Unix Bible Commands

A list of commands and a quick description
alias …….. this allows the user view the current aliases
awk …….. this allows the user to search for a pattern within a file
bdiff …….. compares two large files
bfs ……. scans a large file
cal …….. shows a calendar
cat …….. concatenates and prints a file
cc …….. c compiler
cd …….. changes directories
chgrb …….. changes a file groups ownership
chmod ……. changes the permission on a file
chown ……. changes the individual ownership of a file
cmp …….. compairs two files
comm ……. compares two files so as to determine which lines are common to both
cp ……. copies file to another location
cu …….. calls another unix sysytem
date ……. returns the date and time
df …….. shows all mounted drives on your machine
diff …….. displays the difference between two files
du …….. shows the disk usage in blocks for a directory
echo …….. echoes the data to the screen or file
ed …….. text editor
env …….. lists the current environment variables
ex …….. another text editor
expr …….. evaluates a mathematical formula
find …….. finds a file
f77 ……. fortran complier
format …….. initializes a floppy disk
grep …….. searches for a pattern within a file
help …….. gives help
kill …….. stops a running process
ln …….. creates a link between two files
lpr …….. copies the file to the line printer
ls …….. lists the files in a directory
mail …….. allows the user to send/receive mail
mkdir ……. makes directory
more …….. displays a data file to the screen
mv ……. used to move or rename files
nohup …….. allows a command to continue running even when you log out
nroff …….. used to format text
passwd …….. changes your password
pkgadd …….. installs a new program onto your machine
ps ……. Lists the current processes running
pwd …….. displays the name of the working directory
rm …….. removes files
rmdir …….. removes directories
set …….. lists all the variables in the current shell
setenv …….. sets the environment variables
sleep …….. causes a process to become inactive
source …….. allows the user to execute a file and update any changed values in that file
sort ……. sorts files
spell …….. checks for spelling errors in a file

split ..... divides a file
stty ..... sets the terminal options
tail ..... displays the end of a file
tar ..... copies all specified files into one
touch ..... creates an empty file or updates the time/date stamp on a file
troff ..... outputs formatted output
tset ..... sets the terminal type
umask ..... specify a new creation mask
uniq ..... compairs two files
uucp ..... unix to unix execute
vi ..... full screen editor
vipw ..... opens the vi editor as well as password file for editing
volcheck ..... checks to see if there is a floppy disk mounted to your machine
wc ..... displays detail in the full size
who ..... inf. on other people online
write ..... send a message to another user
! ..... repeats commands

More commands with a better description (Not all commands are listed).
cat: -b, --number-nonblank
Number all nonblank output lines, starting with 1.

-e
Equivalent to -vE.

-n, --number
Number all output lines, starting with 1.

-s, --squeeze-blank
Replace multiple adjacent blank lines with a single blank line.

-t
Equivalent to -vT.

-u
Ignored; for Unix compatibility.

-v, --show-nonprinting
Display control characters except for LFD and TAB using `^' notation and precede characters that have the high bit set with `M-'.

-A, --show-all
Equivalent to -vET.

-E, --show-ends
Display a `$' after the end of each line.

-T, --show-tabs
Display TAB characters as `^I'.

--help
Print a usage message and exit with a status code indicating success.

--version
Print version information on standard output then exit.

cd: directory becomes the new working directory. The process must have execute (search) permission
in directory. If cd is used without arguments, it returns you to your login directory. In csh you
may specify a list of directories in which directory is to be sought as a subdirectory if it is not a
subdirectory of the current directory; see the description of the cdpath variable in csh.
chmod: The format of a symbolic mode is `[ugoa...][[+=][rwxXstugo...]...][,...]'. Multiple symbolic
operations
can be given, separated by commas.

A combination of the letters `ugoa' controls which users' access to the file will be changed: the user
who owns it (u), other users in the file's group (g), other users not in the file's group (o), or all users
(a). If none of these are given, the effect is as if `a' were given, but bits that are set in the umask are
not affected.

The operator `+' causes the permissions selected to be added to the existing permissions of each
file; `-' causes them to be removed; and `=' causes them to be the only permissions that the file has.

The letters `rwxXstugo' select the new permissions for the affected users: read (r), write (w),
execute (or access for directories) (x), execute only if the file is a directory or already has execute
permission for some user (X), set user or group ID on execution (s), save program text on swap
device (t), the permissions that the user who owns the file currently has for it (u), the permissions
that other users in the file's group have for it (g), and the permissions that other users not in the file's
group have for it (o).

