Normalmente ele esquecem−se de fazer as actualizações de segurança.

Administradores de rede sobrecarregados não são os únicos a caírem nessa armadilha. Diversos produtos foram empacotados com essas regras inseguras. Mesmo a Microsoft é culpada. Os filtros IPsec que vieram com o Windows 2000 e com o Windows XP contém uma regra implícita que permite todo o tráfego TCP ou UDP da porta 88 (Kerberos). Em outro caso bastante conhecido, versões do firewall pessoal Zone Alarm, até a versão 2.1.25, permitiam qualquer pacote UDP entrante com a porta de origem 53 (DNS) ou 67 (DHCP).

O Nmap oferece as opções **−g** e **−−source−port** (elas são equivalentes) para explorar essas fraquezas. Apenas forneça um número de porta e o Nmap irá enviar pacotes dessa porta onde for possível. O

Nmap utiliza números de porta diferentes para que certos testes de detecção de SO funcionem direito e as requisições DNS ignoram a flag **−−source−port** porque o Nmap confia nas bibliotecas de sistema para lidar com isso. A maioria dos rastreios(scans) TCP, incluindo o rastreio(scan) SYN, suportam a opção completamente assim como o rastreio(scan) UDP.

−−data−length <número> (Acrescenta dados aleatórios nos pacotes enviados)
Normalmente o Nmap envia pacotes minimalistas contendo apenas o cabeçalho. Dessa forma os pacotes TCP têm normalmente 40 bytes e os echo requests ICMP tem só 28. Esta opção faz com que o Nmap acrescente o número informado de bytes aleatórios na maioria dos pacotes que envia. Os pacotes de detecção de SO (**−O**) não são afectados mas a maioria dos pacotes de ping e rastreio(scan) de portas são. Isso atrasa as coisas mas pode tornar um rastreio(scan) ligeiramente menos chamativo.

−−ttl <valor> (Establece o valor do campo time−to−live)
Estabelece que o campo tempo−de−vida (time−to−live) dos pacotes enviados terá o valor informado.

−−randomize−hosts (Torna aleatória a ordem dos anfitriões(hosts)−alvo)
Informa ao Nmap que ele deve embaralhar cada grupo de, no máximo, 8096 anfitriões(hosts) antes de efectuar o rastreio(scan). Isso torna os rastreios(scans) menos óbvios a vários sistemas de monitoramento de rede, especialmente quando combina isso com as opções de temporização lentas. Se deseja que fazer isso em grupos maiores aumente o PING_GROUP_SZ no nmap.h e recompile. Uma solução alternativa é gerar uma lista de endereços IP−alvos com um rastreio(scan) de lista (**−sL −n −oN** *nomedoarquivo*), embaralhar a lista com um script Perl e então fornecer a lista completa para o Nmap com **−iL**.

−−spoof−mac <endereço mac, prefixo, ou nome do fabricante> (Disfarça o endereço MAC)
Solicita ao Nmap que utilize o endereço MAC informado para todos os frames ethernet em estado bruto (raw) que ele enviar. Esta opção implica em **−−send−eth** para assegurar que o Nmap realmente envie pacotes no nível ethernet. O MAC fornecido pode assumir diversos formatos. Se for apenas a string “0” o Nmap irá escolher um MAC completamente aleatório para a sessão. Se a string informada for um número par de dígitos hexa (com os pares opcionalmente separados por dois pontos) o Nmap irá usa−la como o MAC. Se menos do que 12 dígitos hexa forem informados o Nmap preenche o restante dos 6 bytes com valores aleatórios. Se o argumento não for um 0 ou uma string hexa o Nmap irá procurar no nmap−mac−prefixes para encontrar o nome de um fabricante contendo a string informada (não é sensível a maiúsculas ou minúsculas). Se encontrar, o Nmap usa o OUI (prefixo de 3 bytes) do fabricante e preenche os 3 bytes restantes aleatoriamente. Exemplos de argumentos **−−spoof−mac** válidos são Apple, 0, 01:02:03:04:05:06, deadbeefcafe, 0020F2 e Cisco.

SAíDA (OUTPUT)

Qualquer ferramenta de segurança só é útil se a saída que ela gera também o for. Testes e algorítmos complexos são de pouco valor se não forem apresentados de uma forma organizada e compreensível. Dado o número de formas que o Nmap é utilizado pelas pessoas e por outros softwares, nenhum formato irá agradar a todos. Então o Nmap oferece diversos formatos incluindo o modo interativo para humanos lerem diretamente e o XML para fácil interpretação por um software.

Além de oferecer diversos formatos de saída, o Nmap fornece opções para controlar a verbosidade da saída assim como as mensagens de depuração. Os tipos de saída podem ser enviados para a saída padrão (standard output) ou para arquivos, o qual o Nmap pode acrescentar ou então sobrescrever. Arquivos de saída também podem ser utilizados para se retomar rastreios(scans) abortados.

O Nmap torna a saída disponível em cinco formatos diferentes. O default é chamado de saída interativa (interactive output) e é enviada para a saída padrão (stdout). Há também a saída normal (normal output) que é similar à interativa excepto pelo facto de mostrar menos informações e alertas sobre a execução uma vez que se espera que seja feita uma análise somente após o rastreio(scan) completar, ao invés de interativamente.

A saída XML é um dos tipos de saída mais importantes pois permite a conversão para HTML, é facilmente analisada por programas como a interface gráfica do Nmap ou pode ser importada em banco de dados.

Os dois tipos restantes de saída são a simples saída para o grep (grepable output) que inclui a maioria das informações de um anfitrião(host)−alvo em uma única linha e a s4íd4 sCRiPt KiDDi3 (sCRiPt KiDDi3

0utPUt) para usuários que se consideram 1r4d0z (|<−r4d).

Embora a saída interativa seja a default e não tenha associada nenhuma opção de linha de comando, as outras quatro opções de formato utilizam a mesma sintaxe. Elas recebem um argumento que é o nome do arquivo onde os resultados devem ser armazenados. Formatos múltiplos podem ser especificados mas cada formato só pode ser especificado uma vez. Por exemplo, pode querer armazenar a saída normal para seu uso enquanto grava a saída XML do mesmo rastreio(scan) para análise utilizando programas. Você pode fazer isso com as opções **−oX myscan.xml −oN myscan.nmap**. Embora este capítulo use nomes simples como myscan.xml por uma questão de brevidade, nomes mais descritivos normalmente são recomendados. Os nomes escolhidos são uma questão de preferência pessoal, embora eu use nomes longos que incorporam da data do rastreio(scan) e uma palavra ou duas que descrevam o rastreio(scan), colocados em um diretório com o nome da empresa que eu estou rastreando.

Mesmo que essas opções gravem os resultados em arquivos, o Nmap ainda assim mostra a saída interativa na stdout como de costume. Por exemplo, o comando **nmap −oX myscan.xml target** grava em XML no myscan.xml e enche a saída padrão com os mesmos resultados interativos que teria mostrado se a opção **−oX** não tivesse sido especificada. Você pode mudar isso passando um caracter hífen como argumento de um dos tipos de formato. Isso faz com que o Nmap desactive a saída interativa e apenas grave os resultados no formato que especificou para a saída padrão. Dessa forma, o comando **nmap −oX − target** irá enviar apenas a saída XML para a stdout. Erros sérios ainda podem ser mostrados na saída padrão de erros, stderr.

Ao contrário de alguns argumentos do Nmap o espaço em branco entre a flag da opção (como a **−oX**) e o nome do arquivo ou hífen é obrigatório. Se omitir as flags e informar argumentos como **−oG−** ou **−oXscan.xml**, uma característica de compatibilidade retroactiva do Nmap irá causar a criação de arquivos de saída do tipo *normal format* chamados G− e Xscan.xml respectivamente.

O Nmap também oferece opções para controlar a verbosidade do rastreio(scan) e para acrescentar informações nos arquivos de saída ao invés de sobrepor. Todas essas opções estão descritas abaixo.

Formatos de Saída do Nmap

−oN <especificaçãodearquivo> (Saída normal)
Solicita que a saída normal (normal output) seja direcionada para o arquivo informado. Conforme discutido acima, é um pouco diferente da saída interativa (interactive output).

−oX <especificaçãodearquivo> (Saída em XML)
Solicita que a saída em XML (XML output) seja direcionada para o arquivo informado. O Nmap inclui uma definição do tipo de documento (document type definition, DTD) que permite que os analisadores (parsers) XML validem a saida em XML do Nmap. Embora seja primeiramente voltada para ser usada por programas também pode ajudar os humanos a interpretar a saída em XML do Nmap. A DTD define os elementos válidos do formato e geralmente enumera os atributos e valores que eles podem receber. A última versão está sempre disponível em **http://www.insecure.org/nmap/data/nmap.dtd**.

O XML oferece um formato estável que é facilmente interpretado por software. Interpretadores (parsers) XML gratuitos estão disponível para as principais linguagens de computador, incluindo C/C++, Perl, Python e Java. As pessoas até já escreveram extensões para a maioria dessas linguagens para manipular a saída e a execução especificamente do Nmap. Exemplos são o **Nmap::Scanner**[6] e o **Nmap::Parser**[7] em Perl CPAN. Em quase todos os casos em que uma aplicação não−trivial faz interface com o Nmap, o XML é o formato preferido.

A saída XML faz referência a uma folha de estilo que pode ser usada para formatar os resultados em HTML. A forma mais fácil de se utilizar isso é simplesmente carregar a saída XML em um navegador web como o Firefox ou o IE. Por default, isso só irá funcionar na máquina onde rodou o Nmap (ou em uma máquina similarmente configurada) devido ao caminho (path) do sistema de arquivos (filesystem) gravado de forma inalterável do nmap.xsl. Veja a opção **−−stylesheet** para ver uma forma de criar um arquivo XML portável que possa ser interpretado como um HTML em qualquer máquina conectada à web.

−oS <especificaçãodearquivo> (S4íd4 ScRipT KIdd|3)

A saída script kiddie é como a saída interativa, com a diferença de ser pós−processada para atender melhor ao "hacker de elite" ('l33t HaXXorZ') que antigamente rejeitava o Nmap devido ao uso consistente de maiúsculas e minúsculas e a grafia correta. Pessoas sem senso de humor devem observar que esta opção serve para se fazer graça dos script kiddies antes de me lixar por estar, supostamente, “ajudando−os”.

−oG <especificaçãodearquivo> (Saída para o grep)
Este formato de saída é mencionado por último porque está deprecado. O formato de saída XML é muito mais poderoso e é bastante adequado para usuário avançados. O XML é um padrão para o qual existem dezenas de excelentes interpretadores (parsers) disponíveis, enquanto que a saída para o grep é um quebra−galho feito por mim. O XML é estensível para suportar novas características do Nmap conforme elas forem lançadas, por outro lado, sempre tenho que omitir essas novas características da saída para o grep por falta de onde colocá−las.

Apesar disso a saída para o grep é bastante popular. É um formato simples que lista cada anfitrião(host) em uma linha e pode ser pesquisado de forma trivial e interpretado por qualquer ferramenta padrão do Unix como o grep, awk, cut, sed, diff e Perl. Eu mesmo uso−a para testes rápidos feitos na linha de comando. Descobrir todos os anfitriões(hosts) com a porta ssh aberta ou que estão com o SO Solaris requer apenas um simples grep para identificá−los, concatenado via pipe a um comando awk ou cut para mostrar os campos desejados.

A saída para o grep consiste de comentários (linhas começadas com o símbolo #) e linhas−alvo. Uma linha−alvo inclui uma combinação de 16 campos rotulados, separados por tab e seguidos por dois−pontos. Os campos são Host, Portas (Ports),Protocolos (Protocols), Estado Ignorado (Ignored State), SO (OS), Índice de Seqüência (Seq Index), IPID e Estado (Status).

O campo mais importante é normalmente Portas (Ports), que fornece detalhes de cada porta interessante. É uma lista com a relação de portas separada por vírgula. Cada porta representa uma porta interessante e tem o formato de sete sub−campos separados por barra (/). Esses sub−campos são: Número da Porta (Port number), Estado (State), Protocolo (Protocol), Proprietário (Owner), Serviço (Service), informação sobre o SunRPC (SunRPC info) e informação sobre a Versão (Version info).

Assim como na saída XML, esta página man não permite que se documente o formato todo. Uma visão mais detalhada sobre o formato de saída para o grep do Nmap está disponível em **http://www.unspecific.com/nmap-oG-output**.

−oA <nome−base> (Saída para todos os formatos)
Para facilitar pode especificar **−oA** *nome−base* para armazenar os resultados de rastreio(scan) nos formatos normal, XML e para o grep de uma só vez. Eles são armazenados nos arquivos *nome−base*.nmap, *nome−base*.xml e *nome−base*.gnmap, respectivamente. Como na maioria dos programas pode colocar como prefixo nos nomes de arquivos o caminho de um diretório como ˜/nmaplogs/foocorp/ no UNIX ou c:\hacking\sco no Windows.

Opções de Verbosidade e depuração (debugging)

−v (Aumenta o nível de verbosidade)
Aumenta o nível de verbosidade fazendo com que o Nmap mostre mais informações sobre o progresso do rastreio(scan). Portas abertas são mostradas conforme são encontradas e estimativas de tempo para o término são fornecidas quando o Nmap acha que um rastreio(scan) irá demorar mais do que alguns minutos. Use duas vêzes para uma verbosidade ainda maior. Usar mais do que duas vêzes não surte nenhum efeito.

A maioria das alterações afectam apenas a saída interactiva e algumas também afectam a saída normal e script kiddie. Os outros tipos de saída foram feitos para serem processados por máquinas, então o Nmap pode dar informações bastante detalhadas por default nesse formatos sem cansarem o usuário humano. Entretanto, existem algumas mudanças nos outros modos onde o tamanho da saída pode ser reduzido substancialmente pela omissão de alguns detalhes. Por exemplo, uma linha de comentário na

saída para o grep que fornece uma lista de todas as portas rastreadas só é mostrada no modo verboso porque ela pode ser bem longa.

−d [nível] (Aumenta ou estabelece o nível de depuração)
Se mesmo o modo verboso não fornece dados suficientes para si, o modo de depuração está disponível para inundá−lo com muito mais! Assim como na opção de verbosidade (**−v**), a depuração é habilitada com uma flag na linha de comando (**−d**) e o nível de depuração pode ser aumentado especificando−a múltiplas vêzes. Alternativamente pode estabelecer o nível de depuração fornecendo um argumento para o **−d**. Por exemplo, **−d9** estabelece o nível nove. Esse é efectivamente o nível mais alto e irá produzir milhares de linhas a menos que execute um rastreio(scan) muito simples com poucas portas e alvos.

A saída da depuração é útil quando há a suspeita de um bug no Nmap ou se está simplesmente confuso com o que o Nmap está fazendo e porquê. Como esta opção é na maioria das vêzes destinada a programadores, as linhas de depuração nem sempre são auto−explicativas. Pode obter algo como: Timeout vals: srtt: −1 rttvar: −1 to: 1000000 delta 14987 ==> srtt: 14987 rttvar: 14987 to: 100000. Se não entender alguma linha suas únicas opções serão ignorá−la, procurar no código−fonte ou pedir ajuda na lista de discussão de desenvolvimento (nmap−dev). Algumas linhas são auto−explicativas mas as mensagens ficam cada vez mais obscuras conforme o nível de depuração é aumentado.

−−packet−trace (Rastreia pacotes e dados enviados e recebidos)
Faz com que o Nmap moste um sumário de todos os pacotes enviados ou recebidos. Isto é bastante usado para depuração mas também é uma forma valiosa para novos usuário entenderem exatamente o que o Nmap está fazendo por debaixo dos panos. Para evitar mostrar milhares de linhas, pode querer especificar um número limitado de portas a rastrear(scan) como **−p20−30**. Se tudo o que lhe interessa for saber o que se passa no subsistema de detecção de versão, use o **−−version−trace**.

−−iflist (Lista as interfaces e rotas)
Mostra a lista de interfaces e rotas do sistema conforme detectados pelo Nmap. Isto é útil para depurar problemas de roteamento ou erro de configuração de dispositivo (como, por exemplo, no caso do Nmap tratar uma conexão PPP como se fosse uma Ethernet).

Opções diversas (miscellaneous) de saída

−−append−output (Acrescenta no arquivo de saída ao invés de sobrepor)
Quando especifica um nome de arquivo na flag de formato de saída, como **−oX** ou **−oN**, esse arquivo é sobreposto por default. Se preferir manter o conteúdo existente no arquivo e acrescentar os novos resultados, especifique a opção **−−append−output**. Todos os arquivos de saída especificados na execução do Nmap terão os resultados acrescidos ao invés de sobrepostos. Isso não funciona bem com os dados de rastreio(scan) para XML (**−oX**) pois o arquivo resultante não será adequadamente interpretado até que consserte manualmente.

−−resume <nomedoarquivo> (Retoma um rastreio(scan) abortado)
Algumas execuções extensas do Nmap podem levar muito tempo, na ordem de dias. Tais rastreios(scans) nem sempre rodam até o fim. Podem haver restrições que impeçam que o Nmap seja executado durante o horário de expediente, a rede pode cair, a máquina onde o Nmap está a ser executado pode sofrer um reboot planeado ou não, ou o Nmap pode simplesmente travar. O administrador que está executando o Nmap poderia cancelá−lo por qualquer outra razão bastando teclar ctrl−C. Reiniciar um rastreio(scan) inteiro do começo pode ser indesejável. Felizmente se forem mantidos logs normais (**−oN**) ou para o grep (**−oG**), o usuário pode pedir que o Nmap continue o rastreio(scan) do alvo que estava verificando quando a execução foi interrompida. Simplesmente especifique a opção **−−resume** e informe o arquivo da saída normal/para o grep como argumento. Nenhum outro argumento é permitido, pois o Nmap analisa o arquivo de saída e usa os mesmos argumentos especificados anteriormente. Basta chamar o Nmap com **nmap −−resume** *nomedoarquivodelog*. O Nmap irá acrescentar os novos resultados ao arquivo de dados especificado na execução anterior. Essa retomada de execução não suporta o formato de saída XML porque combinar as duas execuções em um arquivo XML válido seria difícil.

−−stylesheet <caminho ou URL> (Informa a folha de estilo XSL usada para transformar a saída XML)

O Nmap vem com uma folha de estilo (stylesheet) chamada nmap.xsl para visualizar ou traduzir a saída XML em HTML. A saída XML inclui uma diretiva xml−stylesheet que mostra para o nmap.xml onde ele foi inicialmente instalado pelo Nmap (ou para o diretório corrente no Windows). Simplesmente carregue a saída XML do Nmap em um navegador moderno e ele deve conseguir achar o nmap.xsl no sistema de arquivos e utilizá−lo para interpretar os resultados. Se desejar utilizar uma folha de estilo diferente, especifique−a como um argumento para **−−stylesheet**. Deve informar o caminho completo ou a URL. Uma chamada comum é **−−stylesheet http://www.insecure.org/nmap/data/nmap.xsl** . Isso diz ao navegador para carregar a versão mais actual da folha de estilo da Insecure.Org. Isso torna mais fácil ver os resultados em uma máquina que não tenha o Nmap instalado (e consequentemente o nmap.xsl). A URL é normalmente mais útil mas a localização nmap.xsl num sistema de ficheiros(filesystem) local é usada por default por questões de privacidade.

−−no_stylesheet (Omite do XML a declaração da folha de estilo XSL)
Especifique esta opção para evitar que o Nmap associe qualquer folha de estilo XSL à saída XML. A directiva xml−stylesheet é omitida.

OPçõES DIVERSAS (MISCELLANEOUS)

Esta seção descreve algumas opções importantes (e não−tão−importantes) que realmente não couberam em nenhum outro lugar.

−6 (Habilita o rastreio(scan) IPv6)
Desde 2002 que o Nmap oferece suporte a IPv6 na maioria de suas opções mais populares. Em particular o rastreio(scan) com ping (apenas TCP), o rastreio(scan) com connect() e a detecção de versão, todas suportam IPv6. A sintaxe de comando é a mesma de sempre excepto que irá também adicionar a opção **−6**. É claro que deve usar a sintaxe IPv6 se especificar um endereço no lugar de um nome de anfitrião(host). Um endereço pode se parecer com 3ffe:7501:4819:2000:210:f3ff:fe03:14d0, portanto os nomes de anfitrião(host) são recomendados. A saída é a mesma de sempre com o endereço IPv6 na linha “portas interessantes” sendo a única dica visível de que se tratar realmente de IPv6.

Muito embora o IPv6 não tenha exactamente se alastrado pelo mundo, seu uso se torna mais significativo em alguns países (normalmente asiáticos) e a maioria dos sistemas operativos modernos passaram a suportá−lo. Para usar o Nmap com o IPv6, tanto a origem quanto o alvo de seu rastreio(scan), devem estar configurados para IPv6. Se o seu provedor (ISP) (como a maioria) não aloca endereços IPv6 para si, alguns intermediários que fazem o túnel gratuitamente estão amplamente disponíveis e funcionam bem com o Nmap. Um dos melhores é disponibilizado pela BT Exact. Também tenho utilizado um fornecido pela Hurricane Electric em **http://ipv6tb.he.net/**. Túneis 6para4 são outra abordagem gratuita e popular.

−A (Opções agressivas de rastreio(scan))
Esta opção habilita opções adicionais avançadas e agressivas. Ainda não decidi exactamente qual das duas é a certa. Actualmente ela habilita a Detecção de SO (**−O**) e o rastreio(scan) de versão (**−sV**). Mais características poderão ser adicionadas no futuro. A questão é habilitar um conjunto completo de opções de rastreio(scan) sem que as pessoas tenham que se lembrar de um grupo grande de flags. Esta opção apenas habilita as funções e não as opções de temporização (como a **−T4**) ou opções de verbosidade (**−v**) que pode também querer.

−−datadir <nomedodiretório> (Especifica a localização dos arquivos de dados do rastreio(scan))
O Nmap obtém alguns dados especiais em tempo de execução em arquivos chamados nmap−service−probes, nmap−services, nmap−protocols, nmap−rpc, nmap−mac−prefixes e nmap−os−fingerprints. O Nmap primeiramente procura esses arquivos num diretório especificado na opção **−−datadir** (se houver). Qualquer arquivo que não seja encontrado lá é procurado no diretório especificado pela variável de ambiente NMAPDIR. A seguir vem o ˜/.nmap para se achar os UIDs reais e efectivos (apenas em sistemas POSIX) ou a localização do executável do Nmap (apenas Win32) e então, a localização definida na compilação que pode ser /usr/local/share/nmap ou /usr/share/nmap . Como último recurso, o Nmap irá procurar no diretório corrente.

−−send−eth (Usa a transmissão pela ethernet em estado bruto(raw))

Solicita ao Nmap para que envie pacotes na ethernet (data link) em estado bruto (raw) ao invés de usar a camada de nível mais alto IP (rede). Por default, o Nmap escolhe o que for melhor para a plataforma onde está rodando. Soquetes (sockets) em estado bruto (camada IP) são normalmente mais eficientes em máquinas UNIX enquanto que os frames ethernet são necessários nas operações do Windows, uma vez que a Microsoft desabilitou o suporte a sockets em estado bruto. O Nmap ainda usa pacotes IP em estado bruto no UNIX, independentemente desta opção, quando não há outra alternativa (como no caso de conexões não−ethernet).

−−**send−ip** (Envia no nível do IP em estado bruto(raw))
Pede ao Nmap que envie os pacotes pelos sockets IP em estado bruto(raw) ao invés de enviar pelo nível mais baixo dos frames ethernet. É o complemento da opção −−**send−eth** discutida anteriormente.

−−**privileged** (Assume que o usuário é altamente privilegiado)
Informa ao Nmap para simplesmente assumir que ele tem privilégios suficientes para executar transmissões de sockets em estado bruto(raw), farejar (sniff) pacotes e operações similares que normalmente requerem privilégio de root em sistemas UNIX. Por default, o Nmap encerra se tal operação é solicitada mas o geteuid() não é zero −−**privileged**. É útil com as possibilidades oferecidas pelo kernel do Linux e sistemas similares que pode ser configurado para permitir que usuários não−privilegiados executem rastreios(scans) de pacotes em estado bruto. Assegure−se de informar esta flag de opção antes de outras flags de opção que requeiram privilégios (rastreio(scan) SYN, detecção de OS, etc.). A variável NMAP_PRIVILEGED pode ser configurada como uma alternativa equivalente de −−**privileged**.

−−**interactive** (Inicia em modo interactivo)
Inicia o Nmap em modo interactivo oferecendo um prompt interactivo do Nmap, permitindo o início de múltiplos rastreios(scans) (tanto síncronos quanto em background). Isto é útil para pessoas que fzem o rastreio(scan) a partir de sistemas multi−usuários e que normalmente querem testar a segurança sem deixar todos os utilizadores saberem exactamente quais sistemas eles estão rastreando. Use −−**interactive** para ativar este modo e então tecle h para uma ajuda (help). Esta opção é raramente utilizada porque um shell adequado é mais familiar e tem mais opções. Esta opção inclui um operador exclamação (!) para a execução de comandos de shell, o que é uma das muitas razões de não se instalar o Nmap com setuid root.

−**V**; −−**version** (Mostra o número da versão)
Mostra o número da versão do Nmap e sai.

−**h**; −−**help** (Mostra a página do sumário de ajuda)
Mostra uma pequena tela com as flags de comandos mais comuns. Executar o nmap sem nenhum argumento faz a mesma coisa.

INTERAçãO EM TEMPO DE EXECUçãO

Durante a execução do Nmap todas as teclas pressionadas são capturadas. Isso permite que interaja com o programa sem abortá−lo ou reiniciá−lo. Algumas teclas especiais irão mudar as opções enquanto outras irão mostrar uma mensagem de estado dando informações sobre o rastreio(scan). A convenção é que *letras minúsculas aumentam* a quantidade de informação e *letras maiúsculas diminuem*.

v / **V**
Aumenta / Diminui a quantidade de informações (Verbosity)

d / **D**
Aumenta / Diminui o Nível de Depuração (Debugging Level)

p / **P**
Habilita / Desabilita o Rastreamento de Pacotes (Packet Tracing)

Qualquer outra letra
Mostra uma mensagem de estado como esta:

Stats: 0:00:08 elapsed; 111 anfitriões(hosts) completed (5 up), 5 undergoing Service Scan

Service rastreio(scan) Timing: About 28.00% done; ETC: 16:18 (0:00:15 remaining)

EXEMPLOS

Aqui estão alguns exemplos de utilização do Nmap desde o simples e rotineiro, até ao mais complexo e esotérico. Alguns endereços IP reais e nomes de domínio foram utilizados para tornar as coisas mais concretas. Nesses lugares deve substituir os endereços/nomes pelos da *sua própria rede*. Embora eu não ache que o rastreio(scan) de portas de outras redes seja ou deva ser considerado ilegal, alguns administradores de rede não apreciam o rastreio(scan) não−solicitado de suas redes e podem reclamar. Obter a permissão antecipadamente é a melhor opção.

Para fins de teste tem permissão para rastrear(scan) o anfitrião(host) scanme.nmap.org. Esta permissão inclui apenas o rastreio(scan) via Nmap e não tentativas de explorar vulnerabilidades ou ataques de negação de serviço (denial of service). Para preservar a banda, por favor não inicie mais do que uma dúzia de rastreios(scans) contra o anfitrião(host) por dia. Se esse serviço de alvo livre para rastreio(scan) for abusado, será derrubado e o Nmap irá reportar Failed to resolve given hostname/IP: scanme.nmap.org. Essas permissões também se aplicam aos anfitriões(hosts) scanme2.nmap.org, scanme3.nmap.org e assim por diante, embora esses anfitriões(hosts) ainda não existam.

nmap −v scanme.nmap.org

Esta opção rastreia(scan) todas as portas TCP reservadas na máquina scanme.nmap.org. A opção **−v** habilita o modo verboso (verbose).

nmap −sS −O scanme.nmap.org/24

Inicia um rastreio(scan) SYN camuflado contra cada máquina que estiver activa das 255 possíveis da rede “classe C” onde o Scanme reside. Ele também tenta determinar qual o sistema operativo que está sendo executado em cada anfitrião(host) activo. Isto requer privilégio de root por causa do rastreio(scan) SYN e da detecção de SO.

nmap −sV −p 22,53,110,143,4564 198.116.0−255.1−127

Inicia uma enumeração de anfitriões(hosts) e um rastreio(scan) TCP na primeira metade de cada uma das 255 sub−redes de 8 bits possíveis na classe B do espaço de endereçamento 198.116. Também testa se os sistemas estão executando sshd, DNS, pop3d, imapd ou a porta 4564. Para cada uma destas portas encontradas abertas a detecção de versão é usada para determinar qual aplicação está em execução.

nmap −v −iR 100000 −P0 −p 80

Pede ao Nmap para escolher 100.000 anfitriões(hosts) de forma aleatória e rastreia−os procurando por servidores web (porta 80). A enumeração de anfitriões(hosts) é desabilitada com **−P0** uma vez que enviar primeiramente um par de sondagens para determinar se um anfitriões(hosts) está activo é um desperdício quando se está sondando uma porta em cada anfitrião(host) alvo.

nmap −P0 −p80 −oX logs/pb−port80scan.xml −oG logs/pb−port80scan.gnmap 216.163.128.20/20

Este exemplo rastreia(scan) 4096 endereços IP buscando por servidores web (sem usar o ping) e grava a saída nos formatos XML e compatível com o programa grep.

anfitrião(host) −l company.com | cut −d −f 4 | nmap −v −iL −

Faz uma transferência de zona DNS para descobrir os anfitriões(hosts) em company.com e então alimenta o Nmap com os endereços IP. Os comandos acima são para a minha máquina GNU/Linux −− outros sistemas têm comandos diferentes para executar a transferência de zona.