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4,
2, and 1. Any omitted digits are assumed to be leading zeros. The first digit selects the set user ID
(4) and set group ID (2) and save text image (1) attributes. The second digit selects permissions for
the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for
other users in the file's group, with the same values; and the fourth for other users not in the file's
group, with the same values.

chmod never changes the permissions of symbolic links; the chmod system call cannot change their
permissions. This is not a problem since the permissions of symbolic links are never used. However,
for each symbolic link listed on the command line, chmod changes the permissions of the pointedto

file. In contrast, chmod ignores symbolic links encountered during recursive directory traversals.

OPTIONS
-c, --changes
Verbosely describe only files whose permissions actually change.

-f, --silent, --quiet
Do not print error messages about files whose permissions cannot be changed.

-v, --verbose
Verbosely describe changed permissions.

-R, --recursive
Recursively change permissions of directories and their contents.

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

--version
Print version information on standard output then exit successfully.
clear: clear clears your screen if this is possible. It looks in the environment for the terminal type and then
in /etc/termcap to figure out how to clear the screen.
date: This manual page documents the GNU version of date. date with no arguments prints the current
time and date (in the format of the `%c' directive described below). If given an argument that starts
with a `+', it prints the current time and date in a format controlled by that argument, which has the
same format as the format string passed to the `strftime' function. Except for directives that start with
`%', characters in that string are printed unchanged.

The directives are:

%
a literal %

n
a newline

t
a horizontal tab

Time fields:

%H
hour (00..23)

%I
hour (01..12)

%k
hour (0..23)

%I
    hour (1..12)

%M
    minute (00..59)

%p
    locale's AM or PM

%r
    time, 12-hour (hh:mm:ss [AP]M)

%s seconds since 1970-01-01 00:00:00 UTC (a nonstandard extension)

%S
    second (00..61)

%T
    time, 24-hour (hh:mm:ss)

%X
    locale's time representation (%H:%M:%S)

%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)

%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) with 0 corresponding to Sunday

%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...)

By default, date pads numeric fields with zeroes. GNU date recognizes the following nonstandard
numeric modifiers:

-
(hyphen) do not pad the field

_
(underscore) pad the field with spaces

If given an argument that does not start with `+', date sets the system clock to the time and date
specified by that argument. The argument must consist entirely of digits, which have the following
meaning:

MM month

DD
day within month

hh hour

mm minute

CC
first two digits of year (optional)

YY
last two digits of year (optional)

ss
second (optional)

Only the superuser can set the system clock.

OPTIONS
-d datestr, --date datestr
Display the time and date specified in datestr, which can be in almost any common format. The
display is in the default output format, or if an argument starting with `+' is given to

date, in the
format specified by that argument.

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

-s datestr, --set datestr
    Set the time and date to datestr, which can be in almost any common format. It can contain
    month names, timezones, `am' and `pm', etc.

-u, --universal
    Print or set the time and date in Coordinated Universal Time (also known as Greenwich
    Mean Time) instead of in local (wall clock) time.

--version
    Print version information on standard output then exit successfully.
find: find recursively descends the directory hierarchy for each pathname in the pathname-list, seeking
files that match a logical expression written using the operators listed below.

find does not follow symbolic links to other files or directories; it applies the selection criteria to the
symbolic links themselves, as if they were ordinary files (see ln(1V) for a description of symbolic
links).

If the fast-find feature is enabled, find displays pathnames in which a filename component occurs.

USAGE

Operators
In the descriptions, the argument n is used as a decimal integer where +n means more than n, -n
means less than n, and n means exactly n.

-fstype type
    True if the filesystem to which the file belongs is of type type, where type is typically 4.2 or
    nfs.

-name filename True if the
    filename argument matches the current file name. Shell argument syntax can be used if
    escaped (watch out for [, ? and *).

-perm onum
    True if the file permission flags exactly match the octal number onum (see chmod(1V)). If
    onum is prefixed by a minus sign, more flag bits (017777, see chmod(1V)) become
    significant and the flags are compared: (flags&onum)==onum.

-prune
    Always yields true. Has the side effect of pruning the search tree at the file. That is,

if the
current path name is a directory, find will not descend into that directory.

-type c
True if the type of the file is c, where c is one of:

b
for block special file c

c
for character special file

d
for directory

f
for plain file

p
for named pipe (FIFO)

l
for symbolic link

s
for socket

-links n
True if the file has n links.

-user uname
True if the file belongs to the user uname. If uname is numeric and does not appear as a
login name in the /etc/passwd database, it is taken as a user ID.