BUGS

Como o seu autor, o Nmap não é perfeito. Mas pode ajudar a torná−lo melhor enviando relatórios de erros (bug reports) ou mesmo escrevendo correções. Se o Nmap não se comporta da forma que espera, primeiro actualize para a versão mais atual disponível em **http://www.insecure.org/nmap/**. Se o problema persistir, pesquise um pouco para determinar se o problema já foi descoberto e encaminhado. Tente procurar no Google pela mensagem de erro ou navegar nos arquivos da Nmap−dev em **http://seclists.org/**. Se não encontrar nada envie uma mensagem com um relatório do erro para <dev@nmap.org>. Por favor inclua tudo o que souber sobre o problema bem como a versão do Nmap que está executando e em qual versão e

sistema operativo que está a usar.

Correções codificadas para concertar os erros são ainda melhores que os relatórios de erro. Instruções básicas para a criação de arquivos de correções com as suas alterações estão disponíveis em **http://www.insecure.org/nmap/data/HACKING**.

AUTOR

Fyodor <fyodor@nmap.org> (**http://www.insecure.org**)

Centenas de pessoas fizeram contribuições valiosas para o Nmap ao longo dos anos. Isso está detalhado no arquivo CHANGELOG que é distribuído com o Nmap e também está disponível em **http://www.insecure.org/nmap/changelog.html**.

TRADUçãO

Português (Portugal) : José Domingos <jd_pt@yahoo.com> Português (Portugal) : Andreia Gaita <shana.ufie@gmail.com>

Translation Disclaimer: The translation attempts to achieve the highest possible accuracy. The official text is the English version available at [url]. Any discrepancies or differences created in translations are not binding and have no legal effect or compliance or enforcement purposes. If any questions arise in regard to the accuracy of information contained in any translated portion of text, please refer to the official English version. Slangs and language structures in English are not easily translated into another language. Source text that includes jargon common to an industry, may not be translated accurately. Insecure.Com LLC is not responsible for translation errors. We apologize for any translation that is not correct.

Desobrigação da Tradução: A tradução tenta alcançar a maior precisão possível. O texto oficial é a versão em inglês disponível em [url]. Quaisquer discrepâncias ou diferenças criadas pelas traduções não são obrigações e não tem efeito legal, conformidade ou propósitos impositivos. Se qualquer dúvida surgir em relação à precisão da informação contida em qualquer parte traduzida do texto, por favor verifique a versão oficial em inglês. Gírias e estruturas de linguagem em inglês não são facilmente traduzidas em outra língua. Texto original que inclui jargão comum a uma atividade pode não ser traduzido com precisão. A Insecure.Com LLC não é responsável por erros de tradução. Nós pedimos desculpas se alguma parte da tradução não estiver correcta.

AVISOS LEGAIS

Copyright e Licenciamento

O Nmap Security Scanner é (C) 1996−2005 Insecure.Com LLC. O Nmap também é uma marca registada de Insecure.Com LLC. Este programa é um software livre; pode redistribuí−lo e/ou modificá−lo sob os termos da Licença Pública Geral GNU (GNU General Public License) conforme publicado pela Free Software Foundation; Versão 2. Isso garante o seu direito de usar, modificar e redistribuir este software sob certas condições. Se desejar embutir a tecnologia do Nmap em um software proprietário poderemos querer vender licenças alternativas (contate <sales@insecure.com>). Muitos vendedores de scanner de segurança já licenciam a tecnologia do Nmap, tal como a descoberta de anfitriões(hosts), rastreio(scan) de portas, detecção de SO e detecção de serviços/versões.

Observe que a GPL impõe restrições importantes em “trabalhos derivados” embora ela não forneça uma definição detalhada desse termo. Para evitar más−interpretações consideramos que uma aplicação constitui um “trabalho derivado” para o propósito desta licença, se ela se enquadra em um dos seguintes ítens:

- Contém código fonte do Nmap
- Lê ou inclui arquivos de dados do Nmap que são protegidos por copyright, tal como o nmap−os−fingerprints ou nmap−service−probes.
- Executa o Nmap e decompõe (parse) os resultados (diferentemente de uma execução típica de um shell ou aplicações de menu de execução que simplesmente mostram a saída em estado bruto do Nmap e portanto não constituem um trabalho derivado).
- Integra/inclui/agrega o Nmap em um instalador executável proprietário tal como os produzidos pelo InstallShield.

- Estabelece uma ligação (link) com uma biblioteca ou executa um programa que faz qualquer um dos dois ítens em cima.

O termo “Nmap” deve ser considerado como contendo parte ou sendo trabalho derivado do Nmap. Esta lista não é definitiva mas deve ser entendida como uma forma de esclareçer nossa interpretação de trabalho derivado com alguns exemplos comuns. Estas restrições apenas se aplicam quando realmente redistribui o Nmap. Por exemplo, nada impede que escreva e venda um front−end proprietário para o Nmap. Apenas redistribua o seu produto isoladamente e mostre às pessoas onde elas podem descarregar(download) o Nmap.

Nós não consideramos isso como restrições adicionais à GPL mas apenas uma elucidação de como nós interpretamos “trabalhos derivados” pois elas se aplicam ao nosso produto Nmap licenciado no formato GPL. Isso é idêntico à forma como Linus Torvalds anunciou sua interpretação de como os “trabalhos derivados” se aplicam aos módulos do kernel do Linux. A nossa interpretação refere−se apenas ao Nmap − não respondemos por qualquer outro produto GPL.

Se tiver qualquer dúvida quanto às restrições do licenciamento GPL na utilização do Nmap em produtos não−GPL, ficaríamos felizes em ajudar. Como mencionado acima, também oferecemos licenças alternativas para a integração do Nmap em aplicações e dispositivos proprietários. Esses contratos foram vendidos para muitas empresas de segurança e geralmente incluem uma licença perpétua, disponibiliza um suporte para actualizações prioritários, e também nos ajuda financeiramente o desenvolvimento contínuo da tecnologia do Nmap. Por favor, envie um e−mail para <sales@insecure.com> se desejar mais informações.

Como uma exceção especial aos termos da GPL, a Insecure.Com LLC permite que uma ligação (link) do código deste program seja feito com qualquer versão da biblioteca do OpenSSL que seja distribuída sob uma licença idêntica àquela listada no arquivo Copying.OpenSSL incluido e distribuir combinações de ligação incluindo os dois. Deve obedecer à GPL GNU em todos os aspectos para todo o código utilizado que não seja OpenSSL. Se modificar este aquivo pode estender esta excepção para a sua versão do arquivo mas não é obrigado a fazer isso.

Se recebeu estes arquivos com um acordo de licenciamento por escrito ou um contrato ditando termos que não sejam diferentes dos em cima então essa licença alternativa tem precedência sobre estes comentários.

Disponibilidade de código fonte e contribuições da comunidade

O código fonte é fornecido com este software porque acreditamos que os usuários tem o direito de saber exactamente o que um programa irá fazer antes de executá−lo. Isso também permite que vaudite o software procurando por falhas na segurança (nenhuma foi encontrada até agora).

O código fonte também permite que porte o Nmap para novas plataformas consserte problemas e adicione novas características. E altamente encorajado a enviar suas alterações para <fyodor@nmap.org> para uma possível incorporação na distribuição principal. Enviar essas alterações para Fyodor ou para alguém da lista de mensagens de desenvolvimento da Insecure.Org, pressupõe que está oferecendo a Fyodor e à Insecure.Com LLC o direito ilimitado e não−exclusivo para reutilizar, modificar e relicenciar o código. O Nmap sempre estará disponível como um Open Source mas isto é importante porque a impossibilidade de relicenciar o código causou problemas devastadores para outros projetos de Software Livre (tal como o KDE e o NASM). Nós também relicenciamos ocasionalmente o código para terceiros conforme discutido anteriormente. Se deseja especificar condições de licenciamento especiais das suas contribuições deixe isso claro quando enviá−las.

Nenhuma Garantia

Este programa é distribuído na esperança de que será útil mas SEM QUALQUER GARANTIA; sem sequer a garantia implícita de COMERCIALIZAÇÃO ou ADEQUAÇÃO A QUALQUER PROPÓSITO PARTICULAR. Veja a Licença Pública Geral GNU para mais detalhes em **http://www.gnu.org/copyleft/gpl.html** ou no arquivo COPYING incluído com o Nmap.

Também deve ser observado que o Nmap reconhecidamente trava certas aplicações mal−escritas, a pilha TCP/IP e mesmo alguns sistemas operativos. **O Nmap nunca deve ser executado contra sistemas de missão−crítica** a menos que esteja preparado para lidar com o serviço fora do ar (downtime). Nós reconhecemos aqui que o Nmap pode travar os seus sistemas ou redes e nós renunciamos toda e qualquer responsabilidade por qualquer dano ou problema que o Nmap possa causar.

Uso inapropriado

Pelo facto de haver o menor risco de travamento e porque existem pessoas mal−intencionadas (black hats) que gostam de usar o Nmap para reconhecimento antes atacar um sistema, existem administradores que ficam chateados e podem reclamar quando o sistema deles é rastreado. Portanto é normalmente aconselhável que solicite a permissão antes de fazer um rastreio(scan) de uma rede por mais leve que seja.

O Nmap nunca deveria ser instalado com privilégios especiais (p.ex.: suid root) por questões de segurança.

Software de Terceiros

Este produto inclui software desenvolvido pela **Apache Software Foundation**[8]. Uma versão modificada da **biblioteca portátil de captura de pacotes Libpcap**[9] é distribuída junto com o Nmap. A versão para o Windows do Nmap por outro lado utiliza **biblioteca WinPcap**[10], derivada da libpcap. O suporte a expressões regulares é fornecido pela **biblioteca PCRE**[11] que é um software de código aberto escrito por by Philip Hazel. Algumas funções de rede em estado bruto utilizam a biblioteca de rede **Libdnet**[12] que foi escrita por Dug Song. Uma versão modificada é distribuída com o Nmap. O Nmap pode opcionalmente ser ligado ao **conjunto de ferramentas de criptografia do OpenSSL**[13] para o suporte à detecção de versão do SSL. Todos os softwares de terceiros descritos neste parágrafo são distribuídos gratuitamente sob o licenciamento de software no estilo BSD.

Classificação do Controle de Exportação dos EUA

Controle de Exportação dos EUA: A Insecure.Com LLC acredita que o Nmap se enquadra no US ECCN (número de classificação para controle de exportação) 5D992. Essa categoria é chamada de “software de Segurança da Informação não−controlado pela 5D002”. A única restrição a essa classificação é o AT (anti−terrorismo) que se aplica a quase todos os produtos e nega a exportação a um punhado de nações não−confiáveis tais como o Irão e a Coreia do Norte. Portanto, exportar o Nmap não requer nenhuma licença ou permissão especial ou qualquer outro tipo de autorização governamental.

NOTES

1. RFC 1122
   http://www.rfc-editor.org/rfc/rfc1122.txt
2. RFC 792
   http://www.rfc-editor.org/rfc/rfc792.txt
3. UDP
   http://www.rfc-editor.org/rfc/rfc768.txt
4. RFC do TCP
   http://www.rfc-editor.org/rfc/rfc793.txt
5. RFC 959
   http://www.rfc-editor.org/rfc/rfc959.txt
6. Nmap::Scanner
   http://sourceforge.net/projects/nmap-scanner/
7. Nmap::Parser
   http://www.nmapparser.com
8. Apache Software Foundation
   http://www.apache.org
9. biblioteca portátil de captura de pacotes Libpcap
   http://www.tcpdump.org
10. biblioteca WinPcap
    http://www.winpcap.org
11. biblioteca PCRE
    http://www.pcre.org
12. Libdnet
    http://libdnet.sourceforge.net

13. conjunto de ferramentas de criptografia do OpenSSL
http://www.openssl.org

NAME

ps − report a snapshot of the current processes.

SYNOPSIS

ps [*options*]

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use *top*(1) instead.

This version of **ps** accepts several kinds of options:

1 UNIX options, which may be grouped and must be preceded by a dash.
2 BSD options, which may be grouped and must not be used with a dash.
3 GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and **ps** implementations that this **ps** is compatible with.

Note that "**ps −aux**" is distinct from "**ps aux**". The POSIX and UNIX standards require that "**ps −aux**" print all processes owned by a user named "x", as well as printing all processes that would be selected by the **−a** option. If the user named "x" does not exist, this **ps** may interpret the command as "**ps aux**" instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.

By default, **ps** selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD−]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

The use of BSD−style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the **PS_FORMAT** environment variable. The use of BSD−style options will also change the process selection to include processes on other terminals (TTYs) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so **−M** will be considered identical to **Z** and so on.

Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.

EXAMPLES

To see every process on the system using standard syntax:

ps −e
ps −ef
ps −eF
ps −ely

To see every process on the system using BSD syntax:

ps ax
ps axu

To print a process tree:

ps -ejH
ps axjf

To get info about threads:

ps -eLf
ps axms

To get security info:
ps -eo euser,ruser,suser,fuser,f,comm,label
ps axZ
ps -eM

To see every process running as root (real & effective ID) in user format:
ps −U root −u root u

To see every process with a user−defined format:
ps −eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
ps −Ao pid,tt,user,fname,tmout,f,wchan

Print only the process IDs of syslogd:
ps −C syslogd −o pid=

Print only the name of PID 42:
ps −q 42 −o comm=

SIMPLE PROCESS SELECTION

a Lift the BSD−style "only yourself" restriction, which is imposed upon the set of all processes when some BSD−style (without "−") options are used or when the **ps** personality setting is BSD−like. The set of processes selected in this manner is in addition to the set of processes selected by other means. An alternate description is that this option causes **ps** to list all processes with a terminal (tty), or to list all processes when used together with the **x** option.

−A Select all processes. Identical to **−e**.

−a Select all processes except both session leaders (see *getsid*(2)) and processes not associated with a terminal.

−d Select all processes except session leaders.

−−deselect
Select all processes except those that fulfill the specified conditions (negates the selection). Identical to **−N**.

−e Select all processes. Identical to **−A**.

g Really all, even session leaders. This flag is obsolete and may be discontinued in a future release. It is normally implied by the **a** flag, and is only useful when operating in the sunos4 personality.

−N Select all processes except those that fulfill the specified conditions (negates the selection). Identical to **−−deselect**.

T Select all processes associated with this terminal. Identical to the **t** option without any argument.

r Restrict the selection to only running processes.

x Lift the BSD−style "must have a tty" restriction, which is imposed upon the set of all processes when some BSD−style (without "−") options are used or when the **ps** personality setting is BSD−like. The set of processes selected in this manner is in addition to the set of processes selected by other means. An alternate description is that this option causes **ps** to list all processes owned by you (same EUID as **ps**), or to list all processes when used together with the **a** option.

PROCESS SELECTION BY LIST

These options accept a single argument in the form of a blank−separated or comma−separated list. They can be used multiple times. For example: **ps −p "1 2" −p 3,4**

−123 Identical to **−−pid** *123*.

123 Identical to **−−pid** *123*.

−C *cmdlist*
Select by command name. This selects the processes whose executable name is given in *cmdlist*.

−G *grplist*
Select by real group ID (RGID) or name. This selects the processes whose real group name or ID is in the *grplist* list. The real group ID identifies the group of the user who created the process, see *getgid*(2).

−g *grplist*
Select by session OR by effective group name. Selection by session is specified by many standards, but selection by effective group is the logical behavior that several other operating systems use. This **ps** will select by session when the list is completely numeric (as sessions are). Group ID numbers will work only when some group names are also specified. See the **−s** and **−−group** options.

−−Group *grplist*
Select by real group ID (RGID) or name. Identical to **−G**.

−−group *grplist*
Select by effective group ID (EGID) or name. This selects the processes whose effective group name or ID is in *grplist*. The effective group ID describes the group whose file access permissions are used by the process (see *getegid*(2)). The **−g** option is often an alternative to **−−group**.

p *pidlist*
Select by process ID. Identical to **−p** and **−−pid**.

−p *pidlist*
Select by PID. This selects the processes whose process ID numbers appear in *pidlist*. Identical to **p** and **−−pid**.

−−pid *pidlist*
Select by process ID. Identical to **−p** and **p**.

−−ppid *pidlist*
Select by parent process ID. This selects the processes with a parent process ID in *pidlist*. That is, it selects processes that are children of those listed in *pidlist*.

q *pidlist*
Select by process ID (quick mode). Identical to **−q** and **−−quick−pid**.

−q *pidlist*
Select by PID (quick mode). This selects the processes whose process ID numbers appear in *pidlist*. With this option **ps** reads the necessary info only for the pids listed in the *pidlist* and doesn't apply additional filtering rules. The order of pids is unsorted and preserved. No additional selection options, sorting and forest type listings are allowed in this mode. Identical to **q** and **−−quick−pid**.

−−quick−pid *pidlist*
Select by process ID (quick mode). Identical to **−q** and **q**.

−s *sesslist*
Select by session ID. This selects the processes with a session ID specified in *sesslist*.

−−sid *sesslist*
Select by session ID. Identical to **−s**.

t *ttylist* Select by tty. Nearly identical to **−t** and **−−tty**, but can also be used with an empty *ttylist* to indicate the terminal associated with **ps**. Using the **T** option is considered cleaner than using **t** with an empty *ttylist*.

−t *ttylist*
Select by tty. This selects the processes associated with the terminals given in *ttylist*. Terminals (ttys, or screens for text output) can be specified in several forms: /dev/ttyS1, ttyS1, S1. A plain "−" may be used to select processes not attached to any terminal.

−−tty *ttylist*
Select by terminal. Identical to **−t** and **t**.

U *userlist*
Select by effective user ID (EUID) or name. This selects the processes whose effective user name or ID is in *userlist*. The effective user ID describes the user whose file access permissions are used by the process (see *geteuid*(2)). Identical to **−u** and **−−user**.

−U *userlist*
Select by real user ID (RUID) or name. It selects the processes whose real user name or ID is in the *userlist* list. The real user ID identifies the user who created the process, see *getuid*(2).

−u *userlist*
Select by effective user ID (EUID) or name. This selects the processes whose effective user name or ID is in *userlist*.

The effective user ID describes the user whose file access permissions are used by the process (see *geteuid*(2)). Identical to **U** and **−−user**.

−−User *userlist*
Select by real user ID (RUID) or name. Identical to **−U**.

−−user *userlist*
Select by effective user ID (EUID) or name. Identical to **−u** and **U**.

OUTPUT FORMAT CONTROL

These options are used to choose the information displayed by **ps**. The output may differ by personality.

−c Show different scheduler information for the **−l** option.

−−context
Display security context format (for SELinux).

−f Do full−format listing. This option can be combined with many other UNIX−style options to add additional columns. It also causes the command arguments to be printed. When used with **−L**, the NLWP (number of threads) and LWP (thread ID) columns will be added. See the **c** option, the format keyword **args**, and the format keyword **comm**.

−F Extra full format. See the **−f** option, which **−F** implies.

−−format *format*
user−defined format. Identical to **−o** and **o**.

j BSD job control format.

−j Jobs format.

l Display BSD long format.

−l Long format. The **−y** option is often useful with this.

−M Add a column of security data. Identical to **Z** (for SELinux).

O *format*
is preloaded **o** (overloaded). The BSD **O** option can act like **−O** (user−defined output format with some common fields predefined) or can be used to specify sort order. Heuristics are used to determine the behavior of this option. To ensure that the desired behavior is obtained (sorting or formatting), specify the option in some other way (e.g. with **−O** or **−−sort**). When used as a formatting option, it is identical to **−O**, with the BSD personality.

−O *format*
Like **−o**, but preloaded with some default columns. Identical to **−o pid,***format***,state,tname,time, command** or **−o pid,***format***,tname,time,cmd**, see **−o** below.

o *format*
Specify user−defined format. Identical to **−o** and **−−format**.

−o *format*
User−defined format. *format* is a single argument in the form of a blank−separated or comma−separated list, which offers a way to specify individual output columns. The recognized keywords are described in the **STANDARD FORMAT SPECIFIERS** section below. Headers may be renamed (**ps −o pid,ruser=RealUser −o comm=Command**) as desired. If all column headers are empty (**ps −o pid= −o comm=**) then the header line will not be output. Column width will increase as needed for wide headers; this may be used to widen up columns such as WCHAN (**ps −o pid,wchan=WIDE−WCHAN−COLUMN −o comm**). Explicit width control (**ps opid, wchan:42,cmd**) is offered too. The behavior of **ps -o pid=X,comm=Y** varies with personality; output may be one column named "X,comm=Y" or two columns named "X" and "Y". Use multiple **−o** options when in doubt. Use the **PS_FORMAT** environment variable to specify a default as desired; DefSysV and DefBSD are macros that may be used to choose the default UNIX or BSD columns.

s Display signal format.

u Display user−oriented format.

v Display virtual memory format.

X Register format.

−y Do not show flags; show rss in place of addr. This option can only be used with **−l**.

Z Add a column of security data. Identical to **−M** (for SELinux).

OUTPUT MODIFIERS

c Show the true command name. This is derived from the name of the executable file, rather than from the argv value. Command arguments and any modifications to them are thus not shown. This option effectively turns the **args** format keyword into the **comm** format keyword; it is useful with the **−f** format option and with the various BSD−style format options, which all normally display the command arguments. See the **−f** option, the format keyword **args**, and the format keyword **comm**.

−−cols *n*
Set screen width.

−−columns *n*
Set screen width.

−−cumulative
Include some dead child process data (as a sum with the parent).

e Show the environment after the command.

f ASCII art process hierarchy (forest).

−−forest
ASCII art process tree.

h No header. (or, one header per screen in the BSD personality). The **h** option is problematic. Standard BSD **ps** uses this option to print a header on each page of output, but older Linux **ps** uses this option to totally disable the header. This version of **ps** follows the Linux usage of not printing the header unless the BSD personality has been selected, in which case it prints a header on each page of output. Regardless of the current personality, you can use the long options **−−headers** and **−−no−headers** to enable printing headers each page or disable headers entirely, respectively.

−H Show process hierarchy (forest).

−−headers
Repeat header lines, one per page of output.

k *spec* Specify sorting order. Sorting syntax is [+|−]*key*[,[+|−]*key*[,...]]. Choose a multi−letter key from the **STANDARD FORMAT SPECIFIERS** section. The "+" is optional since default direction is increasing numerical or lexicographic order. Identical to **−−sort**.

Examples:
ps jaxkuid,−ppid,+pid
ps axk comm o comm,args
ps kstart_time −ef

−−lines *n*
Set screen height.

n Numeric output for WCHAN and USER (including all types of UID and GID).

−−no−headers
Print no header line at all. **−−no−heading** is an alias for this option.

O *order*
Sorting order (overloaded). The BSD **O** option can act like **−O** (user−defined output format with some common fields predefined) or can be used to specify sort order. Heuristics are used to determine the behavior of this option. To ensure that the desired behavior is obtained (sorting or formatting), specify the option in some other way (e.g. with **−O** or **−−sort**).

For sorting, obsolete BSD **O** option syntax is **O**[+|−]*k1*[,[+|−]*k2*[,...]]. It orders the processes listing according to the multilevel sort specified by the sequence of one−letter short keys *k1*,*k2*, ... described in the **OBSOLETE SORT KEYS** section below. The "+" is currently optional, merely re−iterating the default direction on a key, but may help to distinguish an **O** sort from an **O** format. The "−" reverses direction only on the key it precedes.

−−rows *n*
Set screen height.

S Sum up some information, such as CPU usage, from dead child processes into their parent. This is useful for examining a system where a parent process repeatedly forks off short−lived children to do work.

−−sort *spec*
Specify sorting order. Sorting syntax is [+|−]*key*[,[+|−]*key*[,...]]. Choose a multi−letter key from the **STANDARD FORMAT SPECIFIERS** section. The "+" is optional since default direction is increasing numerical or lexicographic order. Identical to **k**. For example: **ps jax −−sort=uid,−ppid,+pid**

w Wide output. Use this option twice for unlimited width.

−w Wide output. Use this option twice for unlimited width.

−−width *n*
Set screen width.

THREAD DISPLAY

H Show threads as if they were processes.

−L Show threads, possibly with LWP and NLWP columns.

m Show threads after processes.

−m Show threads after processes.

−T Show threads, possibly with SPID column.

OTHER INFORMATION

−−help *section*
Print a help message. The section argument can be one of *s*imple, *l*ist, *o*utput, *t*hreads, *m*isc or *a*ll. The argument can be shortened to one of the underlined letters as in: s|l|o|t|m|a.

−−info Print debugging info.

L List all format specifiers.

V Print the procps-ng version.

−V Print the procps-ng version.

−−version
Print the procps-ng version.

NOTES

This **ps** works by reading the virtual files in /proc. This **ps** does not need to be setuid kmem or have any privileges to run. Do not give this **ps** any special permissions.

CPU usage is currently expressed as the percentage of time spent running during the entire lifetime of a process. This is not ideal, and it does not conform to the standards that **ps** otherwise conforms to. CPU usage is unlikely to add up to exactly 100%.

The SIZE and RSS fields don't count some parts of a process including the page tables, kernel stack, struct thread_info, and struct task_struct. This is usually at least 20 KiB of memory that is always resident. SIZE is the virtual size of the process (code+data+stack).

Processes marked <defunct> are dead processes (so−called "zombies") that remain because their parent has not destroyed them properly. These processes will be destroyed by *init*(8) if the parent process exits.

If the length of the username is greater than the length of the display column, the username will be truncated. See the -o and -O formatting options to customize length.

Commands options such as **ps −aux** are not recommended as it is a confusion of two different standards. According to the POSIX and UNIX standards, the above command asks to display all processes with a TTY (generally the commands users are running) plus all processes owned by a user named "x". If that user doesn't exist, then **ps** will assume you really meant "**ps** *aux*".

PROCESS FLAGS

The sum of these values is displayed in the "F" column, which is provided by the **flags** output specifier:

1 forked but didn't exec
4 used super−user privileges

PROCESS STATE CODES

Here are the different values that the **s**, **stat** and **state** output specifiers (header "STAT" or "S") will display to describe the state of a process:

D uninterruptible sleep (usually IO)
R running or runnable (on run queue)
S interruptible sleep (waiting for an event to complete)
T stopped by job control signal
t stopped by debugger during the tracing
W paging (not valid since the 2.6.xx kernel)
X dead (should never be seen)
Z defunct ("zombie") process, terminated but not reaped by its parent

For BSD formats and when the **stat** keyword is used, additional characters may be displayed:

< high−priority (not nice to other users)
N low−priority (nice to other users)
L has pages locked into memory (for real−time and custom IO)
s is a session leader
l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+ is in the foreground process group

OBSOLETE SORT KEYS

These keys are used by the BSD **O** option (when it is used for sorting). The GNU **−−sort** option doesn't use these keys, but the specifiers described below in the **STANDARD FORMAT SPECIFIERS** section.

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 (e.g. sorting on tty will sort into device number, not according to the terminal name displayed). Pipe **ps** output into the **sort**(1) command if you want to sort the cooked values.

KEY	LONG	DESCRIPTION
c	cmd	simple name of executable
C	pcpu	cpu utilization
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
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
P	ppid	parent process ID
r	rss	resident set size
R	resident	resident pages
s	size	memory size in kilobytes
S	share	amount of shared pages
t	tty	the device number of the controlling tty
T	start_time	time process was started
U	uid	user ID number
u	user	user name
v	vsize	total VM size in KiB
y	priority	kernel scheduling priority

AIX FORMAT DESCRIPTORS

This **ps** supports AIX format descriptors, which work somewhat like the formatting codes of *printf*(1) and *printf*(3). For example, the normal default output can be produced with this: **ps −eo "%p %y %x %c"**. The **NORMAL** codes are described in the next section.

CODE	NORMAL	HEADER
%C	pcpu	%CPU
%G	group	GROUP
%P	ppid	PPID
%U	user	USER
%a	args	COMMAND
%c	comm	COMMAND
%g	rgroup	RGROUP
%n	nice	NI
%p	pid	PID
%r	pgid	PGID
%t	etime	ELAPSED
%u	ruser	RUSER
%x	time	TIME
%y	tty	TTY
%z	vsz	VSZ

STANDARD FORMAT SPECIFIERS

Here are the different keywords that may be used to control the output format (e.g. with option **−o**) or to sort the selected processes with the GNU−style **−−sort** option.

For example: **ps −eo pid,user,args −−sort user**

This version of **ps** tries to recognize most of the keywords used in other implementations of **ps**.

The following user−defined format specifiers may contain spaces:
args, **cmd**, **comm**, **command**, **fname**, **ucmd**, **ucomm**, **lstart**, **bsdstart**, **start**.

Some keywords may not be available for sorting.

CODE	HEADER	DESCRIPTION
%cpu	%CPU	cpu utilization of the process in "##.#" format. Currently, it is the CPU time used divided by the time the process has been running (cputime/realtime ratio), expressed as a percentage. It will not add up to 100% unless you are lucky. (alias **pcpu**).
%mem	%MEM	ratio of the process's resident set size to the physical memory on the machine, expressed as a percentage. (alias **pmem**).
args	COMMAND	command with all its arguments as a string. Modifications to the arguments may be shown. The output in this column may contain spaces. A process marked <defunct> is partly dead, waiting to be fully destroyed by its parent. Sometimes the process args will be unavailable; when this happens, **ps** will instead print the executable name in brackets. (alias **cmd**, **command**). See also the **comm** format keyword, the **−f** option, and the **c** option. When specified last, this column will extend to the edge of the display. If **ps** can not determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined (it may be 80, unlimited, determined by the **TERM** variable, and so on). The **COLUMNS** environment variable or **−−cols** option may be used to exactly determine the width in this case. The **w** or **−w** option may be also be used to adjust width.
blocked	BLOCKED	mask of the blocked signals, see *signal*(7). According to the width of the field, a 32 or 64−bit mask in hexadecimal format is displayed. (alias **sig_block**, **sigmask**).
bsdstart	START	time the command started. If the process was started less than 24 hours ago, the output format is " HH:MM", else it is " Mmm:SS" (where Mmm is the three letters of the month). See also **lstart**, **start**, **start_time**, and **stime**.
bsdtime	TIME	accumulated cpu time, user + system. The display format is usually "MMM:SS", but can be shifted to the right if the process used more than 999 minutes of cpu time.
c	C	processor utilization. Currently, this is the integer value of the percent usage over the lifetime of the process. (see **%cpu**).
caught	CAUGHT	mask of the caught signals, see *signal*(7). According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias **sig_catch**, **sigcatch**).
cgname	CGNAME	display name of control groups to which the process belongs.
cgroup	CGROUP	display control groups to which the process belongs.

class CLS scheduling class of the process. (alias **policy**, **cls**). Field's possible values are:

−	not reported
TS	SCHED_OTHER
FF	SCHED_FIFO
RR	SCHED_RR
B	SCHED_BATCH
ISO	SCHED_ISO
IDL	SCHED_IDLE
?	unknown value

cls CLS scheduling class of the process. (alias **policy**, **cls**). Field's possible values are:

−	not reported
TS	SCHED_OTHER
FF	SCHED_FIFO
RR	SCHED_RR
B	SCHED_BATCH
ISO	SCHED_ISO
IDL	SCHED_IDLE
?	unknown value

cmd CMD see **args**. (alias **args**, **command**).

comm COMMAND command name (only the executable name). Modifications to the command name will not be shown. A process marked <defunct> is partly dead, waiting to be fully destroyed by its parent. The output in this column may contain spaces. (alias **ucmd**, **ucomm**). See also the **args format keyword,** the **−f** option, and the **c** option.
When specified last, this column will extend to the edge of the display. If **ps** can not determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined (it may be 80, unlimited, determined by the **TERM** variable, and so on). The **COLUMNS** environment variable or **−−cols** option may be used to exactly determine the width in this case. The **w** or **−w** option may be also be used to adjust width.

command COMMAND See **args**. (alias **args**, **command**).

cp CP per−mill (tenths of a percent) CPU usage. (see **%cpu**).

cputime TIME cumulative CPU time, "[DD−]hh:mm:ss" format. (alias **time**).

drs DRS data resident set size, the amount of physical memory devoted to other than executable code.

egid EGID effective group ID number of the process as a decimal integer. (alias **gid**).

egroup EGROUP effective group ID of the process. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias **group**).

eip EIP instruction pointer.

esp ESP stack pointer.

etime	ELAPSED	elapsed time since the process was started, in the form [[DD−]hh:]mm:ss.
etimes	ELAPSED	elapsed time since the process was started, in seconds.
euid	EUID	effective user ID (alias **uid**).
euser	EUSER	effective user name. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise. The **n** option can be used to force the decimal representation. (alias **uname**, user**).**
f	F	flags associated with the process, see the **PROCESS FLAGS** section. (alias **flag**, **flags**).
fgid	FGID	filesystem access group ID. (alias **fsgid**).
fgroup	FGROUP	filesystem access group ID. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias **fsgroup**).
flag	F	see **f**. (alias **f**, **flags**).
flags	F	see **f**. (alias **f**, **flag**).
fname	COMMAND	first 8 bytes of the base name of the process's executable file. The output in this column may contain spaces.
fuid	FUID	filesystem access user ID. (alias **fsuid**).
fuser	FUSER	filesystem access user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.
gid	GID	see **egid**. (alias **egid**).
group	GROUP	see **egroup**. (alias **egroup**).
ignored	IGNORED	mask of the ignored signals, see *signal*(7). According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias **sig_ignore**, **sigignore**).
ipcns	IPCNS	Unique inode number describing the namespace the process belongs to. See namespaces(7).
label	LABEL	security label, most commonly used for SELinux context data. This is for the *Mandatory Access Control* ("MAC") found on high−security systems.
lstart	STARTED	time the command started. See also **bsdstart**, **start**, **start_time**, and **stime**.
lsession	SESSION	displays the login session identifier of a process, if systemd support has been included.
lwp	LWP	light weight process (thread) ID of the dispatchable entity (alias **spid**, **tid**). See **tid** for additional information.

lxc	LXC	The name of the lxc container within which a task is running. If a process is not running inside a container, a dash ('−') will be shown.
machine	MACHINE	displays the machine name for processes assigned to VM or container, if systemd support has been included.
maj_flt	MAJFLT	The number of major page faults that have occurred with this process.
min_flt	MINFLT	The number of minor page faults that have occurred with this process.
mntns	MNTNS	Unique inode number describing the namespace the process belongs to. See namespaces(7).
netns	NETNS	Unique inode number describing the namespace the process belongs to. See namespaces(7).
ni	NI	nice value. This ranges from 19 (nicest) to −20 (not nice to others), see *nice*(1). (alias **nice**).
nice	NI	see **ni**.**(alias ni**).
nlwp	NLWP	number of lwps (threads) in the process. (alias **thcount**).
nwchan	WCHAN	address of the kernel function where the process is sleeping (use **wchan** if you want the kernel function name). Running tasks will display a dash ('−') in this column.
ouid	OWNER	displays the Unix user identifier of the owner of the session of a process, if systemd support has been included.
pcpu	%CPU	see **%cpu**. (alias **%cpu**).
pending	PENDING	mask of the pending signals. See *signal*(7). Signals pending on the process are distinct from signals pending on individual threads. Use the **m** option or the **−m** option to see both. According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias **sig**).
pgid	PGID	process group ID or, equivalently, the process ID of the process group leader. (alias **pgrp**).
pgrp	PGRP	see **pgid**. (alias **pgid**).
pid	PID	a number representing the process ID (alias **tgid**).
pidns	PIDNS	Unique inode number describing the namespace the process belongs to. See namespaces(7).
pmem	%MEM	see **%mem**. (alias **%mem**).

policy POL scheduling class of the process. (alias **class**, **cls**). Possible values are:

−	not reported
TS	SCHED_OTHER
FF	SCHED_FIFO
RR	SCHED_RR
B	SCHED_BATCH
ISO	SCHED_ISO
IDL	SCHED_IDLE
?	unknown value

ppid PPID parent process ID.

pri PRI priority of the process. Higher number means lower priority.

psr PSR processor that process is currently assigned to.

rgid RGID real group ID.

rgroup RGROUP real group name. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

rss RSS resident set size, the non−swapped physical memory that a task has used (in kiloBytes). (alias **rssize**, **rsz**).

rssize RSS see **rss**. (alias **rss**, **rsz**).

rsz RSZ see **rss**. (alias **rss**, **rssize**).

rtprio RTPRIO realtime priority.

ruid RUID real user ID.

ruser RUSER real user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

s S minimal state display (one character). See section **PROCESS STATE CODES** for the different values. See also **stat** if you want additional information displayed. (alias **state**).

sched SCH scheduling policy of the process. The policies SCHED_OTHER (SCHED_NORMAL), SCHED_FIFO, SCHED_RR, SCHED_BATCH, SCHED_ISO, and SCHED_IDLE are respectively displayed as 0, 1, 2, 3, 4, and 5.

seat SEAT displays the identifier associated with all hardware devices assigned to a specific workplace, if systemd support has been included.

sess SESS session ID or, equivalently, the process ID of the session leader. (alias **session**, **sid**).

sgi_p P processor that the process is currently executing on. Displays "*" if the process is not currently running or runnable.

sgid	SGID	saved group ID. (alias **svgid**).
sgroup	SGROUP	saved group name. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.
sid	SID	see **sess**. (alias **sess**, **session**).
sig	PENDING	see **pending**. (alias **pending**, **sig_pend**).
sigcatch	CAUGHT	see **caught**. (alias **caught**, **sig_catch**).
sigignore	IGNORED	see **ignored**. (alias **ignored**, **sig_ignore**).
sigmask	BLOCKED	see **blocked**. (alias **blocked**, **sig_block**).
size	SIZE	approximate amount of swap space that would be required if the process were to dirty all writable pages and then be swapped out. This number is very rough!
slice	SLICE	displays the slice unit which a process belongs to, if systemd support has been included.
spid	SPID	see **lwp**. (alias **lwp**, **tid**).
stackp	STACKP	address of the bottom (start) of stack for the process.
start	STARTED	time the command started. If the process was started less than 24 hours ago, the output format is "HH:MM:SS", else it is " Mmm dd" (where Mmm is a three−letter month name). See also **lstart**, **bsdstart**, **start_time**, and **stime**.
start_time	START	starting time or date of the process. Only the year will be displayed if the process was not started the same year **ps** was invoked, or "MmmDD" if it was not started the same day, or "HH:MM" otherwise. See also **bsdstart**, **start**, **lstart**, and **stime**.
stat	STAT	multi−character process state. See section **PROCESS STATE CODES** for the different values meaning. See also **s** and **state** if you just want the first character displayed.
state	S	see **s**. (alias **s**).
suid	SUID	saved user ID. (alias **svuid**).
supgid	SUPGID	group ids of supplementary groups, if any. See **getgroups**(2).
supgrp	SUPGRP	group names of supplementary groups, if any. See **getgroups**(2).
suser	SUSER	saved user name. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias **svuser**).
svgid	SVGID	see **sgid**. (alias **sgid**).
svuid	SVUID	see **suid**. (alias **suid**).

sz	SZ	size in physical pages of the core image of the process. This includes text, data, and stack space. Device mappings are currently excluded; this is subject to change. See **vsz** and **rss**.
tgid	TGID	a number representing the thread group to which a task belongs (alias **pid**). It is the process ID of the thread group leader.
thcount	THCNT	see **nlwp**. (alias **nlwp**). number of kernel threads owned by the process.
tid	TID	the unique number representing a dispatchable entity (alias **lwp**, **spid**). This value may also appear as: a process ID (pid); a process group ID (pgrp); a session ID for the session leader (sid); a thread group ID for the thread group leader (tgid); and a tty process group ID for the process group leader (tpgid).
time	TIME	cumulative CPU time, "[DD−]HH:MM:SS" format. (alias **cputime**).
tname	TTY	controlling tty (terminal). (alias **tt**, **tty**).
tpgid	TPGID	ID of the foreground process group on the tty (terminal) that the process is connected to, or −1 if the process is not connected to a tty.
trs	TRS	text resident set size, the amount of physical memory devoted to executable code.
tt	TT	controlling tty (terminal). (alias **tname**, **tty**).
tty	TT	controlling tty (terminal). (alias **tname**, **tt**).
ucmd	CMD	see **comm**. (alias **comm**, **ucomm**).
ucomm	COMMAND	see **comm**. (alias **comm**, **ucmd**).
uid	UID	see **euid**. (alias **euid**).
uname	USER	see **euser**. (alias **euser**, **user**).
unit	UNIT	displays unit which a process belongs to, if systemd support has been included.
user	USER	see **euser**. (alias **euser**, **uname**).
userns	USERNS	Unique inode number describing the namespace the process belongs to. See namespaces(7).
utsns	UTSNS	Unique inode number describing the namespace the process belongs to. See namespaces(7).
uunit	UUNIT	displays user unit which a process belongs to, if systemd support has been included.
vsize	VSZ	see **vsz**. (alias **vsz**).

vsz	VSZ	virtual memory size of the process in KiB (1024−byte units). Device mappings are currently excluded; this is subject to change. (alias **vsize**).
wchan	WCHAN	name of the kernel function in which the process is sleeping, a "−" if the process is running, or a "*" if the process is multi−threaded and **ps** is not displaying threads.

ENVIRONMENT VARIABLES

The following environment variables could affect **ps**:

COLUMNS
Override default display width.

LINES
Override default display height.

PS_PERSONALITY
Set to one of posix, old, linux, bsd, sun, digital... (see section **PERSONALITY** below).

CMD_ENV
Set to one of posix, old, linux, bsd, sun, digital... (see section **PERSONALITY** below).

I_WANT_A_BROKEN_PS
Force obsolete command line interpretation.

LC_TIME
Date format.

PS_COLORS
Not currently supported.

PS_FORMAT
Default output format override. You may set this to a format string of the type used for the **−o** option. The **DefSysV** and **DefBSD** values are particularly useful.

POSIXLY_CORRECT
Don’t find excuses to ignore bad "features".

POSIX2
When set to "on", acts as **POSIXLY_CORRECT**.

UNIX95
Don’t find excuses to ignore bad "features".

_XPG
Cancel **CMD_ENV**=*irix* non−standard behavior.

In general, it is a bad idea to set these variables. The one exception is **CMD_ENV** or **PS_PERSONALITY**, which could be set to Linux for normal systems. Without that setting, **ps** follows the useless and bad parts of the Unix98 standard.

PERSONALITY

390	like the OS/390 OpenEdition **ps**
aix	like AIX **ps**
bsd	like FreeBSD **ps** (totally non−standard)
compaq	like Digital Unix **ps**
debian	like the old Debian **ps**
digital	like Tru64 (was Digital Unix, was OSF/1) **ps**
gnu	like the old Debian **ps**
hp	like HP−UX **ps**
hpux	like HP−UX **ps**

irix	like Irix **ps**
linux	***** **recommended** *****
old	like the original Linux **ps** (totally non−standard)
os390	like OS/390 Open Edition **ps**
posix	standard
s390	like OS/390 Open Edition **ps**
sco	like SCO **ps**
sgi	like Irix **ps**
solaris2	like Solaris 2+ (SunOS 5) **ps**
sunos4	like SunOS 4 (Solaris 1) **ps** (totally non−standard)
svr4	standard
sysv	standard
tru64	like Tru64 (was Digital Unix, was OSF/1) **ps**
unix	standard
unix95	standard
unix98	standard

SEE ALSO

pgrep(1), **pstree**(1), **top**(1), **proc**(5).

STANDARDS

This **ps** conforms to:

1 Version 2 of the Single Unix Specification
2 The Open Group Technical Standard Base Specifications, Issue 6
3 IEEE Std 1003.1, 2004 Edition
4 X/Open System Interfaces Extension [UP XSI]
5 ISO/IEC 9945:2003

AUTHOR

ps was originally written by Branko Lankester ⟨lankeste@fwi.uva.nl⟩. Michael K. Johnson ⟨johnsonm@redhat.com⟩ re−wrote it significantly to use the proc filesystem, changing a few things in the process. Michael Shields ⟨mjshield@nyx.cs.du.edu⟩ added the pid−list feature. Charles Blake ⟨cblake@bbn.com⟩ added multi−level sorting, the dirent−style library, the device name−to−number mmaped database, the approximate binary search directly on System.map, and many code and documentation cleanups. David Mossberger−Tang wrote the generic BFD support for psupdate. Albert Cahalan ⟨albert@users.sf.net⟩ rewrote ps for full Unix98 and BSD support, along with some ugly hacks for obsolete and foreign syntax.

Please send bug reports to ⟨procps@freelists.org⟩. No subscription is required or suggested.

NOME

rm − apaga arquivos e diretórios

SINOPSE

rm [*opções***]** *arquivo...*

Opções POSIX: **[−fiRr]**

Opções GNU (forma reduzida): **[−dfirvR] [−−help] [−−version] [−−]**

DESCRIÇÃO

rm apaga o *arquivo*. Por padrão, não são apagados diretórios. Mas quando a opção −r ou −R é fornecida, a árvore de diretório abaixo do diretório especificado é removida (e não há limitações de níveis de árvores de diretórios que podem ser apagados por 'rm −r'). É um erro quando o último componente do caminho de busca do *arquivo* é um dos dois . ou .. (para se evitar surpresas desagradáveis com 'rm −r .*' ou 'rm −r ..*').

Se a opção −i é dada, ou se um arquivo não pode ser escrito, a entrada padrão é um terminal, e se a opção **−f** não é fornecida, **rm** pergunta para o usuário se deseja remover o arquivo, escrevendo a questão no erro padrão e lendo a resposta na entrada padrão. Se a resposta for negativa, o arquivo é preservado.

OPÇÕES POSIX

−f Não pergunta por confirmações. Não imprime mensagens de diagnóstico. Não produz mensagens de erro se o erro se deve a arquivos que não existem.

−i Pergunta por confirmação (No caso de ambas −f and −i serem fornecidas, a última delas terá efeito.)

−r or **−R**
Apaga as árvores de diretório de forma recursiva.

DETALHES SVID

A definição da interface System V proíbe a remoção da última ligação para um arquivo binário executável que está sendo executado.

DETALHES GNU

A implementação GNU (no Arquivos de Utilitários 3.16) está interrompida no sentido que há um limite superior à profundidade de hierarquias que podem ser removidas. (Se necessário, um utilitário 'deltree' pode ser usado para remover muitos níveis de árvores.)

OPÇÕES GNU

−d, −−directory
Remove diretórios com **unlink**(2) no lugar de **rmdir**(2), e não exige que um diretório esteja vazio antes de tentar um unlink. Somente funciona se você tem privilégios apropriados. Porque ao desvincular um diretório faz qualquer arquivo no diretório apagado se tornar inacessível, é recomendado passar o **fsck**(8) no sistema de arquivos depois de fazer isto.

−f, −−force
Ignora arquivos não existentes a nunca questiona o usuário.

−i, −−interactive
Questiona se cada arquivo será apagado. Se a reposta for negativa, o arquivo é preservado.

−r, −R, −−recursive
Apaga o conteúdo dos diretórios de forma recursiva.

−v, −−verbose
Imprime o nome de cada arquivo antes de apagá-lo.

OPÇÕES PADRÃO GNU

−−help Imprime a mensagem de uso na saída padrão e sai.

−−version
Imprime a versão na saída padrão e sai.

−− Encerra a lista de opção.

AMBIENTE

As variáveis LANG, LC_ALL, LC_CTYPE and LC_MESSAGES tem seu significado usual.

DE ACORDO COM

POSIX 1003.2, exceto para limitações no nível de profundidade de hierarquia do arquivo.

NOTAS

Esta página descreve **rm** como é encontrado no pacote Utilitários de Arquivo 4.0; outras versões podem ser um pouco diferentes. Envie correções e adições para aeb@cwi.nl. Relatório de problemas no programa para fileutils-bugs@gnu.ai.mit.edu.

TRADUZIDO POR LDP-BR em 21/08/2000.

André L. Fassone Canova <lonelywolf@blv.com.br> (tradução) Ricardo C.O. Freitas <english.quest@best-service.com> (revisão)

NAME

sed − stream editor for filtering and transforming text

SYNOPSIS

sed [*OPTION*]... *{script-only-if-no-other-script}* [*input-file*]...

DESCRIPTION

Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as *ed*), *sed* works by making only one pass over the input(s), and is consequently more efficient. But it is *sed*'s ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

−n, **−−quiet**, **−−silent**

suppress automatic printing of pattern space

−e script, **−−expression**=*script*

add the script to the commands to be executed

−f script-file, **−−file**=*script−file*

add the contents of script-file to the commands to be executed

−−follow−symlinks

follow symlinks when processing in place

−i[SUFFIX], **−−in−place**[=*SUFFIX*]

edit files in place (makes backup if SUFFIX supplied)

−l N, **−−line−length**=*N*

specify the desired line-wrap length for the 'l' command

−−posix

disable all GNU extensions.

−E, **−r**, **−−regexp−extended**

use extended regular expressions in the script (for portability use POSIX **−E**).

−s, **−−separate**

consider files as separate rather than as a single, continuous long stream.

−−sandbox

operate in sandbox mode.

−u, **−−unbuffered**

load minimal amounts of data from the input files and flush the output buffers more often

−z, **−−null−data**

separate lines by NUL characters

−−help

display this help and exit

−−version

output version information and exit

If no **−e**, **−−expression**, **−f**, or **−−file** option is given, then the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read.

GNU sed home page: <http://www.gnu.org/software/sed/>. General help using GNU software: <http://www.gnu.org/gethelp/>. E-mail bug reports to: <bug-sed@gnu.org>.

COMMAND SYNOPSIS

This is just a brief synopsis of *sed* commands to serve as a reminder to those who already know *sed*; other documentation (such as the texinfo document) must be consulted for fuller descriptions.

Zero-address "commands"

: *label* Label for **b** and **t** commands.

#comment
The comment extends until the next newline (or the end of a **-e** script fragment).

} The closing bracket of a { } block.

Zero- or One- address commands

= Print the current line number.

a \

text Append *text*, which has each embedded newline preceded by a backslash.

i \

text Insert *text*, which has each embedded newline preceded by a backslash.

q [*exit-code*]
Immediately quit the *sed* script without processing any more input, except that if auto-print is not disabled the current pattern space will be printed. The exit code argument is a GNU extension.

Q [*exit-code*]
Immediately quit the *sed* script without processing any more input. This is a GNU extension.

r *filename*
Append text read from *filename*.

R *filename*
Append a line read from *filename*. Each invocation of the command reads a line from the file. This is a GNU extension.

Commands which accept address ranges

{ Begin a block of commands (end with a }).

b *label* Branch to *label*; if *label* is omitted, branch to end of script.

c \

text Replace the selected lines with *text*, which has each embedded newline preceded by a backslash.

d Delete pattern space. Start next cycle.

D If pattern space contains no newline, start a normal new cycle as if the d command was issued. Otherwise, delete text in the pattern space up to the first newline, and restart cycle with the resultant pattern space, without reading a new line of input.

h H Copy/append pattern space to hold space.

g G Copy/append hold space to pattern space.

l List out the current line in a ''visually unambiguous'' form.

l *width* List out the current line in a ''visually unambiguous'' form, breaking it at *width* characters. This is a GNU extension.

n N Read/append the next line of input into the pattern space.

p Print the current pattern space.

P Print up to the first embedded newline of the current pattern space.

s/*regexp*/*replacement*/
Attempt to match *regexp* against the pattern space. If successful, replace that portion matched with *replacement*. The *replacement* may contain the special character **&** to refer to that portion of

the pattern space which matched, and the special escapes \1 through \9 to refer to the corresponding matching sub-expressions in the *regexp*.

t *label* If a s/// has done a successful substitution since the last input line was read and since the last t or T command, then branch to *label*; if *label* is omitted, branch to end of script.

T *label* If no s/// has done a successful substitution since the last input line was read and since the last t or T command, then branch to *label*; if *label* is omitted, branch to end of script. This is a GNU extension.

w *filename*
Write the current pattern space to *filename*.

W *filename*
Write the first line of the current pattern space to *filename*. This is a GNU extension.

x Exchange the contents of the hold and pattern spaces.

y/*source*/*dest*/
Transliterate the characters in the pattern space which appear in *source* to the corresponding character in *dest*.

Addresses

Sed commands can be given with no addresses, in which case the command will be executed for all input lines; with one address, in which case the command will only be executed for input lines which match that address; or with two addresses, in which case the command will be executed for all input lines which match the inclusive range of lines starting from the first address and continuing to the second address. Three things to note about address ranges: the syntax is *addr1*,*addr2* (i.e., the addresses are separated by a comma); the line which *addr1* matched will always be accepted, even if *addr2* selects an earlier line; and if *addr2* is a *regexp*, it will not be tested against the line that *addr1* matched.

After the address (or address-range), and before the command, a **!** may be inserted, which specifies that the command shall only be executed if the address (or address-range) does **not** match.

The following address types are supported:

number
Match only the specified line *number* (which increments cumulatively across files, unless the **-s** option is specified on the command line).

first~*step*
Match every *step*'th line starting with line *first*. For example, ‘‘sed -n 1~2p’’ will print all the odd-numbered lines in the input stream, and the address 2~5 will match every fifth line, starting with the second. *first* can be zero; in this case, *sed* operates as if it were equal to *step*. (This is an extension.)

$ Match the last line.

/*regexp*/
Match lines matching the regular expression *regexp*.

c*regexp***c**
Match lines matching the regular expression *regexp*. The **c** may be any character.

GNU *sed* also supports some special 2-address forms:

0,*addr2*
Start out in "matched first address" state, until *addr2* is found. This is similar to 1,*addr2*, except that if *addr2* matches the very first line of input the 0,*addr2* form will be at the end of its range, whereas the 1,*addr2* form will still be at the beginning of its range. This works only when *addr2* is a regular expression.

addr1,+*N*
Will match *addr1* and the *N* lines following *addr1*.

addr1,~*N*

Will match *addr1* and the lines following *addr1* until the next line whose input line number is a multiple of *N*.

REGULAR EXPRESSIONS

POSIX.2 BREs *should* be supported, but they aren't completely because of performance problems. The **\n** sequence in a regular expression matches the newline character, and similarly for **\a**, **\t**, and other sequences. The *-E* option switches to using extended regular expressions instead; the -E option has been supported for years by GNU sed, and is now included in POSIX.

BUGS

E-mail bug reports to **bug-sed@gnu.org**. Also, please include the output of ''sed --version'' in the body of your report if at all possible.

AUTHOR

Written by Jay Fenlason, Tom Lord, Ken Pizzini, and Paolo Bonzini. GNU sed home page: <http://www.gnu.org/software/sed/>. General help using GNU software: <http://www.gnu.org/gethelp/>. E-mail bug reports to: <bug-sed@gnu.org>.

COPYRIGHT

SEE ALSO

awk(1), **ed**(1), **grep**(1), **tr**(1), **perlre**(1), sed.info, any of various books on *sed*, the *sed* FAQ (http://sed.sf.net/grabbag/tutorials/sedfaq.txt), http://sed.sf.net/grabbag/.

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

info sed

should give you access to the complete manual.

NAME

top − display Linux processes

SYNOPSIS

top −**hv**|−**bcHiOSs** −**d** secs −**n** max −**u**|**U** user −**p** pid −**o** fld −**w** [cols]

The traditional switches '−' and whitespace are optional.

DESCRIPTION

The **top** program provides a dynamic real-time view of a running system. It can display **system** summary information a as a list of **processes** or **threads** currently being managed by the Linux kernel. The types of system summary inform shown and the types, order and size of information displayed for processes are all user configurable and that configuratio be made persistent across restarts.

The program provides a limited interactive interface for process manipulation as well as a much more extensive interfa personal configuration −− encompassing every aspect of its operation. And while **top** is referred to throughout this docu you are free to name the program anything you wish. That new name, possibly an alias, will then be reflected on top's d and used when reading and writing a configuration file.

OVERVIEW

Documentation

The remaining Table of Contents

Operation

When operating top, the two most important keys are the help (h or ?) key and quit ('q') key. Alternatively, you could s use the traditional interrupt key (^C) when you're done.

When started for the first time, you'll be presented with these traditional elements on the main top screen: 1) Summary Ar Fields/Columns Header; 3) Task Area. Each of these will be explored in the sections that follow. There is also an Input sage line between the Summary Area and Columns Header which needs no further explanation.

The main top screen is *generally* quite adaptive to changes in terminal dimensions under X-Windows. Other top screen be less so, especially those with static text. It ultimately depends, however, on your particular window manager and ter emulator. There may be occasions when their view of terminal size and current contents differs from top's view, wh always based on operating system calls.

Following any re-size operation, if a top screen is corrupted, appears incomplete or disordered, simply typing som innocuous like a punctuation character or cursor motion key will usually restore it. In extreme cases, the following seq almost certainly will:

key/cmd objective
^Z **suspend** top
fg **resume** top
<Left> force a screen **redraw** (if necessary)

But if the display is still corrupted, there is one more step you could try. Insert this command after top has been suspend before resuming it.

key/cmd objective
reset restore your **terminal settings**

Note: the width of top's display will be limited to 512 positions. Displaying all fields requires approximately 250 chara Remaining screen width is usually allocated to any variable width columns currently visible. The variable width columns as COMMAND, are noted in topic 3a. DESCRIPTIONS of Fields. Actual output width may also be influenced by th switch, which is discussed in topic 1. COMMAND−LINE Options.

Lastly, some of top's screens or functions require the use of cursor motion keys like the standard arrow keys plus the I End, PgUp and PgDn keys. If your terminal or emulator does not provide those keys, the following combinations are ac as alternatives:

key equivalent-key-combinations
Up alt + \ or alt + **k**
Down alt + / or alt + **j**
Left alt + < or alt + **h**
Right alt + > or alt + **l** (lower case L)
PgUp alt + **Up** or alt + ctrl + **k**
PgDn alt + **Down** or alt + ctrl + **j**
Home alt + **Left** or alt + ctrl + **h**
End alt + **Right** or alt + ctrl + **l**

The **Up** and **Down** arrow keys have special significance when prompted for line input terminated with the <Enter> key. keys, or their aliases, can be used to retrieve previous input lines which can then be edited and re-input. And there ar

additional keys available with line oriented input.

key	*special-significance*
Up	recall **older** strings for re-editing
Down	recall **newer** strings or **erase** entire line
Insert	toggle between **insert** and **overtype** modes
Delete	character **removed** at cursor, moving others left
Home	jump to **beginning** of input line
End	jump to **end** of input line

Startup Defaults

The following startup defaults assume no configuration file, thus no user customizations. Even so, items shown with an as
('*') could be overridden through the command-line. All are explained in detail in the sections that follow.

Global-defaults
A − Alt display Off (full-screen)
* d − Delay time 1.5 seconds
* H − Threads mode Off (summarize as tasks)
I − Irix mode On (no, 'solaris' smp)
* p − PID monitoring Off (show all processes)
* s − Secure mode Off (unsecured)
B − Bold enable On (yes, bold globally)
Summary-Area-defaults
l − Load Avg/Uptime On (thus program name)
t − Task/Cpu states On (1+1 lines, see '1')
m − Mem/Swap usage On (2 lines worth)
1 − Single Cpu Off (thus multiple cpus)
Task-Area-defaults
b − Bold hilite Off (use 'reverse')
* c − Command line Off (name, not cmdline)
* i − Idle tasks On (show all tasks)
J − Num align right On (not left justify)
j − Str align right Off (not right justify)
R − Reverse sort On (pids high-to-low)
* S − Cumulative time Off (no, dead children)
* u − User filter Off (show euid only)
* U − User filter Off (show any uid)
V − Forest view On (show as branches)
x − Column hilite Off (no, sort field)
y − Row hilite On (yes, running tasks)
z − color/mono On (show colors)

Linux Memory Types

For our purposes there are three types of memory, and one is optional. First is physical memory, a limited resource where
and data must reside when executed or referenced. Next is the optional swap file, where modified (dirty) memory can be
and later retrieved if too many demands are made on physical memory. Lastly we have virtual memory, a nearly unli
resource serving the following goals:

1. abstraction, free from physical memory addresses/limits
2. isolation, every process in a separate address space
3. sharing, a single mapping can serve multiple needs
4. flexibility, assign a virtual address to a file

Regardless of which of these forms memory may take, all are managed as pages (typically 4096 bytes) but expressed by d
in top as KiB (kibibyte). The memory discussed under topic '2c. MEMORY Usage' deals with physical memory and the

file for the system as a whole. The memory reviewed in topic '3. FIELDS / Columns Display' embraces all three m types, but for individual processes.

For each such process, every memory page is restricted to a single quadrant from the table below. Both physical memo virtual memory can include any of the four, while the swap file only includes #1 through #3. The memory in quadra when modified, acts as its own dedicated swap file.

```
                  Private | Shared
               1          |        2
  Anonymous  . stack              |
           . malloc()             |
           . brk()/sbrk()         | . POSIX shm*
           . mmap(PRIVATE, ANON)  | . mmap(SHARED, ANON)
          ------------------------+----------------------
           . mmap(PRIVATE, fd)    | . mmap(SHARED, fd)
  File-backed  . pgms/shared libs  |
               3          |        4
```

The following may help in interpreting process level memory values displayed as scalable columns and discussed under '3a. DESCRIPTIONS of Fields'.

%MEM − simply RES divided by total physical memory
CODE − the 'pgms' portion of quadrant **3**
DATA − the entire quadrant **1** portion of VIRT plus all
explicit mmap file-backed pages of quadrant **3**
RES − anything occupying physical memory which, beginning with
Linux-4.5, is the sum of the following three fields:
RSan − quadrant **1** pages, which include any
former quadrant **3** pages if modified
RSfd − quadrant **3** and quadrant **4** pages
RSsh − quadrant **2** pages
RSlk − subset of RES which cannot be swapped out (any quadrant)
SHR − subset of RES (excludes **1**, includes all **2** & **4**, some **3**)
SWAP − potentially any quadrant except **4**
USED − simply the sum of RES and SWAP
VIRT − everything in-use and/or reserved (all quadrants)

Note: Even though program images and shared libraries are considered *private* to a process, they will be accounted *shared* (SHR) by the kernel.

1. COMMAND-LINE Options

The command-line syntax for top consists of:

−**hv**|−**bcHiOSs** −**d** secs −**n** max −**u**|**U** user −**p** pid −**o** fld −**w** [cols]

The typically mandatory switch ('−') and even whitespace are completely optional.

−**h** | −**v** :*Help/Version*
Show library version and the usage prompt, then quit.

−**b** :*Batch-mode* operation
Starts top in Batch mode, which could be useful for sending output from top to other programs or to a file. In this top will not accept input and runs until the iterations limit you've set with the '−n' command−line option or until ki

−**c** :*Command-line/Program-name* toggle
Starts top with the last remembered ‘c’ state reversed. Thus, if top was displaying command lines, now that fiel
show program names, and vice versa. See the ‘c’ interactive command for additional information.

−**d** :*Delay-time* interval as: **-d ss.t** (*secs.tenths*)
Specifies the delay between screen updates, and overrides the corresponding value in one’s personal configuration
the startup default. Later this can be changed with the ‘d’ or ‘s’ interactive commands.

Fractional seconds are honored, but a negative number is not allowed. In all cases, however, such changes are proh
if top is running in Secure mode, except for root (unless the ‘s’ command−line option was used). For additional inf
tion on Secure mode see topic 6a. SYSTEM Configuration File.

−**H** :*Threads-mode* operation
Instructs top to display individual threads. Without this command−line option a summation of all threads in each p
is shown. Later this can be changed with the ‘H’ interactive command.

−**i** :*Idle-process* toggle
Starts top with the last remembered ‘i’ state reversed. When this toggle is *Off*, tasks that have not used any CPU sin
last update will not be displayed. For additional information regarding this toggle see topic 4c. TASK AREA
mands, SIZE.

−**n** :*Number-of-iterations* limit as: **−n number**
Specifies the maximum number of iterations, or frames, top should produce before ending.

−**o** :*Override-sort-field* as: **−o fieldname**
Specifies the name of the field on which tasks will be sorted, independent of what is reflected in the configuratio
You can prepend a ‘+’ or ‘−’ to the field name to also override the sort direction. A leading ‘+’ will force sorting h
low, whereas a ‘−’ will ensure a low to high ordering.

This option exists primarily to support automated/scripted batch mode operation.

−**O** :*Output-field-names*
This option acts as a form of help for the above −o option. It will cause top to print each of the available field name
separate line, then quit. Such names are subject to nls translation.

−**p** :*Monitor-PIDs* mode as: **−pN1 -pN2 ...** or **−pN1,N2,N3 ...**
Monitor only processes with specified process IDs. This option can be given up to 20 times, or you can provide a c
delimited list with up to 20 pids. Co-mingling both approaches is permitted.

A pid value of zero will be treated as the process id of the top program itself once it is running.

This is a command−line option only and should you wish to return to normal operation, it is not necessary to qu
restart top −− just issue any of these interactive commands: ‘=’, ‘u’ or ‘U’.

The ‘p’, ‘u’ and ‘U’ command−line options are mutually exclusive.

−**s** :*Secure-mode* operation
Starts top with secure mode forced, even for root. This mode is far better controlled through the system configurati
(see topic 6. FILES).

−**S** :*Cumulative-time* toggle
Starts top with the last remembered 'S' state reversed. When Cumulative time mode is *On*, each process is listed w
cpu time that it and its dead children have used. See the 'S' interactive command for additional information reg
this mode.

−**u** | −**U** :*User-filter-mode* as: **-u** | **-U number** or **name**
Display only processes with a user id or user name matching that given. The '−u' option matches on *effectiv*
whereas the '−U' option matches on *any* user (real, effective, saved, or filesystem).

Prepending an exclamation point ('!') to the user id or name instructs top to display only processes with users not m
ing the one provided.

The 'p', 'u' and 'U' command−line options are mutually exclusive.

−**w** :*Output-width-override* as: **-w** [**number**]
In Batch mode, when used without an argument top will format output using the COLUMNS= and LINES= enviro
variables, if set. Otherwise, width will be fixed at the maximum 512 columns. With an argument, output width c
decreased or increased (up to 512) but the number of rows is considered unlimited.

In normal display mode, when used without an argument top will *attempt* to format output using the COLUMNS
LINES= environment variables, if set. With an argument, output width can only be decreased, not increased. W
using environment variables or an argument with −w, when *not* in Batch mode actual terminal dimensions can ne
exceeded.

Note: Without the use of this command−line option, output width is always based on the terminal at which to
invoked whether or not in Batch mode.

2. SUMMARY Display

Each of the following three areas are individually controlled through one or more interactive commands. See topic 4b.
MARY AREA Commands for additional information regarding these provisions.

2a. UPTIME and LOAD Averages

This portion consists of a single line containing:
program or **window** name, depending on display mode
current time and length of time since last boot
total number of users
system load avg over the last 1, 5 and 15 minutes

2b. TASK and CPU States

This portion consists of a minimum of two lines. In an SMP environment, additional lines can reflect individual CPU sta
centages.

Line 1 shows total **tasks** or **threads**, depending on the state of the Threads-mode toggle. That total is further classified as
running; sleeping; stopped; zombie

Line 2 shows CPU state percentages based on the interval since the last refresh.

As a default, percentages for these individual categories are displayed. Where two labels are shown below, those for
recent kernel versions are shown first.
us, **user** : time running un-niced user processes
sy, **system** : time running kernel processes
ni, **nice** : time running niced user processes

id, **idle** : time spent in the kernel idle handler
wa, **IO-wait** : time waiting for I/O completion
hi : time spent servicing hardware interrupts
si : time spent servicing software interrupts
st : time stolen from this vm by the hypervisor

In the alternate cpu states display modes, beyond the first tasks/threads line, an abbreviated summary is shown consist
these elements:

```
        a   b    c   d
%Cpu(s):  75.0/25.0  100[ ...
```

Where: a) is the combined **us** and **ni** percentage; b) is the **sy** percentage; c) is the total; and d) is one of two visual gra
those representations. See topic 4b. SUMMARY AREA Commands and the 't' command for additional information c
special 4-way toggle.

2c. MEMORY Usage

This portion consists of two lines which may express values in kibibytes (KiB) through exbibytes (EiB) depending on th
ing factor enforced with the 'E' interactive command.

As a default, Line 1 reflects physical memory, classified as:
total, free, used and buff/cache

Line 2 reflects mostly virtual memory, classified as:
total, free, used and avail (which is physical memory)

The **avail** number on line 2 is an estimation of physical memory available for starting new applications, without swa
Unlike the **free** field, it attempts to account for readily reclaimable page cache and memory slabs. It is available on k
3.14, emulated on kernels 2.6.27+, otherwise the same as **free**.

In the alternate memory display modes, two abbreviated summary lines are shown consisting of these elements:

```
        a   b        c
GiB Mem : 18.7/15.738   [ ...
GiB Swap:  0.0/7.999    [ ...
```

Where: a) is the percentage used; b) is the total available; and c) is one of two visual graphs of those representations.

In the case of physical memory, the percentage represents the **total** minus the estimated **avail** noted above. The 'Mem'
itself is divided between **used** and any remaining memory not otherwise accounted for by **avail**. See topic 4b. SUMM
AREA Commands and the 'm' command for additional information on that special 4-way toggle.

This table may help in interpreting the scaled values displayed:

```
KiB = kibibyte = 1024 bytes
MiB = mebibyte = 1024 KiB = 1,048,576 bytes
GiB = gibibyte = 1024 MiB = 1,073,741,824 bytes
TiB = tebibyte = 1024 GiB = 1,099,511,627,776 bytes
PiB = pebibyte = 1024 TiB = 1,125,899,906,842,624 bytes
EiB = exbibyte = 1024 PiB = 1,152,921,504,606,846,976 bytes
```

3. FIELDS / Columns

3a. DESCRIPTIONS of Fields

Listed below are top's available process fields (columns). They are shown in strict ascii alphabetical order. You may cust
their position and whether or not they are displayable with the 'f' or 'F' (Fields Management) interactive commands.

Any field is selectable as the sort field, and you control whether they are sorted high-to-low or low-to-high. For addi
information on sort provisions see topic 4c. TASK AREA Commands, SORTING.

The fields related to physical memory or virtual memory reference '(KiB)' which is the unsuffixed display mode. Such
may, however, be scaled from KiB through PiB. That scaling is influenced via the 'e' interactive command or establish
startup through a build option.

1. **%CPU** −− CPU Usage
   The task's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time.

   In a true SMP environment, if a process is multi-threaded and top is *not* operating in Threads mode, amounts greate
   100% may be reported. You toggle Threads mode with the 'H' interactive command.

   Also for multi-processor environments, if Irix mode is *Off*, top will operate in Solaris mode where a task's cpu usag
   be divided by the total number of CPUs. You toggle Irix/Solaris modes with the 'I' interactive command.

2. **%MEM** −− Memory Usage (RES)
   A task's currently resident share of available physical memory.

   See 'OVERVIEW, Linux Memory Types' for additional details.

3. **CGNAME** −− Control Group Name
   The name of the control group to which a process belongs, or '−' if not applicable for that process.

   This will typically be the last entry in the full list of control groups as shown under the next heading (CGROUPS). A
   is true there, this field is also variable width.

4. **CGROUPS** −− Control Groups
   The names of the control group(s) to which a process belongs, or '−' if not applicable for that process.

   Control Groups provide for allocating resources (cpu, memory, network bandwidth, etc.) among installation-d
   groups of processes. They enable fine-grained control over allocating, denying, prioritizing, managing and moni
   those resources.

   Many different hierarchies of cgroups can exist simultaneously on a system and each hierarchy is attached to one or
   subsystems. A subsystem represents a single resource.

   Note: The CGROUPS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable
   columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable
   fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any
   cated data.

5. **CODE** −− Code Size (KiB)
   The amount of physical memory currently devoted to executable code, also known as the Text Resident Set size or TR

   See 'OVERVIEW, Linux Memory Types' for additional details.

6. **COMMAND** −− Command **Name** or Command **Line**
   Display the command line used to start a task or the name of the associated program. You toggle between comman
   and *name* with 'c', which is both a command−line option and an interactive command.

   When you've chosen to display command lines, processes without a command line (like kernel threads) will be s

with only the program name in brackets, as in this example:
[kthreadd]

This field may also be impacted by the forest view display mode. See the 'V' interactive command for additional inf
tion regarding that mode.

Note: The COMMAND field, unlike most columns, is not fixed-width. When displayed, it plus any other variable
columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable
fields could still suffer truncation. This is especially true for this field when command lines are being displayed (t
interactive command.) See topic 5c. SCROLLING a Window for additional information on accessing any truncated

7. **DATA** −− Data + Stack Size (KiB)
   The amount of private memory *reserved* by a process. It is also known as the Data Resident Set or DRS. Such m
   may not yet be mapped to physical memory (RES) but will always be included in the virtual memory (VIRT) amount

   See 'OVERVIEW, Linux Memory Types' for additional details.

8. **ENVIRON** −− Environment variables
   Display all of the environment variables, if any, as seen by the respective processes. These variables will be displa
   their raw native order, not the sorted order you are accustomed to seeing with an unqualified 'set'.

   Note: The ENVIRON field, unlike most columns, is not fixed-width. When displayed, it plus any other variable widt
   umns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable
   fields could still suffer truncation. This is especially true for this field. See topic 5c. SCROLLING a Window for
   tional information on accessing any truncated data.

9. **Flags** −− Task Flags
   This column represents the task's current scheduling flags which are expressed in hexadecimal notation and with
   suppressed. These flags are officially documented in <linux/sched.h>.

10. **GID** −− Group Id
    The *effective* group ID.

11. **GROUP** −− Group Name
    The *effective* group name.

12. **LXC** −− Lxc Container Name
    The name of the lxc container within which a task is running. If a process is not running inside a container, a dash
    will be shown.

13. **NI** −− Nice Value
    The nice value of the task. A negative nice value means higher priority, whereas a positive nice value means lower
    ity. Zero in this field simply means priority will not be adjusted in determining a task's dispatch-ability.

14. **OOMa** −− Out of Memory Adjustment Factor
    The value, ranging from -1000 to +1000, added to the current out of memory score (OOMs) which is then used to
    mine which task to kill when memory is exhausted.

15. **OOMs** −− Out of Memory Score
    The value, ranging from 0 to +1000, used to select task(s) to kill when memory is exhausted. Zero translates to
    kill' whereas 1000 means 'always kill'.

16. **P** −− Last used CPU (SMP)
    A number representing the last used processor. In a true SMP environment this will likely change frequently since th
    nel intentionally uses weak affinity. Also, the very act of running top may break this weak affinity and cause mor
    cesses to change CPUs more often (because of the extra demand for cpu time).

17. **PGRP** −− Process Group Id
    Every process is member of a unique process group which is used for distribution of signals and by terminals to ar
    requests for their input and output. When a process is created (forked), it becomes a member of the process group
    parent. By convention, this value equals the process ID (see PID) of the first member of a process group, call
    process group leader.

18. **PID** −− Process Id
    The task's unique process ID, which periodically wraps, though never restarting at zero. In kernel terms, it is a dis
    able entity defined by a task_struct.

    This value may also be used as: a process group ID (see PGRP); a session ID for the session leader (see SID); a
    group ID for the thread group leader (see TGID); and a TTY process group ID for the process group leader (see TPG

19. **PPID** −− Parent Process Id
    The process ID (pid) of a task's parent.

20. **PR** −− Priority
    The scheduling priority of the task. If you see 'rt' in this field, it means the task is running under real time schedulin
    ority.

    Under linux, real time priority is somewhat misleading since traditionally the operating itself was not preemptible
    while the 2.6 kernel can be made mostly preemptible, it is not always so.

21. **RES** −− Resident Memory Size (KiB)
    A subset of the virtual address space (VIRT) representing the non-swapped physical memory a task is currently usi
    is also the sum of the RSan, RSfd and RSsh fields.

    It can include private anonymous pages, private pages mapped to files (including program images and shared libr
    plus shared anonymous pages. All such memory is backed by the swap file represented separately under SWAP.

    Lastly, this field may also include shared file-backed pages which, when modified, act as a dedicated swap file an
    will never impact SWAP.

    See 'OVERVIEW, Linux Memory Types' for additional details.

22. **RSan** −− Resident Anonymous Memory Size (KiB)
    A subset of resident memory (RES) representing private pages not mapped to a file.

23. **RSfd** −− Resident File-Backed Memory Size (KiB)
    A subset of resident memory (RES) representing the implicitly shared pages supporting program images and s
    libraries. It also includes explicit file mappings, both private and shared.

24. **RSlk** −− Resident Locked Memory Size (KiB)
    A subset of resident memory (RES) which cannot be swapped out.

25. **RSsh** −− Resident Shared Memory Size (KiB)
A subset of resident memory (RES) representing the explicitly shared anonymous shm*/mmap pages.

26. **RUID** −− Real User Id
The *real* user ID.

27. **RUSER** −− Real User Name
The *real* user name.

28. **S** −− Process Status
The status of the task which can be one of:
D = uninterruptible sleep
R = running
S = sleeping
T = stopped by job control signal
t = stopped by debugger during trace
Z = zombie

Tasks shown as running should be more properly thought of as ready to run −− their task_struct is simply represen
the Linux run-queue. Even without a true SMP machine, you may see numerous tasks in this state depending on
delay interval and nice value.

29. **SHR** −− Shared Memory Size (KiB)
A subset of resident memory (RES) that may be used by other processes. It will include shared anonymous page
shared file-backed pages. It also includes private pages mapped to files representing program images and shared libra

See 'OVERVIEW, Linux Memory Types' for additional details.

30. **SID** −− Session Id
A session is a collection of process groups (see PGRP), usually established by the login shell. A newly forked p
joins the session of its creator. By convention, this value equals the process ID (see PID) of the first member of th
sion, called the session leader, which is usually the login shell.

31. **SUID** −− Saved User Id
The *saved* user ID.

32. **SUPGIDS** −− Supplementary Group IDs
The IDs of any supplementary group(s) established at login or inherited from a task's parent. They are displaye
comma delimited list.

Note: The SUPGIDS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable widt
umns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable
fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any
cated data.

33. **SUPGRPS** −− Supplementary Group Names
The names of any supplementary group(s) established at login or inherited from a task's parent. They are displaye
comma delimited list.

Note: The SUPGRPS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable widt
umns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable

fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any
cated data.

34. **SUSER** −− Saved User Name
The *saved* user name.

35. **SWAP** −− Swapped Size (KiB)
The formerly resident portion of a task's address space written to the swap file when physical memory becomes over
mitted.

See 'OVERVIEW, Linux Memory Types' for additional details.

36. **TGID** −− Thread Group Id
The ID of the thread group to which a task belongs. It is the PID of the thread group leader. In kernel terms, it repr
those tasks that share an mm_struct.

37. **TIME** −− CPU Time
Total CPU time the task has used since it started. When Cumulative mode is *On*, each process is listed with the cp
that it and its dead children have used. You toggle Cumulative mode with 'S', which is both a command−line optio
an interactive command. See the 'S' interactive command for additional information regarding this mode.

38. **TIME+** −− CPU Time, hundredths
The same as TIME, but reflecting more granularity through hundredths of a second.

39. **TPGID** −− Tty Process Group Id
The process group ID of the foreground process for the connected tty, or −1 if a process is not connected to a termina
convention, this value equals the process ID (see PID) of the process group leader (see PGRP).

40. **TTY** −− Controlling Tty
The name of the controlling terminal. This is usually the device (serial port, pty, etc.) from which the process was s
and which it uses for input or output. However, a task need not be associated with a terminal, in which case you'll s
displayed.

41. **UID** −− User Id
The *effective* user ID of the task's owner.

42. **USED** −− Memory in Use (KiB)
This field represents the non-swapped physical memory a task is using (RES) plus the swapped out portion of its a
space (SWAP).

See 'OVERVIEW, Linux Memory Types' for additional details.

43. **USER** −− User Name
The *effective* user name of the task's owner.

44. **VIRT** −− Virtual Memory Size (KiB)
The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages tha
been swapped out and pages that have been mapped but not used.

See 'OVERVIEW, Linux Memory Types' for additional details.

45. **WCHAN** −− Sleeping in Function
    This field will show the name of the kernel function in which the task is currently sleeping. Running tasks will dis
    dash (‘−’) in this column.

46. **nDRT** −− Dirty Pages Count
    The number of pages that have been modified since they were last written to auxiliary storage. Dirty pages must be
    ten to auxiliary storage before the corresponding physical memory location can be used for some other virtual page.

47. **nMaj** −− Major Page Fault Count
    The number of **major** page faults that have occurred for a task. A page fault occurs when a process attempts to read
    or write to a virtual page that is not currently present in its address space. A major page fault is when auxiliary s
    access is involved in making that page available.

48. **nMin** −− Minor Page Fault count
    The number of **minor** page faults that have occurred for a task. A page fault occurs when a process attempts to read
    or write to a virtual page that is not currently present in its address space. A minor page fault does not involve au
    storage access in making that page available.

48. **nTH** −− Number of Threads
    The number of threads associated with a process.

50. **nsIPC** −− IPC namespace
    The Inode of the namespace used to isolate interprocess communication (IPC) resources such as System V IPC o
    and POSIX message queues.

51. **nsMNT** −− MNT namespace
    The Inode of the namespace used to isolate filesystem mount points thus offering different views of the filesystem h
    chy.

52. **nsNET** −− NET namespace
    The Inode of the namespace used to isolate resources such as network devices, IP addresses, IP routing, port numbers

53. **nsPID** −− PID namespace
    The Inode of the namespace used to isolate process ID numbers meaning they need not remain unique. Thus, each
    namespace could have its own ‘init/systemd’ (PID #1) to manage various initialization tasks and reap orphaned chil
    cesses.

54. **nsUSER** −− USER namespace
    The Inode of the namespace used to isolate the user and group ID numbers. Thus, a process could have a normal un
    leged user ID outside a user namespace while having a user ID of 0, with full root privileges, inside that namespace.

55. **nsUTS** −− UTS namespace
    The Inode of the namespace used to isolate hostname and NIS domain name. UTS simply means "UNIX Time-sh
    System".

56. **vMj** −− Major Page Fault Count Delta
    The number of **major** page faults that have occurred since the last update (see nMaj).

57. **vMn** −− Minor Page Fault Count Delta
The number of **minor** page faults that have occurred since the last update (see nMin).

3b. MANAGING Fields

After pressing the interactive command 'f' or 'F' (Fields Management) you will be presented with a screen showing: 'current' window name; 2) the designated sort field; 3) all fields in their current order along with descriptions. Entries m with an asterisk are the currently displayed fields, screen width permitting.

- As the on screen instructions indicate, you navigate among the fields with the **Up** and **Down** arrow keys. The PgDn, Home and End keys can also be used to quickly reach the first or last available field.
- The **Right** arrow key selects a field for repositioning and the **Left** arrow key or the <**Enter**> key commits that placement.
- The '**d**' key or the <**Space**> bar toggles a field's display status, and thus the presence or absence of the asterisk.
- The '**s**' key designates a field as the sort field. See topic 4c. TASK AREA Commands, SORTING for additional mation regarding your selection of a sort field.
- The '**a**' and '**w**' keys can be used to cycle through all available windows and the '**q**' or <**Esc**> keys exit Fields agement.

The Fields Management screen can also be used to change the 'current' window/field group in either full−screen mode or nate−display mode. Whatever was targeted when 'q' or <Esc> was pressed will be made current as you return to the to play. See topic 5. ALTERNATE−DISPLAY Provisions and the 'g' interactive command for insight into 'current' windov field groups.

Note: Any window that has been scrolled *horizontally* will be reset if any field changes are made via the Fields Manag screen. Any *vertical* scrolled position, however, will not be affected. See topic 5c. SCROLLING a Window for addi information regarding vertical and horizontal scrolling.

4. INTERACTIVE Commands

Listed below is a brief index of commands within categories. Some commands appear more than once −− their mean scope may vary depending on the context in which they are issued.

4a. *Global-Commands*
<Ent/Sp> ?, =, 0,
A, B, d, E, e, g, h, H, I, k, q, r, s, W, X, Y, Z
4b. *Summary-Area-Commands*
C, l, t, m, 1, 2, 3
4c. *Task-Area-Commands*
Appearance: b, J, j, x, y, z
Content: c, f, F, o, O, S, u, U, V
Size: #, i, n
Sorting: <, >, f, F, R
4d. *Color-Mapping*
<Ret>, a, B, b, H, M, q, S, T, w, z, 0 − 7
5b. *Commands-for-Windows*
−, _, =, +, A, a, g, G, w
5c. *Scrolling-a-Window*
C, Up, Dn, Left, Right, PgUp, PgDn, Home, End

5d. *Searching-in-a-Window*
L, &

4a. GLOBAL Commands

The global interactive commands are **always** available in both full−screen mode and alternate−display mode. However, of these interactive commands are **not available** when running in Secure mode.

If you wish to know in advance whether or not your top has been secured, simply ask for help and view the system summ the second line.

<**Enter**> or <**Space**> :*Refresh-Display*

These commands awaken top and following receipt of any input the entire display will be repainted. They also an update of any hotplugged cpu or physical memory changes.

Use either of these keys if you have a large delay interval and wish to see current status,

? | **h** :*Help*

There are two help levels available. The first will provide a reminder of all the basic interactive commands. If *secured*, that screen will be abbreviated.

Typing ‘h’ or ‘?’ on that help screen will take you to help for those interactive commands applicable to alternat play mode.

= :*Exit-Task-Limits*

Removes restrictions on which tasks are shown. This command will reverse any ‘i’ (idle tasks) and ‘n’ (max commands that might be active. It also provides for an exit from PID monitoring, User filtering, Other filterin Locate processing. See the ‘−p’ command−line option for a discussion of PID monitoring, the ‘U’ or ‘u’ inter commands for User filtering the ‘O’ or ‘o’ interactive commands for Other filtering and ‘L’ or ‘&’ interactive mands for Locate processing.

Additionally, any window that has been scrolled will be reset with this command. See topic 5c. SCROLLING a dow for additional information regarding vertical and horizontal scrolling.

When operating in alternate−display mode this command has a broader meaning.

0 :*Zero-Suppress* toggle

This command determines whether zeros are shown or suppressed for many of the fields in a task window. Field UID, GID, NI, PR or P are not affected by this toggle.

A :*Alternate-Display-Mode* toggle

This command will switch between full−screen mode and alternate−display mode. See topic 5. ALTERNATE PLAY Provisions and the ‘g’ interactive command for insight into ‘current’ windows and field groups.

B :*Bold-Disable/Enable* toggle

This command will influence use of the bold terminfo capability and alters **both** the summary area and task ar the ‘current’ window. While it is intended primarily for use with dumb terminals, it can be applied anytime.

Note: When this toggle is *On* and top is operating in monochrome mode, the **entire display** will appear as norma Thus, unless the ‘x’ and/or ‘y’ toggles are using reverse for emphasis, there will be no visual confirmation that th even on.

* **d | s** :*Change-Delay-Time-interval*
You will be prompted to enter the delay time, in seconds, between display updates.

Fractional seconds are honored, but a negative number is not allowed. Entering 0 causes (nearly) continuous up
with an unsatisfactory display as the system and tty driver try to keep up with top's demands. The delay va
inversely proportional to system loading, so set it with care.

If at any time you wish to know the current delay time, simply ask for help and view the system summary on th
ond line.

E :*Extend-Memory-Scale* in Summary Area
With this command you can cycle through the available summary area memory scaling which ranges from
(kibibytes or 1,024 bytes) through EiB (exbibytes or 1,152,921,504,606,846,976 bytes).

If you see a '+' between a displayed number and the following label, it means that top was forced to truncate
portion of that number. By raising the scaling factor, such truncation can be avoided.

e :*Extend-Memory-Scale* in Task Windows
With this command you can cycle through the available task window memory scaling which ranges from
(kibibytes or 1,024 bytes) through PiB (pebibytes or 1,125,899,906,842,624 bytes).

While top will try to honor the selected target range, additional scaling might still be necessary in order to acco
date current values. If you wish to see a more homogeneous result in the memory columns, raising the scaling
will usually accomplish that goal. Raising it too high, however, is likely to produce an all zero result which can
suppressed with the '0' interactive command.

g :*Choose-Another-Window/Field-Group*
You will be prompted to enter a number between 1 and 4 designating the field group which should be made the
rent' window. You will soon grow comfortable with these 4 windows, especially after experimenting with
nate−display mode.

H :*Threads-mode* toggle
When this toggle is *On*, individual threads will be displayed for all processes in all visible task windows. Othe
top displays a summation of all threads in each process.

I :*Irix/Solaris-Mode* toggle
When operating in Solaris mode ('I' toggled *Off*), a task's cpu usage will be divided by the total number of
After issuing this command, you'll be told the new state of this toggle.

* **k** :*Kill-a-task*
You will be prompted for a PID and then the signal to send.

Entering no PID or a negative number will be interpreted as the default shown in the prompt (the first task displ
A PID value of zero means the top program itself.

The default signal, as reflected in the prompt, is SIGTERM. However, you can send any signal, via number or na

If you wish to abort the kill process, do one of the following depending on your progress:
1) at the pid prompt, type an invalid number
2) at the signal prompt, type 0 (or any invalid signal)
3) at any prompt, type <Esc>

q :*Quit*

* **r** :*Renice-a-Task*
You will be prompted for a PID and then the value to nice it to.

Entering no PID or a negative number will be interpreted as the default shown in the prompt (the first task displ
A PID value of zero means the top program itself.

A positive nice value will cause a process to lose priority. Conversely, a negative nice value will cause a process
viewed more favorably by the kernel. As a general rule, ordinary users can only increase the nice value and ar
vented from lowering it.

If you wish to abort the renice process, do one of the following depending on your progress:
1) at the pid prompt, type an invalid number
2) at the nice prompt, type <Enter> with no input
3) at any prompt, type <Esc>

W :*Write-the-Configuration-File*
This will save all of your options and toggles plus the current display mode and delay time. By issuing this com
just before quitting top, you will be able restart later in exactly that same state.

X :*Extra-Fixed-Width*
Some fields are fixed width and not scalable. As such, they are subject to truncation which would be indicated by
in the last position.

This interactive command can be used to alter the widths of the following fields:

field	*default*	*field*	*default*	*field*	*default*
GID	5	GROUP	8	WCHAN	10
RUID	5	LXC	8	nsIPC	10
SUID	5	RUSER	8	nsMNT	10
UID	5	SUSER	8	nsNET	10
		TTY	8	nsPID	10
		USER	8	nsUSER	10
				nsUTS	10

You will be prompted for the amount to be added to the default widths shown above. Entering zero forces a ret
those defaults.

If you enter a negative number, top will automatically increase the column size as needed until there is no more
cated data. You can accelerate this process by reducing the delay interval or holding down the <Space> bar.

Note: Whether explicitly or automatically increased, the widths for these fields are never decreased by top. To r
them you must specify a smaller number or restore the defaults.

Y :*Inspect-Other-Output*
After issuing the 'Y' interactive command, you will be prompted for a target PID. Typing a value or acceptin
default results in a separate screen. That screen can be used to view a variety of files or piped command output
the normal top iterative display is paused.

Note: This interactive command is only fully realized when supporting entries have been manually added to the
the top configuration file. For details on creating those entries, see topic 6c. ADDING INSPECT Entries.

Most of the keys used to navigate the Inspect feature are reflected in its header prologue. There are, however, tional keys available once you have selected a particular file or command. They are familiar to anyone who ha the pager 'less' and are summarized here for future reference.

key	*function*
=	alternate status−line, file or pipeline
/	find, equivalent to 'L' locate
n	find next, equivalent to '&' locate next
<Space>	scroll down, equivalent to <PgDn>
b	scroll up, equivalent to <PgUp>
g	first line, equivalent to <Home>
G	last line, equivalent to <End>

Z :*Change-Color-Mapping*
This key will take you to a separate screen where you can change the colors for the 'current' window, or for al dows. For details regarding this interactive command see topic 4d. COLOR Mapping.

* The commands shown with an asterisk ('*') are not available in Secure mode, nor will they be shown on the level- screen.

4b. SUMMARY AREA Commands

The summary area interactive commands are **always available** in both full−screen mode and alternate−display mode. affect the beginning lines of your display and will determine the position of messages and prompts.

These commands always impact just the 'current' window/field group. See topic 5. ALTERNATE−DISPLAY Provisio the 'g' interactive command for insight into 'current' windows and field groups.

C :*Show-scroll-coordinates* toggle
Toggle an informational message which is displayed whenever the message line is not otherwise being used. Fo tional information see topic 5c. SCROLLING a Window.

l :*Load-Average/Uptime* toggle
This is also the line containing the program name (possibly an alias) when operating in full−screen mode or the rent' window name when operating in alternate−display mode.

t :*Task/Cpu-States* toggle
This command affects from 2 to many summary area lines, depending on the state of the '1', '2' or '3' comman gles and whether or not top is running under true SMP.

This portion of the summary area is also influenced by the 'H' interactive command toggle, as reflected in th label which shows either Tasks or Threads.

This command serves as a 4-way toggle, cycling through these modes:
1. detailed percentages by category
2. abbreviated user/system and total % + bar graph
3. abbreviated user/system and total % + block graph
4. turn off task and cpu states display

When operating in either of the graphic modes, the display becomes much more meaningful when individual CP NUMA nodes are also displayed. See the the '1', '2' and '3' commands below for additional information.

m :*Memory/Swap-Usage* toggle
This command affects the two summary area lines dealing with physical and virtual memory.

This command serves as a 4-way toggle, cycling through these modes:
1. detailed percentages by memory type
2. abbreviated % used/total available + bar graph
3. abbreviated % used/total available + block graph
4. turn off memory display

1 :*Single/Separate-Cpu-States* toggle
This command affects how the 't' command's Cpu States portion is shown. Although this toggle exists prima
serve massively-parallel SMP machines, it is not restricted to solely SMP environments.

When you see '%Cpu(s):' in the summary area, the '1' toggle is *On* and all cpu information is gathered in a singl
Otherwise, each cpu is displayed separately as: '%Cpu0, %Cpu1, ...' up to available screen height.

2 :*NUMA-Nodes/Cpu-Summary* toggle
This command toggles between the '1' command cpu summary display (only) or a summary display plus th
usage statistics for each NUMA Node. It is only available if a system has the requisite NUMA support.

3 :*Expand-NUMA-Node*
You will be invited to enter a number representing a NUMA Node. Thereafter, a node summary plus the statist
each cpu in that node will be shown until either the '1' or '2' command toggle is pressed. This interactive comm
only available if a system has the requisite NUMA support.

Note: If the entire summary area has been toggled *Off* for any window, you would be left with just the **message line**. I
way, you will have maximized available task rows but (temporarily) sacrificed the program name in full−screen mode
'current' window name when in alternate−display mode.

4c. TASK AREA Commands

The task area interactive commands are **always** available in full−screen mode.

The task area interactive commands are **never available** in alternate−display mode *if* the 'current' window's task displa
been toggled *Off* (see topic 5. ALTERNATE−DISPLAY Provisions).

APPEARANCE of task window

J :*Justify-Numeric-Columns* toggle
Alternates between right-justified (the default) and left-justified numeric data. If the numeric data completely fi
available column, this command toggle may impact the column header only.

j :*Justify-Character-Columns* toggle
Alternates between left-justified (the default) and right-justified character data. If the character data completel
the available column, this command toggle may impact the column header only.

The following commands will also be influenced by the state of the global 'B' (bold enable) toggle.

b :*Bold/Reverse* toggle
This command will impact how the 'x' and 'y' toggles are displayed. It may also impact the summary area when
graph has been selected for cpu states or memory usage via the 't' or 'm' toggles.

x :*Column-Highlight* toggle
Changes highlighting for the current sort field. If you forget which field is being sorted this command can serv
quick visual reminder, providing the sort field is being displayed. The sort field might *not* be visible because:
1) there is insufficient *Screen Width*
2) the ‘f’ interactive command turned it *Off*

Note: Whenever Searching and/or Other Filtering is active in a window, column highlighting is temporarily dis
See the notes at the end of topics 5d. SEARCHING and 5e. FILTERING for an explanation why.

y :*Row-Highlight* toggle
Changes highlighting for "running" tasks. For additional insight into this task state, see topic 3a. DESCRIPTIO
Fields, the ‘S’ field (Process Status).

Use of this provision provides important insight into your system’s health. The only costs will be a few additio
escape sequences.

z :*Color/Monochrome* toggle
Switches the ‘current’ window between your last used color scheme and the older form of black-on-white or whi
black. This command will alter **both** the summary area and task area but does not affect the state of the ‘x’, ‘y’
toggles.

CONTENT of task window

c :*Command-Line/Program-Name* toggle
This command will be honored whether or not the COMMAND column is currently visible. Later, should tha
come into view, the change you applied will be seen.

f | **F** :*Fields-Management*
These keys display a separate screen where you can change which fields are displayed, their order and also des
the sort field. For additional information on these interactive commands see topic 3b. MANAGING Fields.

o | **O** :*Other-Filtering*
You will be prompted for the selection criteria which then determines which tasks will be shown in the ‘current
dow. Your criteria can be made case sensitive or case can be ignored. And you determine if top should inclu
exclude matching tasks.

See topic 5e. FILTERING in a window for details on these and additional related interactive commands.

S :*Cumulative-Time-Mode* toggle
When Cumulative mode is *On*, each process is listed with the cpu time that it and its dead children have used.

When *Off*, programs that fork into many separate tasks will appear less demanding. For programs like ‘init’ or a
this is appropriate but for others, like compilers, perhaps not. Experiment with two task windows sharing the
sort field but with different ‘S’ states and see which representation you prefer.

After issuing this command, you’ll be informed of the new state of this toggle. If you wish to know in ad
whether or not Cumulative mode is in effect, simply ask for help and view the window summary on the second li

u | **U** :*Show-Specific-User-Only*
You will be prompted for the **uid** or **name** of the user to display. The −u option matches on **effective** user where
−U option matches on **any** user (real, effective, saved, or filesystem).

Thereafter, in that task window only matching users will be shown, or possibly no processes will be shown. Pre
ing an exclamation point ('!') to the user id or name instructs top to display only processes with users not matchi
one provided.

Different task windows can be used to filter different users. Later, if you wish to monitor all users again in the
rent' window, re-issue this command but just press <Enter> at the prompt.

V :*Forest-View-Mode* toggle
In this mode, processes are reordered according to their parents and the layout of the COMMAND column rese
that of a tree. In forest view mode it is still possible to toggle between program name and command line (see t
interactive command) or between processes and threads (see the 'H' interactive command).

Note: Typing any key affecting the sort order will exit forest view mode in the 'current' window. See topic 4c.
AREA Commands, SORTING for information on those keys.

SIZE of task window

i :*Idle-Process* toggle
Displays all tasks or just active tasks. When this toggle is *Off*, tasks that have not used any CPU since the last u
will not be displayed. However, due to the granularity of the %CPU and TIME+ fields, some processes may s
displayed that *appear* to have used *no* CPU.

If this command is applied to the last task display when in alternate−display mode, then it will not affect the win
size, as all prior task displays will have already been painted.

n | **#** :*Set-Maximum-Tasks*
You will be prompted to enter the number of tasks to display. The lessor of your number and available screen
will be used.

When used in alternate−display mode, this is the command that gives you precise control over the size of eac
rently visible task display, except for the very last. It will not affect the last window's size, as all prior task di
will have already been painted.

Note: If you wish to increase the size of the last visible task display when in alternate−display mode, simply de
the size of the task display(s) above it.

SORTING of task window

For compatibility, this top supports most of the former top sort keys. Since this is primarily a service to former top
these commands do not appear on any help screen.

command	*sorted-field*	*supported*
A	start time (non-display)	**No**
M	%MEM	Yes
N	PID	Yes
P	%CPU	Yes
T	TIME+	Yes

Before using any of the following sort provisions, top suggests that you temporarily turn on column highlighting usi
'x' interactive command. That will help ensure that the actual sort environment matches your intent.

The following interactive commands will **only** be honored when the current sort field is **visible**. The sort field might
visible because:
1) there is insufficient *Screen Width*
2) the 'f' interactive command turned it *Off*

< :*Move-Sort-Field-Left*
Moves the sort column to the left unless the current sort field is the first field being displayed.

> :*Move-Sort-Field-Right*
Moves the sort column to the right unless the current sort field is the last field being displayed.

The following interactive commands will **always** be honored whether or not the current sort field is visible.

f | **F** :*Fields-Management*
These keys display a separate screen where you can change which field is used as the sort column, among functions. This can be a convenient way to simply verify the current sort field, when running top with c highlighting turned *Off*.

R :*Reverse/Normal-Sort-Field* toggle
Using this interactive command you can alternate between high-to-low and low-to-high sorts.

Note: Field sorting uses internal values, not those in column display. Thus, the TTY and WCHAN fields will violate ASCII collating sequence.

4d. COLOR Mapping

When you issue the 'Z' interactive command, you will be presented with a separate screen. That screen can be used to c the colors in just the 'current' window or in all four windows before returning to the top display.

The following interactive commands are available.
4 upper case letters to select a **target**
8 numbers to select a **color**
normal toggles available
B :bold disable/enable
b :running tasks "bold"/reverse
z :color/mono
other commands available
a/w :apply, then go to next/prior
<Enter> :apply and exit
q :abandon current changes and exit

If you use 'a' or 'w' to cycle the targeted window, you will have applied the color scheme that was displayed when you le window. You can, of course, easily return to any window and reapply different colors or turn colors *Off* completely with t toggle.

The Color Mapping screen can also be used to change the 'current' window/field group in either full−screen mode or nate−display mode. Whatever was targeted when 'q' or <Enter> was pressed will be made current as you return to the to play.

5. ALTERNATE−DISPLAY Provisions

5a. WINDOWS Overview

Field Groups/Windows:
In full−screen mode there is a single window represented by the entire screen. That single window can still be chan display 1 of 4 different **field groups** (see the 'g' interactive command, repeated below). Each of the 4 field groups unique separately configurable **summary area** and its own configurable **task area**.

In alternate−display mode, those 4 underlying field groups can now be made visible simultaneously, or can be turn individually at your command.

The summary area will always exist, even if it's only the message line. At any given time only *one* summary area d displayed. However, depending on your commands, there could be from *zero* to *four* separate task displays currently ing on the screen.

Current Window:
The 'current' window is the window associated with the summary area and the window to which task related comman always directed. Since in alternate−display mode you can toggle the task display *Off*, some commands might be res for the 'current' window.

A further complication arises when you have toggled the first summary area line *Off*. With the loss of the window (the 'l' toggled line), you'll not easily know what window is the 'current' window.

5b. COMMANDS for Windows

- | _ :*Show/Hide-Window(s)* toggles
The '−' key turns the 'current' window's task display *On* and *Off*. When *On*, that task area will show a minim the columns header you've established with the 'f' interactive command. It will also reflect any other tas options/toggles you've applied yielding zero or more tasks.

The '_' key does the same for all task displays. In other words, it switches between the currently visible tas play(s) and any task display(s) you had toggled *Off*. If all 4 task displays are currently visible, this interactive mand will leave the summary area as the only display element.

* **= | +** :*Equalize-(reinitialize)-Window(s)*
The '=' key forces the 'current' window's task display to be visible. It also reverses any 'i' (idle tasks), 'n' tasks), 'u/U' (user filter), `o/O' (other filter) and 'L' (locate) commands that might be active. Also, if the windo been scrolled, it will be reset with this command. See topic 5c. SCROLLING a Window for additional inform regarding vertical and horizontal scrolling.

The '+' key does the same for all windows. The four task displays will reappear, evenly balanced. They will also retained any customizations you had previously applied, except for the 'i' (idle tasks), 'n' (max tasks), 'u/U' (us ter), 'o/O' (other filter), 'L' (locate) and scrolling interactive commands.

* **A** :*Alternate-Display-Mode* toggle
This command will switch between full−screen mode and alternate−display mode.

The first time you issue this command, all four task displays will be shown. Thereafter when you switch mode will see only the task display(s) you've chosen to make visible.

* **a | w** :*Next-Window-Forward/Backward*
This will change the 'current' window, which in turn changes the window to which commands are directed. keys act in a circular fashion so you can reach any desired window using either key.

Assuming the window name is visible (you have not toggled 'l' *Off*), whenever the 'current' window name lo emphasis/color, that's a reminder the task display is *Off* and many commands will be restricted.

* **g** :*Choose-Another-Window/Field-Group*
You will be prompted to enter a number between 1 and 4 designating the field group which should be made the rent' window.

In full−screen mode, this command is necessary to alter the 'current' window. In alternate−display mode, it is s a less convenient alternative to the 'a' and 'w' commands.

G :*Change-Window/Field-Group-Name*
You will be prompted for a new name to be applied to the 'current' window. It does not require that the window be visible (the 'l' toggle to be *On*).

* The interactive commands shown with an asterisk ('*') have use beyond alternate−display mode.
=, A, g are always available
a, w act the same with color mapping
and fields management

5c. SCROLLING a Window

Typically a task window is a partial view into a systems's total tasks/threads which shows only some of the available field umns. With these scrolling keys, you can move that view vertically or horizontally to reveal any desired task or column.

Up,**PgUp** :*Scroll-Tasks*
Move the view up toward the first task row, until the first task is displayed at the top of the 'current' window. T arrow key moves a single line while *PgUp* scrolls the entire window.

Down,**PgDn** :*Scroll-Tasks*
Move the view down toward the last task row, until the last task is the only task displayed at the top of the 'current dow. The *Down* arrow key moves a single line while *PgDn* scrolls the entire window.

Left,**Right** :*Scroll-Columns*
Move the view of displayable fields horizontally one column at a time.

Note: As a reminder, some fields/columns are not fixed-width but allocated all remaining screen width when v When scrolling right or left, that feature may produce some unexpected results initially.

Additionally, there are special provisions for any variable width field when positioned as the last displayed field. that field is reached via the right arrow key, and is thus the only column shown, you can continue scrolling horizo within such a field. See the 'C' interactive command below for additional information.

Home :*Jump-to-Home-Position*
Reposition the display to the un-scrolled coordinates.

End :*Jump-to-End-Position*
Reposition the display so that the rightmost column reflects the last displayable field and the bottom task row repr the last task.

Note: From this position it is still possible to scroll *down* and *right* using the arrow keys. This is true until a single c and a single task is left as the only display element.

C :*Show-scroll-coordinates* toggle
Toggle an informational message which is displayed whenever the message line is not otherwise being used. Tha sage will take one of two forms depending on whether or not a variable width column has also been scrolled.

scroll coordinates: y = n/n (tasks), x = n/n (fields)
scroll coordinates: y = n/n (tasks), x = n/n (fields) **+ nn**

The coordinates shown as **n**/**n** are relative to the upper left corner of the 'current' window. The additional '**+ nn**' sents the displacement into a variable width column when it has been scrolled horizontally. Such displacement occ normal 8 character tab stop amounts via the right and left arrow keys.

y = n/n (tasks)
The first **n** represents the topmost visible task and is controlled by scrolling keys. The second **n** is updated auto
cally to reflect total tasks.

x = n/n (fields)
The first **n** represents the leftmost displayed column and is controlled by scrolling keys. The second **n** is th
number of displayable fields and is established with the '**f**' interactive command.

The above interactive commands are **always** available in full−screen mode but **never** available in alternate−display mode
'current' window's task display has been toggled *Off*.

Note: When any form of filtering is active, you can expect some slight aberrations when scrolling since not all tasks will b
ible. This is particularly apparent when using the Up/Down arrow keys.

5d. SEARCHING in a Window

You can use these interactive commands to locate a task row containing a particular value.

L :*Locate-a-string*
You will be prompted for the case-sensitive string to locate starting from the current window coordinates. There
restrictions on search string content.

Searches are not limited to values from a single field or column. All of the values displayed in a task row are allowe
search string. You may include spaces, numbers, symbols and even forest view artwork.

Keying <Enter> with no input will effectively disable the '&' key until a new search string is entered.

& :*Locate-next*
Assuming a search string has been established, top will attempt to locate the next occurrence.

When a match is found, the current window is repositioned vertically so the task row containing that string is first. The
coordinates message can provide confirmation of such vertical repositioning (see the 'C' interactive command). Hori
scrolling, however, is never altered via searching.

The availability of a matching string will be influenced by the following factors.

a. Which fields are displayable from the total available,
see topic 3b. MANAGING Fields.

b. Scrolling a window vertically and/or horizontally,
see topic 5c. SCROLLING a Window.

c. The state of the command/command-line toggle,
see the 'c' interactive command.

d. The stability of the chosen sort column,
for example PID is good but %CPU bad.

If a search fails, restoring the 'current' window home (unscrolled) position, scrolling horizontally, displaying command-li
choosing a more stable sort field could yet produce a successful '&' search.

The above interactive commands are **always** available in full−screen mode but **never** available in alternate−display mode
'current' window's task display has been toggled *Off*.

Note: Whenever a Search is active in a window, top will turn column highlighting *Off* to prevent false matches on interna

display escape sequences. Such highlighting will be restored when a window's search string is empty. See the 'x' inter command for additional information on sort column highlighting.

5e. FILTERING in a Window

You can use this Other Filter feature to establish selection criteria which will then determine which tasks are shown in the rent' window.

Establishing a filter requires: 1) a field name; 2) an operator; and 3) a selection value, as a minimum. This is the most co of top's user input requirements so, when you make a mistake, command recall will be your friend. Remember the Up/ arrow keys or their aliases when prompted for input.

Filter Basics

1. field names are case sensitive and spelled as in the header
2. selection values need not comprise the full displayed field
3. a selection is either case insensitive or sensitive to case
4. the default is inclusion, prepending '!' denotes exclusions
5. multiple selection criteria can be applied to a task window
6. inclusion and exclusion criteria can be used simultaneously
7. the 1 equality and 2 relational filters can be freely mixed
8. separate unique filters are maintained for each task window

If a field is not turned on or is not currently in view, then your selection criteria will not affect the display. Later, sh filtered field become visible, the selection criteria will then be applied.

Keyboard Summary

o :*Other-Filter* (lower case)
You will be prompted to establish a filter that **ignores case** when matching.

O :*Other-Filter* (upper case)
You will be prompted to establish a **case sensitive** filter.

^O :*Show-Active-Filters* (Ctrl key + 'o')
This can serve as a reminder of which filters are active in the 'current' window. A summary will be shown on the sage line until you press the <Enter> key.

= :*Reset-Filtering* in current window
This clears all of your selection criteria in the 'current' window. It also has additional impact so please see top GLOBAL Commands.

+ :*Reset-Filtering* in all windows
This clears the selection criteria in all windows, assuming you are in alternate−display mode. As with the '=' in tive command, it too has additional consequences so you might wish to see topic 5b. COMMANDS for Windows.

Input Requirements

When prompted for selection criteria, the data you provide must take one of two forms. There are 3 required pie information, with a 4th as optional. These examples use spaces for clarity but your input generally would not.

```
  #1        #2  #3          ( required )
   Field−Name   ?  include−if−value
 ! Field−Name   ?  exclude−if−value
 #4                      ( optional )
```

Items #1, #3 and #4 should be self−explanatory. Item **#2** represents both a required *delimiter* and the *operator* which be one of either equality (‘=’) or relation (‘<’ or ‘>’).

The ‘=’ equality operator requires only a partial match and that can reduce your ‘if−value’ input requirements. The ‘<’ relational operators always employ string comparisons, even with numeric fields. They are designed to work field’s default *justification* and with homogeneous data. When some field’s numeric amounts have been subjected to *s* while others have not, that data is no longer homogeneous.

If you establish a relational filter and you **have** changed the default Numeric or Character *justification*, that filter is lik fail. When a relational filter is applied to a memory field and you **have not** changed the *scaling*, it may produce misle results. This happens, for example, because ‘100.0m’ (MiB) would appear greater than ‘1.000g’ (GiB) when compa strings.

If your filtered results appear suspect, simply altering justification or scaling may yet achieve the desired objective. S ‘j’, ‘J’ and ‘e’ interactive commands for additional information.

Potential Problems

These **GROUP** filters could produce the exact same results or the second one might not display anything at all, just a task window.

```
GROUP=root      ( only the same results when )
GROUP=ROOT      ( invoked via lower case ‘o’ )
```

Either of these **RES** filters might yield inconsistent and/or misleading results, depending on the current memory scalin tor. Or both filters could produce the exact same results.

```
RES>9999        ( only the same results when )
!RES<10000      ( memory scaling is at ‘KiB’ )
```

This **nMin** filter illustrates a problem unique to scalable fields. This particular field can display a maximum of 4 beyond which values are automatically scaled to KiB or above. So while amounts greater than 9999 exist, they will a as 2.6m, 197k, etc.

```
nMin>9999       ( always a blank task window )
```

Potential Solutions

These examples illustrate how Other Filtering can be creatively applied to achieve almost any desired result. Single are sometimes shown to delimit the spaces which are part of a filter or to represent a request for status (^O) accurately if you used them with if-values in real life, no matches would be found.

Assuming field **nTH** is displayed, the first filter will result in only multi-threaded processes being shown. It also remi that a trailing space is part of every displayed field. The second filter achieves the exact same results with less typing.

```
!nTH=‘ 1 ’              ( ’ for clarity only )
nTH>1                   ( same with less i/p )
```

With Forest View mode active and the **COMMAND** column in view, this filter effectively collapses child processes s just 3 levels are shown.

```
!COMMAND=‘      ‘- ’    ( ’ for clarity only )
```

The final two filters appear as in response to the status request key (^O). In reality, each filter would have required se input. The **PR** example shows the two concurrent filters necessary to display tasks with priorities of 20 or more, since might be negative. Then by exploiting trailing spaces, the **nMin** series of filters could achieve the failed ‘9999’ obj discussed above.

```
‘PR>20’ + ‘!PR=-’       ( 2 for right result )
‘!nMin=0 ’ + ‘!nMin=1 ’ + ‘!nMin=2 ’ + ‘!nMin=3 ’ ...
```

Note: Whenever Other Filtering is active in a window, top will turn column highlighting *Off* to prevent false matches on nal non-display escape sequences. Such highlighting will be restored when a window is no longer subject to filtering. S

'x' interactive command for additional information on sort column highlighting.

6. FILES

6a. SYSTEM Configuration File

The presence of this file will influence which version of the help screen is shown to an ordinary user. More importantly, limit what ordinary users are allowed to do when top is running. They will not be able to issue the following commands.

```
k       Kill a task
r       Renice a task
d or s  Change delay/sleep interval
```

The system configuration file is **not** created by top. Rather, you create this file manually and place it in the */etc* directo name must be 'toprc' and must have no leading '.' (period). It must have only two lines.

Here is an example of the contents of */etc/toprc*:

```
s       # line 1: secure mode switch
5.0     # line 2: delay interval in seconds
```

6b. PERSONAL Configuration File

This file is written as '$HOME/.your−name−4−top' + 'rc'. Use the 'W' interactive command to create it or update it.

Here is the general layout:

```
global   # line  1: the program name/alias notation
  "      # line  2: id,altscr,irixps,delay,curwin
per ea   # line  a: winname,fieldscur
window   # line  b: winflags,sortindx,maxtasks,graph modes
  "      # line  c: summclr,msgsclr,headclr,taskclr
global   # line 15: additional miscellaneous settings
  "      # any remaining lines are devoted to the
  "      # generalized inspect provisions
  "      # discussed below
```

If the $HOME variable is not present, top will try to write the personal configuration file to the current directory, subject t missions.

6c. ADDING INSPECT Entries

To exploit the 'Y' interactive command, you must add entries at the **end** of the top personal configuration file. Such e simply reflect a file to be read or command/pipeline to be executed whose results will then be displayed in a separate scro searchable window.

If you don't know the location or name of your top rcfile, use the 'W' interactive command to rewrite it and note those det

Inspect entries can be added with a redirected echo or by editing the configuration file. Redirecting an echo risks overw the rcfile should it replace (>) rather than append (>>) to that file. Conversely, when using an editor care must be taken corrupt existing lines, some of which will contain unprintable data or unusual characters.

Those Inspect entries beginning with a '#' character are ignored, regardless of content. Otherwise they consist of the foll 3 elements, each of which *must* be separated by a tab character (thus 2 '\t' total):

```
.type:  literal 'file' or 'pipe'
.name:  selection shown on the Inspect screen
.fmts:  string representing a path or command
```

The two types of Inspect entries are *not* interchangeable. Those designated '**file**' will be accessed using fopen and

reference a single file in the '.fmts' element. Entries specifying '**pipe**' will employ popen, their '.fmts' element could c
many pipelined commands and, none can be interactive.

If the file or pipeline represented in your '.fmts' deals with the specific PID input or accepted when prompted, then the f
string must also contain the '**%d**' specifier, as these examples illustrate.

```
.fmts=  /proc/%d/numa_maps
.fmts=  lsof -P -p %d
```

For '**pipe**' type entries only, you may also wish to redirect stderr to stdout for a more comprehensive result. Thus the f
string becomes:

```
.fmts=  pmap -x %d 2>&1
```

Here are examples of both types of Inspect entries as they might appear in the rcfile. The first entry will be ignored due
initial '#' character. For clarity, the pseudo tab depictions (^I) are surrounded by an extra space but the actual tabs wou
be.

```
# pipe ^I Sockets ^I lsof -n -P -i 2>&1
pipe ^I Open Files ^I lsof -P -p %d 2>&1
file ^I NUMA Info ^I /proc/%d/numa_maps
pipe ^I Log ^I tail -n100 /var/log/syslog | sort -Mr
```

Except for the commented entry above, these next examples show what could be echoed to achieve similar results, ass
the rcfile name was '.toprc'. However, due to the embedded tab characters, each of these lines should be preced
'**/bin/echo −e**', not just a simple an 'echo', to enable backslash interpretation regardless of which shell you use.

```
"pipe\tOpen Files\tlsof -P -p %d 2>&1" >> ~/.toprc
"file\tNUMA Info\t/proc/%d/numa_maps" >> ~/.toprc
"pipe\tLog\ttail -n200 /var/log/syslog | sort -Mr" >> ~/.toprc
```

Caution: If any inspect entry you create produces output with unprintable characters they will be displayed in either
notation or hexadecimal <FF> form, depending on their value. This applies to tab characters as well, which will show a
If you want a truer representation, any embedded tabs should be expanded.

```
# next would have contained '\t' ...
# file ^I <your_name> ^I /proc/%d/status
# but this will eliminate embedded '\t' ...
pipe ^I <your_name> ^I cat /proc/%d/status | expand −
```

The above example takes what could have been a 'file' entry but employs a 'pipe' instead so as to expand the embedded ta

Note: While '**pipe**' type entries have been discussed in terms of pipelines and commands, there is nothing to prevent you
including *shell scripts* as well. Perhaps even newly created scripts designed specifically for the 'Y' interactive command.

Lastly, as the number of your Inspect entries grows over time, the 'Options:' row will be truncated when screen wi
exceeded. That does not affect operation other than to make some selections invisible.

However, if some choices are lost to truncation but you want to see more options, there is an easy solution hinted at below

```
Inspection Pause at pid ...
Use:  left/right then <Enter> ...
Options:  help  1  2  3  4  5  6  7  8  9  10  11 ...
```

The entries in the top rcfile would have a number for the '.name' element and the 'help' entry would identify a shell
you've written explaining what those numbered selections actually mean. In that way, many more choices can be made vi

7. STUPID TRICKS Sampler

Many of these tricks work best when you give top a scheduling boost. So plan on starting him with a nice value of −10, a
ing you've got the authority.

7a. Kernel Magic

For these stupid tricks, top needs full−screen mode.

- The user interface, through prompts and help, intentionally implies that the delay interval is limited to tenths of a se
  However, you're free to set any desired delay. If you want to see Linux at his scheduling best, try a delay of .09 seco
  less.

  For this experiment, under x-windows open an xterm and maximize it. Then do the following:
  . provide a scheduling boost and tiny delay via:
  nice -n -10 top -d.09
  . keep sorted column highlighting *Off* so as to
  minimize path length
  . turn *On* reverse row highlighting for emphasis
  . try various sort columns (TIME/MEM work well),
  and normal or reverse sorts to bring the most
  active processes into view

  What you'll see is a very busy Linux doing what he's always done for you, but there was no program available to illu
  this.

- Under an xterm using 'white-on-black' colors, on top's Color Mapping screen set the task color to black and be sur
  task highlighting is set to bold, not reverse. Then set the delay interval to around .3 seconds.

  After bringing the most active processes into view, what you'll see are the ghostly images of just the currently ru
  tasks.

- Delete the existing rcfile, or create a new symlink. Start this new version then type 'T' (a secret key, see topic 4c. Task
  Commands, SORTING) followed by 'W' and 'q'. Finally, restart the program with −d0 (zero delay).

  Your display will be refreshed at three times the rate of the former top, a 300% speed advantage. As top climbs the
  ladder, be as patient as you can while speculating on whether or not top will ever reach the top.

7b. Bouncing Windows

For these stupid tricks, top needs alternate−display mode.

- With 3 or 4 task displays visible, pick any window other than the last and turn idle processes *Off* using the 'i' comman
  gle. Depending on where you applied 'i', sometimes several task displays are bouncing and sometimes it's like an
  dion, as top tries his best to allocate space.

- Set each window's summary lines differently: one with no memory ('m'); another with no states ('t'); maybe one with
  ing at all, just the message line. Then hold down 'a' or 'w' and watch a variation on bouncing windows −− hopping
  dows.

- Display all 4 windows and for each, in turn, set idle processes to *Off* using the 'i' command toggle. You've just enter
  "extreme bounce" zone.

7c. The Big Bird Window

This stupid trick also requires alternate−display mode.

- Display all 4 windows and make sure that 1:Def is the ‘current’ window. Then, keep increasing window size with t interactive command until all the other task displays are "pushed out of the nest".

  When they’ve all been displaced, toggle between all visible/invisible windows using the ‘_’ command toggle. Then p this:

  is top fibbing or telling honestly your imposed truth?

7d. The Ol’ Switcheroo

This stupid trick works best without alternate−display mode, since justification is active on a per window basis.

- Start top and make COMMAND the last (rightmost) column displayed. If necessary, use the ‘c’ command toggle to d command lines and ensure that forest view mode is active with the ‘V’ command toggle.

  Then use the up/down arrow keys to position the display so that some truncated command lines are shown (‘+’ in las tion). You may have to resize your xterm to produce truncation.

  Lastly, use the ‘j’ command toggle to make the COMMAND column right justified.

  Now use the right arrow key to reach the COMMAND column. Continuing with the right arrow key, watch close direction of travel for the command lines being shown.

  some lines travel left, while others travel right

  eventually all lines will Switcheroo, and move right

8. BUGS

Please send bug reports to ⟨procps@freelists.org⟩.

9. SEE Also

free(1), **ps**(1), **uptime**(1), **atop**(1), **slabtop**(1), **vmstat**(8), **w**(1)

NAME

tr − translate or delete characters

SYNOPSIS

tr [*OPTION*]... *SET1* [*SET2*]

DESCRIPTION

Translate, squeeze, and/or delete characters from standard input, writing to standard output.

−c, **−C**, **−−complement**
use the complement of SET1

−d, **−−delete**
delete characters in SET1, do not translate

−s, **−−squeeze−repeats**
replace each sequence of a repeated character that is listed in the last specified SET, with a single occurrence of that character

−t, **−−truncate−set1**
first truncate SET1 to length of SET2

−−help display this help and exit

−−version
output version information and exit

SETs are specified as strings of characters. Most represent themselves. Interpreted sequences are:

\NNN character with octal value NNN (1 to 3 octal digits)

\\ backslash

\a audible BEL

\b backspace

\f form feed

\n new line

\r return

\t horizontal tab

\v vertical tab

CHAR1−CHAR2
all characters from CHAR1 to CHAR2 in ascending order

[CHAR*]
in SET2, copies of CHAR until length of SET1

[CHAR*REPEAT]
REPEAT copies of CHAR, REPEAT octal if starting with 0

[:alnum:]
all letters and digits

[:alpha:]
all letters

[:blank:]
all horizontal whitespace

[:cntrl:] all control characters

[:digit:] all digits

[:graph:]
all printable characters, not including space

[:lower:]
all lower case letters

[:print:]
all printable characters, including space

[:punct:]
all punctuation characters

[:space:]
all horizontal or vertical whitespace

[:upper:]
all upper case letters

[:xdigit:]
all hexadecimal digits

[=CHAR=]
all characters which are equivalent to CHAR

Translation occurs if **−d** is not given and both SET1 and SET2 appear. **−t** may be used only when translating. SET2 is extended to length of SET1 by repeating its last character as necessary. Excess characters of SET2 are ignored. Only [:lower:] and [:upper:] are guaranteed to expand in ascending order; used in SET2 while translating, they may only be used in pairs to specify case conversion. **−s** uses the last specified SET, and occurs after translation or deletion.

AUTHOR

Written by Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Report tr translation bugs to <http://translationproject.org/team/>

COPYRIGHT

SEE ALSO

Full documentation at: <http://www.gnu.org/software/coreutils/tr>
or available locally via: info '(coreutils) tr invocation'

NAME

vim − Vi IMproved, a programmers text editor

SYNOPSIS

vim [options] [file ..]
vim [options] −
vim [options] −t tag
vim [options] −q [errorfile]

ex
view
gvim gview evim eview
rvim rview rgvim rgview

DESCRIPTION

Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between **Vim** and Vi.

While running **Vim** a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below.

Most often **Vim** is started to edit a single file with the command

vim file

More generally **Vim** is started with:

vim [options] [filelist]

If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited.

file .. A list of filenames. The first one will be the current file and read into the buffer. The cursor will be positioned on the first line of the buffer. You can get to the other files with the ":next" command. To edit a file that starts with a dash, precede the filelist with "−−".

− The file to edit is read from stdin. Commands are read from stderr, which should be a tty.

−t {tag} The file to edit and the initial cursor position depends on a "tag", a sort of goto label. {tag} is looked up in the tags file, the associated file becomes the current file and the associated command is executed. Mostly this is used for C programs, in which case {tag} could be a function name. The effect is that the file containing that function becomes the current file and the cursor is positioned on the start of the function. See ":help tag−commands".

−q [errorfile] Start in quickFix mode. The file [errorfile] is read and the first error is displayed. If [errorfile] is omitted, the filename is obtained from the 'errorfile' option (defaults to "AztecC.Err" for the Amiga, "errors.err" on other systems). Further errors can be jumped to with the ":cn" command. See ":help quickfix".

Vim behaves differently, depending on the name of the command (the executable may still be the same file).

vim The "normal" way, everything is default.

ex Start in Ex mode. Go to Normal mode with the ":vi" command. Can also be done with the "−e" argument.

view Start in read-only mode. You will be protected from writing the files. Can also be done with the "−R" argument.

gvim gview
: The GUI version. Starts a new window. Can also be done with the "−g" argument.

evim eview
: The GUI version in easy mode. Starts a new window. Can also be done with the "−y" argument.

rvim rview rgvim rgview
: Like the above, but with restrictions. It will not be possible to start shell commands, or suspend **Vim.** Can also be done with the "−Z" argument.

OPTIONS

The options may be given in any order, before or after filenames. Options without an argument can be combined after a single dash.

+[num] For the first file the cursor will be positioned on line "num". If "num" is missing, the cursor will be positioned on the last line.

+/{pat} For the first file the cursor will be positioned in the line with the first occurrence of {pat}. See ":help search−pattern" for the available search patterns.

+{command}

−c {command}
: {command} will be executed after the first file has been read. {command} is interpreted as an Ex command. If the {command} contains spaces it must be enclosed in double quotes (this depends on the shell that is used). Example: Vim "+set si" main.c
Note: You can use up to 10 "+" or "−c" commands.

−S {file} {file} will be sourced after the first file has been read. This is equivalent to −c "source {file}". {file} cannot start with '−'. If {file} is omitted "Session.vim" is used (only works when −S is the last argument).

−−cmd {command}
: Like using "−c", but the command is executed just before processing any vimrc file. You can use up to 10 of these commands, independently from "−c" commands.

−A If **Vim** has been compiled with ARABIC support for editing right-to-left oriented files and Arabic keyboard mapping, this option starts **Vim** in Arabic mode, i.e. 'arabic' is set. Otherwise an error message is given and **Vim** aborts.

−b Binary mode. A few options will be set that makes it possible to edit a binary or executable file.

−C Compatible. Set the 'compatible' option. This will make **Vim** behave mostly like Vi, even though a .vimrc file exists.

−d Start in diff mode. There should be two, three or four file name arguments. **Vim** will open all the files and show differences between them. Works like vimdiff(1).

−d {device} Open {device} for use as a terminal. Only on the Amiga. Example: "−d con:20/30/600/150".

−D Debugging. Go to debugging mode when executing the first command from a script.

−e Start **Vim** in Ex mode, just like the executable was called "ex".

−E Start **Vim** in improved Ex mode, just like the executable was called "exim".

−f Foreground. For the GUI version, **Vim** will not fork and detach from the shell it was started in. On the Amiga, **Vim** is not restarted to open a new window. This option should be used when **Vim** is executed by a program that will wait for the edit session to finish (e.g. mail). On the Amiga the ":sh" and ":!" commands will not work.

−−nofork Foreground. For the GUI version, **Vim** will not fork and detach from the shell it was started in.

−F If **Vim** has been compiled with FKMAP support for editing right-to-left oriented files and Farsi keyboard mapping, this option starts **Vim** in Farsi mode, i.e. 'fkmap' and 'rightleft' are set. Otherwise an error message is given and **Vim** aborts.

−g If **Vim** has been compiled with GUI support, this option enables the GUI. If no GUI support was compiled in, an error message is given and **Vim** aborts.

−h Give a bit of help about the command line arguments and options. After this **Vim** exits.

−H If **Vim** has been compiled with RIGHTLEFT support for editing right-to-left oriented files and Hebrew keyboard mapping, this option starts **Vim** in Hebrew mode, i.e. 'hkmap' and 'rightleft' are set. Otherwise an error message is given and **Vim** aborts.

−i {viminfo} When using the viminfo file is enabled, this option sets the filename to use, instead of the default "~/.viminfo". This can also be used to skip the use of the .viminfo file, by giving the name "NONE".

−L Same as −r.

−l Lisp mode. Sets the 'lisp' and 'showmatch' options on.

−m Modifying files is disabled. Resets the 'write' option. You can still modify the buffer, but writing a file is not possible.

−M Modifications not allowed. The 'modifiable' and 'write' options will be unset, so that changes are not allowed and files can not be written. Note that these options can be set to enable making modifications.

−N No-compatible mode. Reset the 'compatible' option. This will make **Vim** behave a bit better, but less Vi compatible, even though a .vimrc file does not exist.

−n No swap file will be used. Recovery after a crash will be impossible. Handy if you want to edit a file on a very slow medium (e.g. floppy). Can also be done with ":set uc=0". Can be undone with ":set uc=200".

−nb Become an editor server for NetBeans. See the docs for details.

−o[N] Open N windows stacked. When N is omitted, open one window for each file.

−O[N] Open N windows side by side. When N is omitted, open one window for each file.

−p[N] Open N tab pages. When N is omitted, open one tab page for each file.

−R Read-only mode. The 'readonly' option will be set. You can still edit the buffer, but will be prevented from accidentally overwriting a file. If you do want to overwrite a file, add an exclamation mark to the Ex command, as in ":w!". The −R option also implies the −n option (see below). The 'readonly' option can be reset with ":set noro". See ":help 'readonly'".

−r List swap files, with information about using them for recovery.

−r {file} Recovery mode. The swap file is used to recover a crashed editing session. The swap file is a file with the same filename as the text file with ".swp" appended. See ":help recovery".

−s Silent mode. Only when started as "Ex" or when the "−e" option was given before the "−s" option.

−s {scriptin} The script file {scriptin} is read. The characters in the file are interpreted as if you had typed them. The same can be done with the command ":source! {scriptin}". If the end of the file is reached before the editor exits, further characters are read from the keyboard.

−T {terminal}

Tells **Vim** the name of the terminal you are using. Only required when the automatic way doesn't work. Should be a terminal known to **Vim** (builtin) or defined in the termcap or terminfo file.

−u {vimrc} Use the commands in the file {vimrc} for initializations. All the other initializations are skipped. Use this to edit a special kind of files. It can also be used to skip all initializations by giving the name "NONE". See ":help initialization" within vim for more details.

−U {gvimrc} Use the commands in the file {gvimrc} for GUI initializations. All the other GUI initializations are skipped. It can also be used to skip all GUI initializations by giving the name "NONE". See ":help gui−init" within vim for more details.

−V[N] Verbose. Give messages about which files are sourced and for reading and writing a viminfo file. The optional number N is the value for 'verbose'. Default is 10.

−v Start **Vim** in Vi mode, just like the executable was called "vi". This only has effect when the executable is called "ex".

−w {scriptout}
All the characters that you type are recorded in the file {scriptout}, until you exit **Vim.** This is useful if you want to create a script file to be used with "vim −s" or ":source!". If the {scriptout} file exists, characters are appended.

−W {scriptout}
Like −w, but an existing file is overwritten.

−x Use encryption when writing files. Will prompt for a crypt key.

−X Don't connect to the X server. Shortens startup time in a terminal, but the window title and clipboard will not be used.

−y Start **Vim** in easy mode, just like the executable was called "evim" or "eview". Makes **Vim** behave like a click-and-type editor.

−Z Restricted mode. Works like the executable starts with "r".

−− Denotes the end of the options. Arguments after this will be handled as a file name. This can be used to edit a filename that starts with a '−'.

−−echo−wid GTK GUI only: Echo the Window ID on stdout.

−−help Give a help message and exit, just like "−h".

−−literal Take file name arguments literally, do not expand wildcards. This has no effect on Unix where the shell expands wildcards.

−−noplugin Skip loading plugins. Implied by −u NONE.

−−remote Connect to a Vim server and make it edit the files given in the rest of the arguments. If no server is found a warning is given and the files are edited in the current Vim.

−−remote−expr {expr}
Connect to a Vim server, evaluate {expr} in it and print the result on stdout.

−−remote−send {keys}
Connect to a Vim server and send {keys} to it.

−−remote−silent
As −−remote, but without the warning when no server is found.

−−remote−wait
As −−remote, but Vim does not exit until the files have been edited.

−−remote−wait−silent
As −−remote−wait, but without the warning when no server is found.

−−serverlist List the names of all Vim servers that can be found.

−−servername {name}
Use {name} as the server name. Used for the current Vim, unless used with a −−remote argument, then it's the name of the server to connect to.

−−socketid {id}
GTK GUI only: Use the GtkPlug mechanism to run gvim in another window.

−−version Print version information and exit.

ON-LINE HELP

Type ":help" in **Vim** to get started. Type ":help subject" to get help on a specific subject. For example: ":help ZZ" to get help for the "ZZ" command. Use <Tab> and CTRL-D to complete subjects (":help cmd-line−completion"). Tags are present to jump from one place to another (sort of hypertext links, see ":help"). All documentation files can be viewed in this way, for example ":help syntax.txt".

FILES

/usr/share/vim/vim80/doc/*.txt
The **Vim** documentation files. Use ":help doc−file−list" to get the complete list.

/usr/share/vim/vim80/doc/tags
The tags file used for finding information in the documentation files.

/usr/share/vim/vim80/syntax/syntax.vim
System wide syntax initializations.

/usr/share/vim/vim80/syntax/*.vim
Syntax files for various languages.

/usr/share/vim/vimrc
System wide **Vim** initializations.

~/.vimrc Your personal **Vim** initializations.

/usr/share/vim/gvimrc
System wide gvim initializations.

~/.gvimrc Your personal gvim initializations.

/usr/share/vim/vim80/optwin.vim
Script used for the ":options" command, a nice way to view and set options.

/usr/share/vim/vim80/menu.vim
System wide menu initializations for gvim.

/usr/share/vim/vim80/bugreport.vim
Script to generate a bug report. See ":help bugs".

/usr/share/vim/vim80/filetype.vim
Script to detect the type of a file by its name. See ":help 'filetype'".

/usr/share/vim/vim80/scripts.vim
Script to detect the type of a file by its contents. See ":help 'filetype'".

/usr/share/vim/vim80/print/*.ps
Files used for PostScript printing.

For recent info read the VIM home page:
<URL:http://www.vim.org/>

SEE ALSO

vimtutor(1)

AUTHOR

Most of **Vim** was made by Bram Moolenaar, with a lot of help from others. See ":help credits" in **Vim.**
Vim is based on Stevie, worked on by: Tim Thompson, Tony Andrews and G.R. (Fred) Walter. Although hardly any of the original code remains.

BUGS

Probably. See ":help todo" for a list of known problems.

Note that a number of things that may be regarded as bugs by some, are in fact caused by a too-faithful

reproduction of Vi’s behaviour. And if you think other things are bugs "because Vi does it differently", you should take a closer look at the vi_diff.txt file (or type :help vi_diff.txt when in Vim). Also have a look at the ’compatible’ and ’cpoptions’ options.

NAME

Wget − The non−interactive network downloader.

SYNOPSIS

wget [*option*]... [*URL*]...

DESCRIPTION

GNU Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.

Wget is non-interactive, meaning that it can work in the background, while the user is not logged on. This allows you to start a retrieval and disconnect from the system, letting Wget finish the work. By contrast, most of the Web browsers require constant user's presence, which can be a great hindrance when transferring a lot of data.

Wget can follow links in HTML, XHTML, and CSS pages, to create local versions of remote web sites, fully recreating the directory structure of the original site. This is sometimes referred to as ''recursive downloading.'' While doing that, Wget respects the Robot Exclusion Standard (*/robots.txt*). Wget can be instructed to convert the links in downloaded files to point at the local files, for offline viewing.

Wget has been designed for robustness over slow or unstable network connections; if a download fails due to a network problem, it will keep retrying until the whole file has been retrieved. If the server supports regetting, it will instruct the server to continue the download from where it left off.

Wget does not support Client Revocation Lists (CRLs) so the HTTPS certificate you are connecting to might be revoked by the siteowner.

OPTIONS

Option Syntax

Since Wget uses GNU getopt to process command-line arguments, every option has a long form along with the short one. Long options are more convenient to remember, but take time to type. You may freely mix different option styles, or specify options after the command-line arguments. Thus you may write:

```
wget −r −−tries=10 http://fly.srk.fer.hr/ −o log
```

The space between the option accepting an argument and the argument may be omitted. Instead of **−o log** you can write **−olog**.

You may put several options that do not require arguments together, like:

```
wget −drc <URL>
```

This is completely equivalent to:

```
wget −d −r −c <URL>
```

Since the options can be specified after the arguments, you may terminate them with **−−**. So the following will try to download URL **−x**, reporting failure to *log*:

```
wget −o log −− −x
```

The options that accept comma-separated lists all respect the convention that specifying an empty list clears its value. This can be useful to clear the *.wgetrc* settings. For instance, if your *.wgetrc* sets `exclude_directories` to */cgi−bin*, the following example will first reset it, and then set it to exclude */˜nobody* and */˜somebody*. You can also clear the lists in *.wgetrc*.

```
wget −X " −X /˜nobody,/˜somebody
```

Most options that do not accept arguments are *boolean* options, so named because their state can be captured with a yes-or-no (''boolean'') variable. For example, **−−follow−ftp** tells Wget to follow FTP links from HTML files and, on the other hand, **−−no−glob** tells it not to perform file globbing on FTP URLs. A boolean option is either *affirmative* or *negative* (beginning with **−−no**). All such options share several properties.

Unless stated otherwise, it is assumed that the default behavior is the opposite of what the option accomplishes. For example, the documented existence of **−−follow−ftp** assumes that the default is to *not*

follow FTP links from HTML pages.

Affirmative options can be negated by prepending the **--no-** to the option name; negative options can be negated by omitting the **--no-** prefix. This might seem superfluous---if the default for an affirmative option is to not do something, then why provide a way to explicitly turn it off? But the startup file may in fact change the default. For instance, using `follow_ftp = on` in *.wgetrc* makes Wget *follow* FTP links by default, and using **--no-follow-ftp** is the only way to restore the factory default from the command line.

Basic Startup Options

-V
--version
Display the version of Wget.

-h
--help
Print a help message describing all of Wget's command-line options.

-b
--background
Go to background immediately after startup. If no output file is specified via the **-o**, output is redirected to *wget-log*.

-e *command*
--execute *command*
Execute *command* as if it were a part of *.wgetrc*. A command thus invoked will be executed *after* the commands in *.wgetrc*, thus taking precedence over them. If you need to specify more than one wgetrc command, use multiple instances of **-e**.

Logging and Input File Options

-o *logfile*
--output-file=*logfile*
Log all messages to *logfile*. The messages are normally reported to standard error.

-a *logfile*
--append-output=*logfile*
Append to *logfile*. This is the same as **-o**, only it appends to *logfile* instead of overwriting the old log file. If *logfile* does not exist, a new file is created.

-d
--debug
Turn on debug output, meaning various information important to the developers of Wget if it does not work properly. Your system administrator may have chosen to compile Wget without debug support, in which case **-d** will not work. Please note that compiling with debug support is always safe---Wget compiled with the debug support will *not* print any debug info unless requested with **-d**.

-q
--quiet
Turn off Wget's output.

-v
--verbose
Turn on verbose output, with all the available data. The default output is verbose.

-nv
--no-verbose
Turn off verbose without being completely quiet (use **-q** for that), which means that error messages and basic information still get printed.

--report-speed=*type*
Output bandwidth as *type*. The only accepted value is **bits**.

−i *file*
−−input−file=*file*
Read URLs from a local or external *file*. If − is specified as *file*, URLs are read from the standard input. (Use **./−** to read from a file literally named −.)

If this function is used, no URLs need be present on the command line. If there are URLs both on the command line and in an input file, those on the command lines will be the first ones to be retrieved. If **−−force−html** is not specified, then *file* should consist of a series of URLs, one per line.

However, if you specify **−−force−html**, the document will be regarded as **html**. In that case you may have problems with relative links, which you can solve either by adding `<base href="url">` to the documents or by specifying **−−base=***url* on the command line.

If the *file* is an external one, the document will be automatically treated as **html** if the Content-Type matches **text/html**. Furthermore, the *file*'s location will be implicitly used as base href if none was specified.

−−input−metalink=*file*
Downloads files covered in local Metalink *file*. Metalink version 3 and 4 are supported.

−−metalink−over−http
Issues HTTP HEAD request instead of GET and extracts Metalink metadata from response headers. Then it switches to Metalink download. If no valid Metalink metadata is found, it falls back to ordinary HTTP download.

−−preferred−location
Set preferred location for Metalink resources. This has effect if multiple resources with same priority are available.

−F
−−force−html
When input is read from a file, force it to be treated as an HTML file. This enables you to retrieve relative links from existing HTML files on your local disk, by adding `<base href="url">` to HTML, or using the **−−base** command-line option.

−B *URL*
−−base=*URL*
Resolves relative links using *URL* as the point of reference, when reading links from an HTML file specified via the **−i**/**−−input−file** option (together with **−−force−html**, or when the input file was fetched remotely from a server describing it as HTML). This is equivalent to the presence of a `BASE` tag in the HTML input file, with *URL* as the value for the `href` attribute.

For instance, if you specify **http://foo/bar/a.html** for *URL*, and Wget reads **../baz/b.html** from the input file, it would be resolved to **http://foo/baz/b.html**.

−−config=*FILE*
Specify the location of a startup file you wish to use.

−−rejected−log=*logfile*
Logs all URL rejections to *logfile* as comma separated values. The values include the reason of rejection, the URL and the parent URL it was found in.

Download Options
−−bind−address=*ADDRESS*
When making client TCP/IP connections, bind to *ADDRESS* on the local machine. *ADDRESS* may be specified as a hostname or IP address. This option can be useful if your machine is bound to multiple IPs.

−−bind−dns−address=*ADDRESS*
[libcares only] This address overrides the route for DNS requests. If you ever need to circumvent the standard settings from /etc/resolv.conf, this option together with **−−dns−servers** is your friend. *ADDRESS* must be specified either as IPv4 or IPv6 address. Wget needs to be built with libcares for

this option to be available.

−−dns−servers=*ADDRESSES*
[libcares only] The given address(es) override the standard nameserver addresses, e.g. as configured in /etc/resolv.conf. *ADDRESSES* may be specified either as IPv4 or IPv6 addresses, comma-separated. Wget needs to be built with libcares for this option to be available.

−t *number*
−−tries=*number*
Set number of tries to *number*. Specify 0 or **inf** for infinite retrying. The default is to retry 20 times, with the exception of fatal errors like ‘‘connection refused’’ or ‘‘not found’’ (404), which are not retried.

−O *file*
−−output−document=*file*
The documents will not be written to the appropriate files, but all will be concatenated together and written to *file*. If − is used as *file*, documents will be printed to standard output, disabling link conversion. (Use **./−** to print to a file literally named −.)

Use of **−O** is *not* intended to mean simply "use the name *file* instead of the one in the URL;" rather, it is analogous to shell redirection: **wget −O file http://foo** is intended to work like **wget −O − http://foo > file**; *file* will be truncated immediately, and *all* downloaded content will be written there.

For this reason, **−N** (for timestamp-checking) is not supported in combination with **−O**: since *file* is always newly created, it will always have a very new timestamp. A warning will be issued if this combination is used.

Similarly, using **−r** or **−p** with **−O** may not work as you expect: Wget won’t just download the first file to *file* and then download the rest to their normal names: *all* downloaded content will be placed in *file*. This was disabled in version 1.11, but has been reinstated (with a warning) in 1.11.2, as there are some cases where this behavior can actually have some use.

A combination with **−nc** is only accepted if the given output file does not exist.

Note that a combination with **−k** is only permitted when downloading a single document, as in that case it will just convert all relative URIs to external ones; **−k** makes no sense for multiple URIs when they’re all being downloaded to a single file; **−k** can be used only when the output is a regular file.

−nc
−−no−clobber
If a file is downloaded more than once in the same directory, Wget’s behavior depends on a few options, including **−nc**. In certain cases, the local file will be *clobbered*, or overwritten, upon repeated download. In other cases it will be preserved.

When running Wget without **−N**, **−nc**, **−r**, or **−p**, downloading the same file in the same directory will result in the original copy of *file* being preserved and the second copy being named *file***.1**. If that file is downloaded yet again, the third copy will be named *file***.2**, and so on. (This is also the behavior with **−nd**, even if **−r** or **−p** are in effect.) When **−nc** is specified, this behavior is suppressed, and Wget will refuse to download newer copies of *file*. Therefore, "`no-clobber`" is actually a misnomer in this mode−−−it’s not clobbering that’s prevented (as the numeric suffixes were already preventing clobbering), but rather the multiple version saving that’s prevented.

When running Wget with **−r** or **−p**, but without **−N**, **−nd**, or **−nc**, re-downloading a file will result in the new copy simply overwriting the old. Adding **−nc** will prevent this behavior, instead causing the original version to be preserved and any newer copies on the server to be ignored.

When running Wget with **−N**, with or without **−r** or **−p**, the decision as to whether or not to download a newer copy of a file depends on the local and remote timestamp and size of the file. **−nc** may not be specified at the same time as **−N**.

A combination with **−O**/**−−output−document** is only accepted if the given output file does not exist.

Note that when **−nc** is specified, files with the suffixes **.html** or **.htm** will be loaded from the local disk and parsed as if they had been retrieved from the Web.

−−backups=*backups*
Before (over)writing a file, back up an existing file by adding a **.1** suffix (**_1** on VMS) to the file name. Such backup files are rotated to **.2**, **.3**, and so on, up to *backups* (and lost beyond that).

−c
−−continue
Continue getting a partially-downloaded file. This is useful when you want to finish up a download started by a previous instance of Wget, or by another program. For instance:

```
wget −c ftp://sunsite.doc.ic.ac.uk/ls−lR.Z
```

If there is a file named *ls−lR.Z* in the current directory, Wget will assume that it is the first portion of the remote file, and will ask the server to continue the retrieval from an offset equal to the length of the local file.

Note that you don't need to specify this option if you just want the current invocation of Wget to retry downloading a file should the connection be lost midway through. This is the default behavior. **−c** only affects resumption of downloads started *prior* to this invocation of Wget, and whose local files are still sitting around.

Without **−c**, the previous example would just download the remote file to *ls−lR.Z.1*, leaving the truncated *ls−lR.Z* file alone.

If you use **−c** on a non-empty file, and the server does not support continued downloading, Wget will restart the download from scratch and overwrite the existing file entirely.

Beginning with Wget 1.7, if you use **−c** on a file which is of equal size as the one on the server, Wget will refuse to download the file and print an explanatory message. The same happens when the file is smaller on the server than locally (presumably because it was changed on the server since your last download attempt)−−−because ''continuing'' is not meaningful, no download occurs.

On the other side of the coin, while using **−c**, any file that's bigger on the server than locally will be considered an incomplete download and only `(length(remote) − length(local))` bytes will be downloaded and tacked onto the end of the local file. This behavior can be desirable in certain cases−−−for instance, you can use **wget −c** to download just the new portion that's been appended to a data collection or log file.

However, if the file is bigger on the server because it's been *changed*, as opposed to just *appended* to, you'll end up with a garbled file. Wget has no way of verifying that the local file is really a valid prefix of the remote file. You need to be especially careful of this when using **−c** in conjunction with **−r**, since every file will be considered as an ''incomplete download'' candidate.

Another instance where you'll get a garbled file if you try to use **−c** is if you have a lame HTTP proxy that inserts a ''transfer interrupted'' string into the local file. In the future a ''rollback'' option may be added to deal with this case.

Note that **−c** only works with FTP servers and with HTTP servers that support the `Range` header.

−−start−pos=*OFFSET*
Start downloading at zero-based position *OFFSET*. Offset may be expressed in bytes, kilobytes with the 'k' suffix, or megabytes with the 'm' suffix, etc.

−−start−pos has higher precedence over **−−continue**. When **−−start−pos** and **−−continue** are both specified, wget will emit a warning then proceed as if **−−continue** was absent.

Server support for continued download is required, otherwise **−−start−pos** cannot help. See **−c** for details.

−−progress=*type*
Select the type of the progress indicator you wish to use. Legal indicators are ‘‘dot’’ and ‘‘bar’’.

The ‘‘bar’’ indicator is used by default. It draws an ASCII progress bar graphics (a.k.a ‘‘thermometer’’ display) indicating the status of retrieval. If the output is not a TTY, the ‘‘dot’’ bar will be used by default.

Use **−−progress=dot** to switch to the ‘‘dot’’ display. It traces the retrieval by printing dots on the screen, each dot representing a fixed amount of downloaded data.

The progress *type* can also take one or more parameters. The parameters vary based on the *type* selected. Parameters to *type* are passed by appending them to the type sperated by a colon (:) like this: **−−progress=***type***:***parameter1***:***parameter2*.

When using the dotted retrieval, you may set the *style* by specifying the type as **dot:***style*. Different styles assign different meaning to one dot. With the `default` style each dot represents 1K, there are ten dots in a cluster and 50 dots in a line. The `binary` style has a more ‘‘computer’’−like orientation−−−8K dots, 16−dots clusters and 48 dots per line (which makes for 384K lines). The `mega` style is suitable for downloading large files−−−each dot represents 64K retrieved, there are eight dots in a cluster, and 48 dots on each line (so each line contains 3M). If `mega` is not enough then you can use the `giga` style−−−each dot represents 1M retrieved, there are eight dots in a cluster, and 32 dots on each line (so each line contains 32M).

With **−−progress=bar**, there are currently two possible parameters, *force* and *noscroll*.

When the output is not a TTY, the progress bar always falls back to ‘‘dot’’, even if **−−progress=bar** was passed to Wget during invokation. This behaviour can be overridden and the ‘‘bar’’ output forced by using the ‘‘force’’ parameter as **−−progress=bar:force**.

By default, the **bar** style progress bar scroll the name of the file from left to right for the file being downloaded if the filename exceeds the maximum length allotted for its display. In certain cases, such as with **−−progress=bar:force**, one may not want the scrolling filename in the progress bar. By passing the ‘‘noscroll’’ parameter, Wget can be forced to display as much of the filename as possible without scrolling through it.

Note that you can set the default style using the `progress` command in *.wgetrc*. That setting may be overridden from the command line. For example, to force the bar output without scrolling, use **−−progress=bar:force:noscroll**.

−−show−progress
Force wget to display the progress bar in any verbosity.

By default, wget only displays the progress bar in verbose mode. One may however, want wget to display the progress bar on screen in conjunction with any other verbosity modes like **−−no−verbose** or **−−quiet**. This is often a desired a property when invoking wget to download several small/large files. In such a case, wget could simply be invoked with this parameter to get a much cleaner output on the screen.

This option will also force the progress bar to be printed to *stderr* when used alongside the **−−logfile** option.

−N
−−timestamping
Turn on time-stamping.

−−no−if−modified−since
Do not send If-Modified-Since header in **−N** mode. Send preliminary HEAD request instead. This has only effect in **−N** mode.

−−no−use−server−timestamps
Don’t set the local file’s timestamp by the one on the server.

By default, when a file is downloaded, its timestamps are set to match those from the remote file. This allows the use of **--timestamping** on subsequent invocations of wget. However, it is sometimes useful to base the local file's timestamp on when it was actually downloaded; for that purpose, the **--no-use-server-timestamps** option has been provided.

-S
--server-response
Print the headers sent by HTTP servers and responses sent by FTP servers.

--spider
When invoked with this option, Wget will behave as a Web *spider*, which means that it will not download the pages, just check that they are there. For example, you can use Wget to check your bookmarks:

```
wget --spider --force-html -i bookmarks.html
```

This feature needs much more work for Wget to get close to the functionality of real web spiders.

-T seconds
--timeout=*seconds*
Set the network timeout to *seconds* seconds. This is equivalent to specifying **--dns-timeout**, **--connect-timeout**, and **--read-timeout**, all at the same time.

When interacting with the network, Wget can check for timeout and abort the operation if it takes too long. This prevents anomalies like hanging reads and infinite connects. The only timeout enabled by default is a 900-second read timeout. Setting a timeout to 0 disables it altogether. Unless you know what you are doing, it is best not to change the default timeout settings.

All timeout-related options accept decimal values, as well as subsecond values. For example, **0.1** seconds is a legal (though unwise) choice of timeout. Subsecond timeouts are useful for checking server response times or for testing network latency.

--dns-timeout=*seconds*
Set the DNS lookup timeout to *seconds* seconds. DNS lookups that don't complete within the specified time will fail. By default, there is no timeout on DNS lookups, other than that implemented by system libraries.

--connect-timeout=*seconds*
Set the connect timeout to *seconds* seconds. TCP connections that take longer to establish will be aborted. By default, there is no connect timeout, other than that implemented by system libraries.

--read-timeout=*seconds*
Set the read (and write) timeout to *seconds* seconds. The ''time'' of this timeout refers to *idle time*: if, at any point in the download, no data is received for more than the specified number of seconds, reading fails and the download is restarted. This option does not directly affect the duration of the entire download.

Of course, the remote server may choose to terminate the connection sooner than this option requires. The default read timeout is 900 seconds.

--limit-rate=*amount*
Limit the download speed to *amount* bytes per second. Amount may be expressed in bytes, kilobytes with the **k** suffix, or megabytes with the **m** suffix. For example, **--limit-rate=20k** will limit the retrieval rate to 20KB/s. This is useful when, for whatever reason, you don't want Wget to consume the entire available bandwidth.

This option allows the use of decimal numbers, usually in conjunction with power suffixes; for example, **--limit-rate=2.5k** is a legal value.

Note that Wget implements the limiting by sleeping the appropriate amount of time after a network read that took less time than specified by the rate. Eventually this strategy causes the TCP transfer to slow down to approximately the specified rate. However, it may take some time for this balance to be

achieved, so don't be surprised if limiting the rate doesn't work well with very small files.

−w *seconds*
−−wait=*seconds*
Wait the specified number of seconds between the retrievals. Use of this option is recommended, as it lightens the server load by making the requests less frequent. Instead of in seconds, the time can be specified in minutes using the `m` suffix, in hours using `h` suffix, or in days using `d` suffix.

Specifying a large value for this option is useful if the network or the destination host is down, so that Wget can wait long enough to reasonably expect the network error to be fixed before the retry. The waiting interval specified by this function is influenced by `−−random−wait`, which see.

−−waitretry=*seconds*
If you don't want Wget to wait between *every* retrieval, but only between retries of failed downloads, you can use this option. Wget will use *linear backoff*, waiting 1 second after the first failure on a given file, then waiting 2 seconds after the second failure on that file, up to the maximum number of *seconds* you specify.

By default, Wget will assume a value of 10 seconds.

−−random−wait
Some web sites may perform log analysis to identify retrieval programs such as Wget by looking for statistically significant similarities in the time between requests. This option causes the time between requests to vary between 0.5 and 1.5 * *wait* seconds, where *wait* was specified using the **−−wait** option, in order to mask Wget's presence from such analysis.

A 2001 article in a publication devoted to development on a popular consumer platform provided code to perform this analysis on the fly. Its author suggested blocking at the class C address level to ensure automated retrieval programs were blocked despite changing DHCP-supplied addresses.

The **−−random−wait** option was inspired by this ill-advised recommendation to block many unrelated users from a web site due to the actions of one.

−−no−proxy
Don't use proxies, even if the appropriate `*_proxy` environment variable is defined.

−Q *quota*
−−quota=*quota*
Specify download quota for automatic retrievals. The value can be specified in bytes (default), kilobytes (with **k** suffix), or megabytes (with **m** suffix).

Note that quota will never affect downloading a single file. So if you specify **wget −Q10k https://example.com/ls−lR.gz**, all of the *ls−lR.gz* will be downloaded. The same goes even when several URLs are specified on the command-line. However, quota is respected when retrieving either recursively, or from an input file. Thus you may safely type **wget −Q2m −i sites**−−−download will be aborted when the quota is exceeded.

Setting quota to 0 or to **inf** unlimits the download quota.

−−no−dns−cache
Turn off caching of DNS lookups. Normally, Wget remembers the IP addresses it looked up from DNS so it doesn't have to repeatedly contact the DNS server for the same (typically small) set of hosts it retrieves from. This cache exists in memory only; a new Wget run will contact DNS again.

However, it has been reported that in some situations it is not desirable to cache host names, even for the duration of a short-running application like Wget. With this option Wget issues a new DNS lookup (more precisely, a new call to `gethostbyname` or `getaddrinfo`) each time it makes a new connection. Please note that this option will *not* affect caching that might be performed by the resolving library or by an external caching layer, such as NSCD.

If you don't understand exactly what this option does, you probably won't need it.

−−restrict−file−names=*modes*

Change which characters found in remote URLs must be escaped during generation of local filenames. Characters that are *restricted* by this option are escaped, i.e. replaced with **%HH**, where **HH** is the hexadecimal number that corresponds to the restricted character. This option may also be used to force all alphabetical cases to be either lower− or uppercase.

By default, Wget escapes the characters that are not valid or safe as part of file names on your operating system, as well as control characters that are typically unprintable. This option is useful for changing these defaults, perhaps because you are downloading to a non-native partition, or because you want to disable escaping of the control characters, or you want to further restrict characters to only those in the ASCII range of values.

The *modes* are a comma-separated set of text values. The acceptable values are **unix**, **windows**, **nocontrol**, **ascii**, **lowercase**, and **uppercase**. The values **unix** and **windows** are mutually exclusive (one will override the other), as are **lowercase** and **uppercase**. Those last are special cases, as they do not change the set of characters that would be escaped, but rather force local file paths to be converted either to lower− or uppercase.

When ‘‘unix’’ is specified, Wget escapes the character / and the control characters in the ranges 0−−31 and 128−−159. This is the default on Unix-like operating systems.

When ‘‘windows’’ is given, Wget escapes the characters \, |, /, **:**, **?**, ", *****, <, >, and the control characters in the ranges 0−−31 and 128−−159. In addition to this, Wget in Windows mode uses **+** instead of **:** to separate host and port in local file names, and uses **@** instead of **?** to separate the query portion of the file name from the rest. Therefore, a URL that would be saved as **www.xemacs.org:4300/search.pl?input=blah** in Unix mode would be saved as **www.xemacs.org+4300/search.pl@input=blah** in Windows mode. This mode is the default on Windows.

If you specify **nocontrol**, then the escaping of the control characters is also switched off. This option may make sense when you are downloading URLs whose names contain UTF−8 characters, on a system which can save and display filenames in UTF−8 (some possible byte values used in UTF−8 byte sequences fall in the range of values designated by Wget as ‘‘controls’’).

The **ascii** mode is used to specify that any bytes whose values are outside the range of ASCII characters (that is, greater than 127) shall be escaped. This can be useful when saving filenames whose encoding does not match the one used locally.

−4
−−inet4−only
−6
−−inet6−only

Force connecting to IPv4 or IPv6 addresses. With **−−inet4−only** or **−4**, Wget will only connect to IPv4 hosts, ignoring AAAA records in DNS, and refusing to connect to IPv6 addresses specified in URLs. Conversely, with **−−inet6−only** or **−6**, Wget will only connect to IPv6 hosts and ignore A records and IPv4 addresses.

Neither options should be needed normally. By default, an IPv6−aware Wget will use the address family specified by the host’s DNS record. If the DNS responds with both IPv4 and IPv6 addresses, Wget will try them in sequence until it finds one it can connect to. (Also see `−−prefer−family` option described below.)

These options can be used to deliberately force the use of IPv4 or IPv6 address families on dual family systems, usually to aid debugging or to deal with broken network configuration. Only one of **−−inet6−only** and **−−inet4−only** may be specified at the same time. Neither option is available in Wget compiled without IPv6 support.

−−prefer−family=none/IPv4/IPv6

When given a choice of several addresses, connect to the addresses with specified address family first. The address order returned by DNS is used without change by default.

This avoids spurious errors and connect attempts when accessing hosts that resolve to both IPv6 and IPv4 addresses from IPv4 networks. For example, **www.kame.net** resolves to **2001:200:0:8002:203:47ff:fea5:3085** and to **203.178.141.194**. When the preferred family is `IPv4`, the IPv4 address is used first; when the preferred family is `IPv6`, the IPv6 address is used first; if the specified value is `none`, the address order returned by DNS is used without change.

Unlike **−4** and **−6**, this option doesn't inhibit access to any address family, it only changes the *order* in which the addresses are accessed. Also note that the reordering performed by this option is *stable*−−−it doesn't affect order of addresses of the same family. That is, the relative order of all IPv4 addresses and of all IPv6 addresses remains intact in all cases.

−−retry−connrefused
Consider ''connection refused'' a transient error and try again. Normally Wget gives up on a URL when it is unable to connect to the site because failure to connect is taken as a sign that the server is not running at all and that retries would not help. This option is for mirroring unreliable sites whose servers tend to disappear for short periods of time.

−−user=*user*
−−password=*password*
Specify the username *user* and password *password* for both FTP and HTTP file retrieval. These parameters can be overridden using the **−−ftp−user** and **−−ftp−password** options for FTP connections and the **−−http−user** and **−−http−password** options for HTTP connections.

−−ask−password
Prompt for a password for each connection established. Cannot be specified when **−−password** is being used, because they are mutually exclusive.

−−no−iri
Turn off internationalized URI (IRI) support. Use **−−iri** to turn it on. IRI support is activated by default.

You can set the default state of IRI support using the `iri` command in *.wgetrc*. That setting may be overridden from the command line.

−−local−encoding=*encoding*
Force Wget to use *encoding* as the default system encoding. That affects how Wget converts URLs specified as arguments from locale to UTF−8 for IRI support.

Wget use the function `nl_langinfo()` and then the `CHARSET` environment variable to get the locale. If it fails, ASCII is used.

You can set the default local encoding using the `local_encoding` command in *.wgetrc*. That setting may be overridden from the command line.

−−remote−encoding=*encoding*
Force Wget to use *encoding* as the default remote server encoding. That affects how Wget converts URIs found in files from remote encoding to UTF−8 during a recursive fetch. This options is only useful for IRI support, for the interpretation of non-ASCII characters.

For HTTP, remote encoding can be found in HTTP `Content−Type` header and in HTML `Content−Type http−equiv` meta tag.

You can set the default encoding using the `remoteencoding` command in *.wgetrc*. That setting may be overridden from the command line.

−−unlink
Force Wget to unlink file instead of clobbering existing file. This option is useful for downloading to the directory with hardlinks.

Directory Options
−nd

−−no−directories
Do not create a hierarchy of directories when retrieving recursively. With this option turned on, all files will get saved to the current directory, without clobbering (if a name shows up more than once, the filenames will get extensions **.n**).

−x
−−force−directories
The opposite of **−nd**−−−create a hierarchy of directories, even if one would not have been created otherwise. E.g. **wget −x http://fly.srk.fer.hr/robots.txt** will save the downloaded file to *fly.srk.fer.hr/robots.txt*.

−nH
−−no−host−directories
Disable generation of host-prefixed directories. By default, invoking Wget with **−r http://fly.srk.fer.hr/** will create a structure of directories beginning with *fly.srk.fer.hr/*. This option disables such behavior.

−−protocol−directories
Use the protocol name as a directory component of local file names. For example, with this option, **wget −r http://***host* will save to **http/***host***/...** rather than just to *host***/...**.

−−cut−dirs=*number*
Ignore *number* directory components. This is useful for getting a fine-grained control over the directory where recursive retrieval will be saved.

Take, for example, the directory at **ftp://ftp.xemacs.org/pub/xemacs/**. If you retrieve it with **−r**, it will be saved locally under *ftp.xemacs.org/pub/xemacs/*. While the **−nH** option can remove the *ftp.xemacs.org/* part, you are still stuck with *pub/xemacs*. This is where **−−cut−dirs** comes in handy; it makes Wget not ‘‘see’’ *number* remote directory components. Here are several examples of how **−−cut−dirs** option works.

```
No options        -> ftp.xemacs.org/pub/xemacs/
-nH               -> pub/xemacs/
-nH --cut-dirs=1  -> xemacs/
-nH --cut-dirs=2  -> .

--cut-dirs=1      -> ftp.xemacs.org/xemacs/
...
```

If you just want to get rid of the directory structure, this option is similar to a combination of **−nd** and **−P**. However, unlike **−nd**, **−−cut−dirs** does not lose with subdirectories−−−for instance, with **−nH −−cut−dirs=1**, a *beta/* subdirectory will be placed to *xemacs/beta*, as one would expect.

−P *prefix*
−−directory−prefix=*prefix*
Set directory prefix to *prefix*. The *directory prefix* is the directory where all other files and subdirectories will be saved to, i.e. the top of the retrieval tree. The default is **.** (the current directory).

HTTP Options

−−default−page=*name*
Use *name* as the default file name when it isn’t known (i.e., for URLs that end in a slash), instead of *index.html*.

−E
−−adjust−extension
If a file of type **application/xhtml+xml** or **text/html** is downloaded and the URL does not end with the regexp **\.[Hh][Tt][Mm][Ll]?**, this option will cause the suffix **.html** to be appended to the local filename. This is useful, for instance, when you’re mirroring a remote site that uses **.asp** pages, but you want the mirrored pages to be viewable on your stock Apache server. Another good use for this is when you’re downloading CGI-generated materials. A URL like **http://site.com/article.cgi?25** will be

saved as *article.cgi?25.html*.

Note that filenames changed in this way will be re-downloaded every time you re-mirror a site, because Wget can't tell that the local *X.html* file corresponds to remote URL *X* (since it doesn't yet know that the URL produces output of type **text/html** or **application/xhtml+xml**.

As of version 1.12, Wget will also ensure that any downloaded files of type **text/css** end in the suffix **.css**, and the option was renamed from **--html-extension**, to better reflect its new behavior. The old option name is still acceptable, but should now be considered deprecated.

At some point in the future, this option may well be expanded to include suffixes for other types of content, including content types that are not parsed by Wget.

--http-user=*user*

--http-password=*password*

Specify the username *user* and password *password* on an HTTP server. According to the type of the challenge, Wget will encode them using either the `basic` (insecure), the `digest`, or the Windows `NTLM` authentication scheme.

Another way to specify username and password is in the URL itself. Either method reveals your password to anyone who bothers to run `ps`. To prevent the passwords from being seen, store them in *.wgetrc* or *.netrc*, and make sure to protect those files from other users with `chmod`. If the passwords are really important, do not leave them lying in those files either---edit the files and delete them after Wget has started the download.

--no-http-keep-alive

Turn off the ''keep-alive'' feature for HTTP downloads. Normally, Wget asks the server to keep the connection open so that, when you download more than one document from the same server, they get transferred over the same TCP connection. This saves time and at the same time reduces the load on the server.

This option is useful when, for some reason, persistent (keep-alive) connections don't work for you, for example due to a server bug or due to the inability of server-side scripts to cope with the connections.

--no-cache

Disable server-side cache. In this case, Wget will send the remote server an appropriate directive (**Pragma: no-cache**) to get the file from the remote service, rather than returning the cached version. This is especially useful for retrieving and flushing out-of-date documents on proxy servers.

Caching is allowed by default.

--no-cookies

Disable the use of cookies. Cookies are a mechanism for maintaining server-side state. The server sends the client a cookie using the `Set-Cookie` header, and the client responds with the same cookie upon further requests. Since cookies allow the server owners to keep track of visitors and for sites to exchange this information, some consider them a breach of privacy. The default is to use cookies; however, *storing* cookies is not on by default.

--load-cookies *file*

Load cookies from *file* before the first HTTP retrieval. *file* is a textual file in the format originally used by Netscape's *cookies.txt* file.

You will typically use this option when mirroring sites that require that you be logged in to access some or all of their content. The login process typically works by the web server issuing an HTTP cookie upon receiving and verifying your credentials. The cookie is then resent by the browser when accessing that part of the site, and so proves your identity.

Mirroring such a site requires Wget to send the same cookies your browser sends when communicating with the site. This is achieved by **--load-cookies**---simply point Wget to the location of the *cookies.txt* file, and it will send the same cookies your browser would send in the same

situation. Different browsers keep textual cookie files in different locations:

`Netscape 4.x.`
The cookies are in *~/.netscape/cookies.txt*.

`Mozilla and Netscape 6.x.`
Mozilla's cookie file is also named *cookies.txt*, located somewhere under *~/.mozilla*, in the directory of your profile. The full path usually ends up looking somewhat like *~/.mozilla/default/some-weird-string/cookies.txt*.

`Internet Explorer.`
You can produce a cookie file Wget can use by using the File menu, Import and Export, Export Cookies. This has been tested with Internet Explorer 5; it is not guaranteed to work with earlier versions.

`Other browsers.`
If you are using a different browser to create your cookies, **--load-cookies** will only work if you can locate or produce a cookie file in the Netscape format that Wget expects.

If you cannot use **--load-cookies**, there might still be an alternative. If your browser supports a ''cookie manager'', you can use it to view the cookies used when accessing the site you're mirroring. Write down the name and value of the cookie, and manually instruct Wget to send those cookies, bypassing the ''official'' cookie support:

```
wget --no-cookies --header "Cookie: <name>=<value>"
```

--save-cookies *file*
Save cookies to *file* before exiting. This will not save cookies that have expired or that have no expiry time (so-called ''session cookies''), but also see **--keep-session-cookies**.

--keep-session-cookies
When specified, causes **--save-cookies** to also save session cookies. Session cookies are normally not saved because they are meant to be kept in memory and forgotten when you exit the browser. Saving them is useful on sites that require you to log in or to visit the home page before you can access some pages. With this option, multiple Wget runs are considered a single browser session as far as the site is concerned.

Since the cookie file format does not normally carry session cookies, Wget marks them with an expiry timestamp of 0. Wget's **--load-cookies** recognizes those as session cookies, but it might confuse other browsers. Also note that cookies so loaded will be treated as other session cookies, which means that if you want **--save-cookies** to preserve them again, you must use **--keep-session-cookies** again.

--ignore-length
Unfortunately, some HTTP servers (CGI programs, to be more precise) send out bogus `Content-Length` headers, which makes Wget go wild, as it thinks not all the document was retrieved. You can spot this syndrome if Wget retries getting the same document again and again, each time claiming that the (otherwise normal) connection has closed on the very same byte.

With this option, Wget will ignore the `Content-Length` header---as if it never existed.

--header=*header-line*
Send *header-line* along with the rest of the headers in each HTTP request. The supplied header is sent as-is, which means it must contain name and value separated by colon, and must not contain newlines.

You may define more than one additional header by specifying **--header** more than once.

```
wget --header='Accept-Charset: iso-8859-2' \
     --header='Accept-Language: hr'        \
       http://fly.srk.fer.hr/
```

Specification of an empty string as the header value will clear all previous user-defined headers.

As of Wget 1.10, this option can be used to override headers otherwise generated automatically. This example instructs Wget to connect to localhost, but to specify **foo.bar** in the `Host` header:

```
wget --header="Host: foo.bar" http://localhost/
```

In versions of Wget prior to 1.10 such use of **--header** caused sending of duplicate headers.

--max-redirect=*number*
Specifies the maximum number of redirections to follow for a resource. The default is 20, which is usually far more than necessary. However, on those occasions where you want to allow more (or fewer), this is the option to use.

--proxy-user=*user*
--proxy-password=*password*
Specify the username *user* and password *password* for authentication on a proxy server. Wget will encode them using the `basic` authentication scheme.

Security considerations similar to those with **--http-password** pertain here as well.

--referer=*url*
Include 'Referer: *url*' header in HTTP request. Useful for retrieving documents with server-side processing that assume they are always being retrieved by interactive web browsers and only come out properly when Referer is set to one of the pages that point to them.

--save-headers
Save the headers sent by the HTTP server to the file, preceding the actual contents, with an empty line as the separator.

-U *agent-string*
--user-agent=*agent-string*
Identify as *agent-string* to the HTTP server.

The HTTP protocol allows the clients to identify themselves using a `User-Agent` header field. This enables distinguishing the WWW software, usually for statistical purposes or for tracing of protocol violations. Wget normally identifies as **Wget/***version*, *version* being the current version number of Wget.

However, some sites have been known to impose the policy of tailoring the output according to the `User-Agent`-supplied information. While this is not such a bad idea in theory, it has been abused by servers denying information to clients other than (historically) Netscape or, more frequently, Microsoft Internet Explorer. This option allows you to change the `User-Agent` line issued by Wget. Use of this option is discouraged, unless you really know what you are doing.

Specifying empty user agent with **--user-agent=""** instructs Wget not to send the `User-Agent` header in HTTP requests.

--post-data=*string*
--post-file=*file*
Use POST as the method for all HTTP requests and send the specified data in the request body. **--post-data** sends *string* as data, whereas **--post-file** sends the contents of *file*. Other than that, they work in exactly the same way. In particular, they *both* expect content of the form `key1=value1&key2=value2`, with percent-encoding for special characters; the only difference is that one expects its content as a command-line parameter and the other accepts its content from a file. In particular, **--post-file** is *not* for transmitting files as form attachments: those must appear as `key=value` data (with appropriate percent-coding) just like everything else. Wget does not currently support `multipart/form-data` for transmitting POST data; only `application/x-www-form-urlencoded`. Only one of **--post-data** and **--post-file** should be specified.

Please note that wget does not require the content to be of the form `key1=value1&key2=value2`, and neither does it test for it. Wget will simply transmit whatever data is provided to it. Most servers however expect the POST data to be in the above format when processing HTML Forms.

When sending a POST request using the **--post-file** option, Wget treats the file as a binary file and will send every character in the POST request without stripping trailing newline or formfeed characters. Any other control characters in the text will also be sent as-is in the POST request.

Please be aware that Wget needs to know the size of the POST data in advance. Therefore the argument to `--post-file` must be a regular file; specifying a FIFO or something like */dev/stdin* won't work. It's not quite clear how to work around this limitation inherent in HTTP/1.0. Although HTTP/1.1 introduces *chunked* transfer that doesn't require knowing the request length in advance, a client can't use chunked unless it knows it's talking to an HTTP/1.1 server. And it can't know that until it receives a response, which in turn requires the request to have been completed — a chicken-and-egg problem.

Note: As of version 1.15 if Wget is redirected after the POST request is completed, its behaviour will depend on the response code returned by the server. In case of a 301 Moved Permanently, 302 Moved Temporarily or 307 Temporary Redirect, Wget will, in accordance with RFC2616, continue to send a POST request. In case a server wants the client to change the Request method upon redirection, it should send a 303 See Other response code.

This example shows how to log in to a server using POST and then proceed to download the desired pages, presumably only accessible to authorized users:

```
# Log in to the server.  This can be done only once.
wget --save-cookies cookies.txt \
     --post-data 'user=foo&password=bar' \
     http://example.com/auth.php

# Now grab the page or pages we care about.
wget --load-cookies cookies.txt \
     -p http://example.com/interesting/article.php
```

If the server is using session cookies to track user authentication, the above will not work because **--save-cookies** will not save them (and neither will browsers) and the *cookies.txt* file will be empty. In that case use **--keep-session-cookies** along with **--save-cookies** to force saving of session cookies.

--method=*HTTP-Method*

For the purpose of RESTful scripting, Wget allows sending of other HTTP Methods without the need to explicitly set them using **--header=Header-Line**. Wget will use whatever string is passed to it after **--method** as the HTTP Method to the server.

--body-data=*Data-String*

--body-file=*Data-File*

Must be set when additional data needs to be sent to the server along with the Method specified using **--method**. **--body-data** sends *string* as data, whereas **--body-file** sends the contents of *file*. Other than that, they work in exactly the same way.

Currently, **--body-file** is *not* for transmitting files as a whole. Wget does not currently support `multipart/form-data` for transmitting data; only `application/x-www-form-urlencoded`. In the future, this may be changed so that wget sends the **--body-file** as a complete file instead of sending its contents to the server. Please be aware that Wget needs to know the contents of BODY Data in advance, and hence the argument to **--body-file** should be a regular file. See **--post-file** for a more detailed explanation. Only one of **--body-data** and **--body-file** should be specified.

If Wget is redirected after the request is completed, Wget will suspend the current method and send a GET request till the redirection is completed. This is true for all redirection response codes except 307 Temporary Redirect which is used to explicitly specify that the request method should *not* change. Another exception is when the method is set to `POST`, in which case the redirection rules specified under **--post-data** are followed.

−−content−disposition
If this is set to on, experimental (not fully-functional) support for `Content-Disposition` headers is enabled. This can currently result in extra round-trips to the server for a `HEAD` request, and is known to suffer from a few bugs, which is why it is not currently enabled by default.

This option is useful for some file-downloading CGI programs that use `Content-Disposition` headers to describe what the name of a downloaded file should be.

−−content−on−error
If this is set to on, wget will not skip the content when the server responds with a http status code that indicates error.

−−trust−server−names
If this is set to on, on a redirect the last component of the redirection URL will be used as the local file name. By default it is used the last component in the original URL.

−−auth−no−challenge
If this option is given, Wget will send Basic HTTP authentication information (plaintext username and password) for all requests, just like Wget 1.10.2 and prior did by default.

Use of this option is not recommended, and is intended only to support some few obscure servers, which never send HTTP authentication challenges, but accept unsolicited auth info, say, in addition to form-based authentication.

HTTPS (SSL/TLS) Options
To support encrypted HTTP (HTTPS) downloads, Wget must be compiled with an external SSL library. The current default is GnuTLS. In addition, Wget also supports HSTS (HTTP Strict Transport Security). If Wget is compiled without SSL support, none of these options are available.

−−secure−protocol=*protocol*
Choose the secure protocol to be used. Legal values are **auto**, **SSLv2**, **SSLv3**, **TLSv1**, **TLSv1_1**, **TLSv1_2** and **PFS**. If **auto** is used, the SSL library is given the liberty of choosing the appropriate protocol automatically, which is achieved by sending a TLSv1 greeting. This is the default.

Specifying **SSLv2**, **SSLv3**, **TLSv1**, **TLSv1_1** or **TLSv1_2** forces the use of the corresponding protocol. This is useful when talking to old and buggy SSL server implementations that make it hard for the underlying SSL library to choose the correct protocol version. Fortunately, such servers are quite rare.

Specifying **PFS** enforces the use of the so-called Perfect Forward Security cipher suites. In short, PFS adds security by creating a one-time key for each SSL connection. It has a bit more CPU impact on client and server. We use known to be secure ciphers (e.g. no MD4) and the TLS protocol.

−−https−only
When in recursive mode, only HTTPS links are followed.

−−no−check−certificate
Don’t check the server certificate against the available certificate authorities. Also don’t require the URL host name to match the common name presented by the certificate.

As of Wget 1.10, the default is to verify the server’s certificate against the recognized certificate authorities, breaking the SSL handshake and aborting the download if the verification fails. Although this provides more secure downloads, it does break interoperability with some sites that worked with previous Wget versions, particularly those using self-signed, expired, or otherwise invalid certificates. This option forces an ‘‘insecure’’ mode of operation that turns the certificate verification errors into warnings and allows you to proceed.

If you encounter ‘‘certificate verification’’ errors or ones saying that ‘‘common name doesn’t match requested host name’’, you can use this option to bypass the verification and proceed with the download. *Only use this option if you are otherwise convinced of the site’s authenticity, or if you really don’t care about the validity of its certificate.* It is almost always a bad idea not to check the certificates when transmitting confidential or important data. For self−signed/internal certificates, you

should download the certificate and verify against that instead of forcing this insecure mode. If you are really sure of not desiring any certificate verification, you can specify --check-certificate=quiet to tell wget to not print any warning about invalid certificates, albeit in most cases this is the wrong thing to do.

--certificate=*file*
Use the client certificate stored in *file*. This is needed for servers that are configured to require certificates from the clients that connect to them. Normally a certificate is not required and this switch is optional.

--certificate-type=*type*
Specify the type of the client certificate. Legal values are **PEM** (assumed by default) and **DER**, also known as **ASN1**.

--private-key=*file*
Read the private key from *file*. This allows you to provide the private key in a file separate from the certificate.

--private-key-type=*type*
Specify the type of the private key. Accepted values are **PEM** (the default) and **DER**.

--ca-certificate=*file*
Use *file* as the file with the bundle of certificate authorities (''CA'') to verify the peers. The certificates must be in PEM format.

Without this option Wget looks for CA certificates at the system-specified locations, chosen at OpenSSL installation time.

--ca-directory=*directory*
Specifies directory containing CA certificates in PEM format. Each file contains one CA certificate, and the file name is based on a hash value derived from the certificate. This is achieved by processing a certificate directory with the `c_rehash` utility supplied with OpenSSL. Using **--ca-directory** is more efficient than **--ca-certificate** when many certificates are installed because it allows Wget to fetch certificates on demand.

Without this option Wget looks for CA certificates at the system-specified locations, chosen at OpenSSL installation time.

--crl-file=*file*
Specifies a CRL file in *file*. This is needed for certificates that have been revocated by the CAs.

--pinnedpubkey=file/hashes
Tells wget to use the specified public key file (or hashes) to verify the peer. This can be a path to a file which contains a single public key in PEM or DER format, or any number of base64 encoded sha256 hashes preceded by ''sha256//'' and separated by '';''

When negotiating a TLS or SSL connection, the server sends a certificate indicating its identity. A public key is extracted from this certificate and if it does not exactly match the public key(s) provided to this option, wget will abort the connection before sending or receiving any data.

--random-file=*file*
[OpenSSL and LibreSSL only] Use *file* as the source of random data for seeding the pseudo-random number generator on systems without */dev/urandom*.

On such systems the SSL library needs an external source of randomness to initialize. Randomness may be provided by EGD (see **--egd-file** below) or read from an external source specified by the user. If this option is not specified, Wget looks for random data in `$RANDFILE` or, if that is unset, in *`$HOME`/.rnd*.

If you're getting the ''Could not seed OpenSSL PRNG; disabling SSL.'' error, you should provide random data using some of the methods described above.

−−egd−file=*file*

[OpenSSL only] Use *file* as the EGD socket. EGD stands for *Entropy Gathering Daemon*, a user-space program that collects data from various unpredictable system sources and makes it available to other programs that might need it. Encryption software, such as the SSL library, needs sources of non-repeating randomness to seed the random number generator used to produce cryptographically strong keys.

OpenSSL allows the user to specify his own source of entropy using the `RAND_FILE` environment variable. If this variable is unset, or if the specified file does not produce enough randomness, OpenSSL will read random data from EGD socket specified using this option.

If this option is not specified (and the equivalent startup command is not used), EGD is never contacted. EGD is not needed on modern Unix systems that support */dev/urandom*.

−−no−hsts

Wget supports HSTS (HTTP Strict Transport Security, RFC 6797) by default. Use **−−no−hsts** to make Wget act as a non-HSTS-compliant UA. As a consequence, Wget would ignore all the `Strict−Transport−Security` headers, and would not enforce any existing HSTS policy.

−−hsts−file=*file*

By default, Wget stores its HSTS database in *~/.wget−hsts*. You can use **−−hsts−file** to override this. Wget will use the supplied file as the HSTS database. Such file must conform to the correct HSTS database format used by Wget. If Wget cannot parse the provided file, the behaviour is unspecified.

The Wget's HSTS database is a plain text file. Each line contains an HSTS entry (ie. a site that has issued a `Strict−Transport−Security` header and that therefore has specified a concrete HSTS policy to be applied). Lines starting with a dash (`#`) are ignored by Wget. Please note that in spite of this convenient human-readability hand-hacking the HSTS database is generally not a good idea.

An HSTS entry line consists of several fields separated by one or more whitespace:

`<hostname> SP [<port>] SP <include subdomains> SP <created> SP <max−age>`

The *hostname* and *port* fields indicate the hostname and port to which the given HSTS policy applies. The *port* field may be zero, and it will, in most of the cases. That means that the port number will not be taken into account when deciding whether such HSTS policy should be applied on a given request (only the hostname will be evaluated). When *port* is different to zero, both the target hostname and the port will be evaluated and the HSTS policy will only be applied if both of them match. This feature has been included for testing/development purposes only. The Wget testsuite (in *testenv/*) creates HSTS databases with explicit ports with the purpose of ensuring Wget's correct behaviour. Applying HSTS policies to ports other than the default ones is discouraged by RFC 6797 (see Appendix B ''Differences between HSTS Policy and Same-Origin Policy''). Thus, this functionality should not be used in production environments and *port* will typically be zero. The last three fields do what they are expected to. The field *include_subdomains* can either be `1` or `0` and it signals whether the subdomains of the target domain should be part of the given HSTS policy as well. The *created* and *max-age* fields hold the timestamp values of when such entry was created (first seen by Wget) and the HSTS-defined value 'max−age', which states how long should that HSTS policy remain active, measured in seconds elapsed since the timestamp stored in *created*. Once that time has passed, that HSTS policy will no longer be valid and will eventually be removed from the database.

If you supply your own HSTS database via **−−hsts−file**, be aware that Wget may modify the provided file if any change occurs between the HSTS policies requested by the remote servers and those in the file. When Wget exists, it effectively updates the HSTS database by rewriting the database file with the new entries.

If the supplied file does not exist, Wget will create one. This file will contain the new HSTS entries. If no HSTS entries were generated (no `Strict−Transport−Security` headers were sent by any of the servers) then no file will be created, not even an empty one. This behaviour applies to the default database file (*~/.wget−hsts*) as well: it will not be created until some server enforces an HSTS policy.

Care is taken not to override possible changes made by other Wget processes at the same time over the HSTS database. Before dumping the updated HSTS entries on the file, Wget will re-read it and merge the changes.

Using a custom HSTS database and/or modifying an existing one is discouraged. For more information about the potential security threats arised from such practice, see section 14 ‘‘Security Considerations’’ of RFC 6797, specially section 14.9 ‘‘Creative Manipulation of HSTS Policy Store’’.

−−warc−file=*file*
Use *file* as the destination WARC file.

−−warc−header=*string*
Use *string* into as the warcinfo record.

−−warc−max−size=*size*
Set the maximum size of the WARC files to *size*.

−−warc−cdx
Write CDX index files.

−−warc−dedup=*file*
Do not store records listed in this CDX file.

−−no−warc−compression
Do not compress WARC files with GZIP.

−−no−warc−digests
Do not calculate SHA1 digests.

−−no−warc−keep−log
Do not store the log file in a WARC record.

−−warc−tempdir=*dir*
Specify the location for temporary files created by the WARC writer.

FTP Options

−−ftp−user=*user*

−−ftp−password=*password*
Specify the username *user* and password *password* on an FTP server. Without this, or the corresponding startup option, the password defaults to **−wget@**, normally used for anonymous FTP.

Another way to specify username and password is in the URL itself. Either method reveals your password to anyone who bothers to run `ps`. To prevent the passwords from being seen, store them in *.wgetrc* or *.netrc*, and make sure to protect those files from other users with `chmod`. If the passwords are really important, do not leave them lying in those files either−−−edit the files and delete them after Wget has started the download.

−−no−remove−listing
Don’t remove the temporary *.listing* files generated by FTP retrievals. Normally, these files contain the raw directory listings received from FTP servers. Not removing them can be useful for debugging purposes, or when you want to be able to easily check on the contents of remote server directories (e.g. to verify that a mirror you’re running is complete).

Note that even though Wget writes to a known filename for this file, this is not a security hole in the scenario of a user making *.listing* a symbolic link to */etc/passwd* or something and asking `root` to run Wget in his or her directory. Depending on the options used, either Wget will refuse to write to *.listing*, making the globbing/recursion/time−stamping operation fail, or the symbolic link will be deleted and replaced with the actual *.listing* file, or the listing will be written to a *.listing.number* file.

Even though this situation isn’t a problem, though, `root` should never run Wget in a non-trusted user’s directory. A user could do something as simple as linking *index.html* to */etc/passwd* and asking `root` to run Wget with **−N** or **−r** so the file will be overwritten.

−−no−glob
Turn off FTP globbing. Globbing refers to the use of shell-like special characters (*wildcards*), like *****, **?**, **[** and **]** to retrieve more than one file from the same directory at once, like:

```
wget ftp://gnjilux.srk.fer.hr/*.msg
```

By default, globbing will be turned on if the URL contains a globbing character. This option may be used to turn globbing on or off permanently.

You may have to quote the URL to protect it from being expanded by your shell. Globbing makes Wget look for a directory listing, which is system-specific. This is why it currently works only with Unix FTP servers (and the ones emulating Unix `ls` output).

−−no−passive−ftp
Disable the use of the *passive* FTP transfer mode. Passive FTP mandates that the client connect to the server to establish the data connection rather than the other way around.

If the machine is connected to the Internet directly, both passive and active FTP should work equally well. Behind most firewall and NAT configurations passive FTP has a better chance of working. However, in some rare firewall configurations, active FTP actually works when passive FTP doesn't. If you suspect this to be the case, use this option, or set `passive_ftp=off` in your init file.

−−preserve−permissions
Preserve remote file permissions instead of permissions set by umask.

−−retr−symlinks
By default, when retrieving FTP directories recursively and a symbolic link is encountered, the symbolic link is traversed and the pointed-to files are retrieved. Currently, Wget does not traverse symbolic links to directories to download them recursively, though this feature may be added in the future.

When **−−retr−symlinks=no** is specified, the linked-to file is not downloaded. Instead, a matching symbolic link is created on the local filesystem. The pointed-to file will not be retrieved unless this recursive retrieval would have encountered it separately and downloaded it anyway. This option poses a security risk where a malicious FTP Server may cause Wget to write to files outside of the intended directories through a specially crafted .LISTING file.

Note that when retrieving a file (not a directory) because it was specified on the command-line, rather than because it was recursed to, this option has no effect. Symbolic links are always traversed in this case.

FTPS Options

−−ftps−implicit
This option tells Wget to use FTPS implicitly. Implicit FTPS consists of initializing SSL/TLS from the very beginning of the control connection. This option does not send an `AUTH TLS` command: it assumes the server speaks FTPS and directly starts an SSL/TLS connection. If the attempt is successful, the session continues just like regular FTPS (`PBSZ` and `PROT` are sent, etc.). Implicit FTPS is no longer a requirement for FTPS implementations, and thus many servers may not support it. If **−−ftps−implicit** is passed and no explicit port number specified, the default port for implicit FTPS, 990, will be used, instead of the default port for the ''normal'' (explicit) FTPS which is the same as that of FTP, 21.

−−no−ftps−resume−ssl
Do not resume the SSL/TLS session in the data channel. When starting a data connection, Wget tries to resume the SSL/TLS session previously started in the control connection. SSL/TLS session resumption avoids performing an entirely new handshake by reusing the SSL/TLS parameters of a previous session. Typically, the FTPS servers want it that way, so Wget does this by default. Under rare circumstances however, one might want to start an entirely new SSL/TLS session in every data connection. This is what **−−no−ftps−resume−ssl** is for.

−−ftps−clear−data−connection
All the data connections will be in plain text. Only the control connection will be under SSL/TLS. Wget will send a `PROT C` command to achieve this, which must be approved by the server.

−−ftps−fallback−to−ftp
Fall back to FTP if FTPS is not supported by the target server. For security reasons, this option is not asserted by default. The default behaviour is to exit with an error. If a server does not successfully reply to the initial `AUTH TLS` command, or in the case of implicit FTPS, if the initial SSL/TLS connection attempt is rejected, it is considered that such server does not support FTPS.

Recursive Retrieval Options

−r
−−recursive
Turn on recursive retrieving. The default maximum depth is 5.

−l *depth*
−−level=*depth*
Specify recursion maximum depth level *depth*.

−−delete−after
This option tells Wget to delete every single file it downloads, *after* having done so. It is useful for pre-fetching popular pages through a proxy, e.g.:

```
wget −r −nd −−delete−after http://whatever.com/~popular/page/
```

The **−r** option is to retrieve recursively, and **−nd** to not create directories.

Note that **−−delete−after** deletes files on the local machine. It does not issue the **DELE** command to remote FTP sites, for instance. Also note that when **−−delete−after** is specified, **−−convert−links** is ignored, so **.orig** files are simply not created in the first place.

−k
−−convert−links
After the download is complete, convert the links in the document to make them suitable for local viewing. This affects not only the visible hyperlinks, but any part of the document that links to external content, such as embedded images, links to style sheets, hyperlinks to non-HTML content, etc.

Each link will be changed in one of the two ways:

- The links to files that have been downloaded by Wget will be changed to refer to the file they point to as a relative link.

  Example: if the downloaded file */foo/doc.html* links to */bar/img.gif*, also downloaded, then the link in *doc.html* will be modified to point to **../bar/img.gif**. This kind of transformation works reliably for arbitrary combinations of directories.

- The links to files that have not been downloaded by Wget will be changed to include host name and absolute path of the location they point to.

  Example: if the downloaded file */foo/doc.html* links to */bar/img.gif* (or to *../bar/img.gif*), then the link in *doc.html* will be modified to point to *http://hostname/bar/img.gif*.

Because of this, local browsing works reliably: if a linked file was downloaded, the link will refer to its local name; if it was not downloaded, the link will refer to its full Internet address rather than presenting a broken link. The fact that the former links are converted to relative links ensures that you can move the downloaded hierarchy to another directory.

Note that only at the end of the download can Wget know which links have been downloaded. Because of that, the work done by **−k** will be performed at the end of all the downloads.

−−convert−file−only
This option converts only the filename part of the URLs, leaving the rest of the URLs untouched. This filename part is sometimes referred to as the ‘‘basename’’, although we avoid that term here in order not to cause confusion.

It works particularly well in conjunction with **−−adjust−extension**, although this coupling is not enforced. It proves useful to populate Internet caches with files downloaded from different hosts.

Example: if some link points to *//foo.com/bar.cgi?xyz* with **−−adjust−extension** asserted and its local destination is intended to be *./foo.com/bar.cgi?xyz.css*, then the link would be converted to *//foo.com/bar.cgi?xyz.css*. Note that only the filename part has been modified. The rest of the URL has been left untouched, including the net path (`//`) which would otherwise be processed by Wget and converted to the effective scheme (ie. `http://`).

−K
−−backup−converted
When converting a file, back up the original version with a **.orig** suffix. Affects the behavior of **−N**.

−m
−−mirror
Turn on options suitable for mirroring. This option turns on recursion and time-stamping, sets infinite recursion depth and keeps FTP directory listings. It is currently equivalent to **−r −N −l inf −−no−remove−listing**.

−p
−−page−requisites
This option causes Wget to download all the files that are necessary to properly display a given HTML page. This includes such things as inlined images, sounds, and referenced stylesheets.

Ordinarily, when downloading a single HTML page, any requisite documents that may be needed to display it properly are not downloaded. Using **−r** together with **−l** can help, but since Wget does not ordinarily distinguish between external and inlined documents, one is generally left with ‘‘leaf documents’’ that are missing their requisites.

For instance, say document *1.html* contains an `<IMG>` tag referencing *1.gif* and an `<A>` tag pointing to external document *2.html*. Say that *2.html* is similar but that its image is *2.gif* and it links to *3.html*. Say this continues up to some arbitrarily high number.

If one executes the command:

```
wget −r −l 2 http://<site>/1.html
```

then *1.html*, *1.gif*, *2.html*, *2.gif*, and *3.html* will be downloaded. As you can see, *3.html* is without its requisite *3.gif* because Wget is simply counting the number of hops (up to 2) away from *1.html* in order to determine where to stop the recursion. However, with this command:

```
wget −r −l 2 −p http://<site>/1.html
```

all the above files *and 3.html*’s requisite *3.gif* will be downloaded. Similarly,

```
wget −r −l 1 −p http://<site>/1.html
```

will cause *1.html*, *1.gif*, *2.html*, and *2.gif* to be downloaded. One might think that:

```
wget −r −l 0 −p http://<site>/1.html
```

would download just *1.html* and *1.gif*, but unfortunately this is not the case, because **−l 0** is equivalent to **−l inf**−−−that is, infinite recursion. To download a single HTML page (or a handful of them, all specified on the command-line or in a **−i** URL input file) and its (or their) requisites, simply leave off **−r** and **−l**:

```
wget −p http://<site>/1.html
```

Note that Wget will behave as if **−r** had been specified, but only that single page and its requisites will

be downloaded. Links from that page to external documents will not be followed. Actually, to download a single page and all its requisites (even if they exist on separate websites), and make sure the lot displays properly locally, this author likes to use a few options in addition to **−p**:

```
wget −E −H −k −K −p http://<site>/<document>
```

To finish off this topic, it's worth knowing that Wget's idea of an external document link is any URL specified in an `<A>` tag, an `<AREA>` tag, or a `<LINK>` tag other than `<LINK REL="stylesheet">`.

−−strict−comments

Turn on strict parsing of HTML comments. The default is to terminate comments at the first occurrence of −−>.

According to specifications, HTML comments are expressed as SGML *declarations*. Declaration is special markup that begins with **<!** and ends with **>**, such as **<!DOCTYPE ...>**, that may contain comments between a pair of −− delimiters. HTML comments are ''empty declarations'', SGML declarations without any non-comment text. Therefore, **<!−−foo−−>** is a valid comment, and so is **<!−−one — −−two−−>**, but **<!−−1−−2−−>** is not.

On the other hand, most HTML writers don't perceive comments as anything other than text delimited with **<!−−** and **−−>**, which is not quite the same. For example, something like **<!−−−−−−−−−−−−>** works as a valid comment as long as the number of dashes is a multiple of four (!). If not, the comment technically lasts until the next −−, which may be at the other end of the document. Because of this, many popular browsers completely ignore the specification and implement what users have come to expect: comments delimited with **<!−−** and **−−>**.

Until version 1.9, Wget interpreted comments strictly, which resulted in missing links in many web pages that displayed fine in browsers, but had the misfortune of containing non-compliant comments. Beginning with version 1.9, Wget has joined the ranks of clients that implements ''naive'' comments, terminating each comment at the first occurrence of −−>.

If, for whatever reason, you want strict comment parsing, use this option to turn it on.

Recursive Accept/Reject Options

−A *acclist* **−−accept** *acclist*

−R *rejlist* **−−reject** *rejlist*

Specify comma-separated lists of file name suffixes or patterns to accept or reject. Note that if any of the wildcard characters, *****, **?**, **[** or **]**, appear in an element of *acclist* or *rejlist*, it will be treated as a pattern, rather than a suffix. In this case, you have to enclose the pattern into quotes to prevent your shell from expanding it, like in **−A "*.mp3"** or **−A '*.mp3'**.

−−accept−regex *urlregex*

−−reject−regex *urlregex*

Specify a regular expression to accept or reject the complete URL.

−−regex−type *regextype*

Specify the regular expression type. Possible types are **posix** or **pcre**. Note that to be able to use **pcre** type, wget has to be compiled with libpcre support.

−D *domain-list*

−−domains=*domain-list*

Set domains to be followed. *domain-list* is a comma-separated list of domains. Note that it does *not* turn on **−H**.

−−exclude−domains *domain-list*

Specify the domains that are *not* to be followed.

−−follow−ftp

Follow FTP links from HTML documents. Without this option, Wget will ignore all the FTP links.

−−follow−tags=*list*
Wget has an internal table of HTML tag / attribute pairs that it considers when looking for linked documents during a recursive retrieval. If a user wants only a subset of those tags to be considered, however, he or she should be specify such tags in a comma-separated *list* with this option.

−−ignore−tags=*list*
This is the opposite of the **−−follow−tags** option. To skip certain HTML tags when recursively looking for documents to download, specify them in a comma-separated *list*.

In the past, this option was the best bet for downloading a single page and its requisites, using a command-line like:

```
wget --ignore-tags=a,area -H -k -K -r http://<site>/<document>
```

However, the author of this option came across a page with tags like `<LINK REL="home" HREF="/">` and came to the realization that specifying tags to ignore was not enough. One can't just tell Wget to ignore `<LINK>`, because then stylesheets will not be downloaded. Now the best bet for downloading a single page and its requisites is the dedicated **−−page−requisites** option.

−−ignore−case
Ignore case when matching files and directories. This influences the behavior of −R, −A, −I, and −X options, as well as globbing implemented when downloading from FTP sites. For example, with this option, **−A "*.txt"** will match **file1.txt**, but also **file2.TXT**, **file3.TxT**, and so on. The quotes in the example are to prevent the shell from expanding the pattern.

−H
−−span−hosts
Enable spanning across hosts when doing recursive retrieving.

−L
−−relative
Follow relative links only. Useful for retrieving a specific home page without any distractions, not even those from the same hosts.

−I *list*
−−include−directories=*list*
Specify a comma-separated list of directories you wish to follow when downloading. Elements of *list* may contain wildcards.

−X *list*
−−exclude−directories=*list*
Specify a comma-separated list of directories you wish to exclude from download. Elements of *list* may contain wildcards.

−np
−−no−parent
Do not ever ascend to the parent directory when retrieving recursively. This is a useful option, since it guarantees that only the files *below* a certain hierarchy will be downloaded.

ENVIRONMENT

Wget supports proxies for both HTTP and FTP retrievals. The standard way to specify proxy location, which Wget recognizes, is using the following environment variables:

http_proxy
https_proxy
If set, the **http_proxy** and **https_proxy** variables should contain the URLs of the proxies for HTTP and HTTPS connections respectively.

ftp_proxy
This variable should contain the URL of the proxy for FTP connections. It is quite common that **http_proxy** and **ftp_proxy** are set to the same URL.

no_proxy
: This variable should contain a comma-separated list of domain extensions proxy should *not* be used for. For instance, if the value of **no_proxy** is **.mit.edu**, proxy will not be used to retrieve documents from MIT.

EXIT STATUS

Wget may return one of several error codes if it encounters problems.

0 No problems occurred.

1 Generic error code.

2 Parse error---for instance, when parsing command-line options, the **.wgetrc** or **.netrc**...

3 File I/O error.

4 Network failure.

5 SSL verification failure.

6 Username/password authentication failure.

7 Protocol errors.

8 Server issued an error response.

With the exceptions of 0 and 1, the lower-numbered exit codes take precedence over higher-numbered ones, when multiple types of errors are encountered.

In versions of Wget prior to 1.12, Wget's exit status tended to be unhelpful and inconsistent. Recursive downloads would virtually always return 0 (success), regardless of any issues encountered, and non-recursive fetches only returned the status corresponding to the most recently-attempted download.

FILES

/etc/wgetrc
: Default location of the *global* startup file.

.wgetrc
: User startup file.

BUGS

You are welcome to submit bug reports via the GNU Wget bug tracker (see <**https://savannah.gnu.org/bugs/?func=additem&group=wget**>).

Before actually submitting a bug report, please try to follow a few simple guidelines.

1. Please try to ascertain that the behavior you see really is a bug. If Wget crashes, it's a bug. If Wget does not behave as documented, it's a bug. If things work strange, but you are not sure about the way they are supposed to work, it might well be a bug, but you might want to double-check the documentation and the mailing lists.

2. Try to repeat the bug in as simple circumstances as possible. E.g. if Wget crashes while downloading **wget -rl0 -kKE -t5 --no-proxy http://example.com -o /tmp/log**, you should try to see if the crash is repeatable, and if will occur with a simpler set of options. You might even try to start the download at the page where the crash occurred to see if that page somehow triggered the crash.

   Also, while I will probably be interested to know the contents of your *.wgetrc* file, just dumping it into the debug message is probably a bad idea. Instead, you should first try to see if the bug repeats with *.wgetrc* moved out of the way. Only if it turns out that *.wgetrc* settings affect the bug, mail me the relevant parts of the file.

3. Please start Wget with **-d** option and send us the resulting output (or relevant parts thereof). If Wget was compiled without debug support, recompile it---it is *much* easier to trace bugs with debug support on.

   Note: please make sure to remove any potentially sensitive information from the debug log before sending it to the bug address. The `-d` won't go out of its way to collect sensitive information, but the

log *will* contain a fairly complete transcript of Wget's communication with the server, which may include passwords and pieces of downloaded data. Since the bug address is publically archived, you may assume that all bug reports are visible to the public.

4. If Wget has crashed, try to run it in a debugger, e.g. `gdb `which wget` core` and type `where` to get the backtrace. This may not work if the system administrator has disabled core files, but it is safe to try.

SEE ALSO

This is **not** the complete manual for GNU Wget. For more complete information, including more detailed explanations of some of the options, and a number of commands available for use with *.wgetrc* files and the **−e** option, see the GNU Info entry for *wget*.

AUTHOR

Originally written by Hrvoje NikXiX <hniksic@xemacs.org>.

COPYRIGHT