-nouser
True if the file belongs to a user not in the /etc/passwd database.

-group gname
True if the file belongs to group gname. If gname is numeric and does not appear as a login
name in the /etc/group database, it is taken as a group ID.

-nogroup
True if the file belongs to a group not in the /etc/group database.

-size n
True if the file is n blocks long (512 bytes per block). If n is followed by a c, the size is in
characters.

-inum n
True if the file has inode number n.

-atime n
True if the file has been accessed in n days. Note: the access time of directories in

path-name-list is changed by find itself.

-mtime n
True if the file has been modified in n days.

-ctime n
True if the file has been changed in n days. "Changed" means either that the file has been modified or some attribute of the file (its owner, its group, the number of links to it, etc.) has been changed.

-exec command
True if the executed command returns a zero value as exit status. The end of command must be punctuated by an escaped semicolon. A command argument {} is replaced by the current pathname.

-ok command
Like -exec except that the generated command is written on the standard output, then the standard input is read and the command executed only upon response y.

-print
Always true; the current pathname is printed.

-ls
Always true; prints current pathname together with its associated statistics. These include (respectively) inode number, size in kilobytes (1024 bytes), protection mode, number of hard links, user, group, size in bytes, and modification time. If the file is a special file the size field will instead contain the major and minor device numbers. If the file is a symbolic link the pathname of the linked-to file is printed preceded by `->'. The format is identical to that of ls -gilds (see ls(1V)). Note: formatting is done internally, without executing the ls program.

-cpio device
Always true; write the current file on device in cpio(5) format (5120-byte records).

-ncpio device
Always true; write the current file on device in cpio -c format (5120-byte records).

-newer file
True if the current file has been modified more recently than the argument filename.

-xdev
Always true; find does not traverse down into a file system different from the one on which current argument pathname resides.

-depth
    Always true; find descends the directory hierarchy, acting on the entries in a directory before
    acting on the directory itself. This can be useful when find is used with cpio to transfer files
    that are contained in directories without write permission.

(expression)
    True if the parenthesized expression is true. Note: Parentheses are special to the shell and
    must be escaped.

!primary
    True if the primary is false (! is the unary not operator).

primary1 [-a] primary2
True if both primary1 and primary2 are true. The -a is not required. It is implied by the juxtaposition of two primaries.

primary1 -o primary2
True if either primary1 or primary2 is true (-o is the or operator).

Fast-Find
The fast-find feature is enabled by the presence of the find.codes database in /usr/lib/find. You must
be root to build or update this database by running the updatedb script in that same directory. You
may wish to modify the updatedb script to suit your needs.

An alternate database can be specified by setting the FCODES environment variable.
cp: cp copies the contents of filename1 onto filename2. The mode and owner of filename2 are
preserved if it already existed; the mode of the source file is used otherwise. If filename1 is a
symbolic link, or a duplicate hard link, the contents of the file that the link refers to are copied; links
are not preserved.

In the second form, cp recursively copies directory1, along with its contents and subdirectories, to
directory2. If directory2 does not exist, cp creates it and duplicates the files and subdirectories of
directory1 within it. If direc_tory2 does exist, cp makes a copy of the directory1 directory within
directory2 (as a subdirectory), along with its files and subdirectories.

In the third form, each filename is copied to the indicated directory; the basename of the copy
corresponds to that of the original. The destination directory must already exist for the copy to
succeed.

cp refuses to copy a file onto itself.
finger: By default, finger displays information about each logged-in user, including his or her: login name, full
name, terminal name (prepended with a `*' if write-permission is denied), idle time,

login time, and
location (comment field in /etc/ttytab for users logged in locally, hostname for users logged in
remotely) if known.

Idle time is minutes if it is a single integer, hours and minutes if a `:' is present, or days and hours if a
d is present.

When one or more name arguments are given, more detailed information is given for each name
specified, whether they are logged in or not. A name may be a first or last name, or an account
name. Information is presented in a multiline format, and includes, in addition to the information
mentioned above:
the user's home directory and login shell the time they logged in if they are currently logged in, or the
time they last logged in if they are not, as well as the terminal or host from which they logged in and,
if a terminal, the comment field in /etc/ttytab for that terminal
the last time they received mail, and the last time they read their mail
any plan contained in the file .plan in the user's home directory
and any project on which they are working described in the file .project (also in that directory)

If a name argument contains an at-sign, `@', then a connection is attempted to the machine named
after the at-sign, and the remote finger daemon is queried. The data returned by that daemon is
printed. If a long format printout is to be produced, finger passes the -l option to the remote finger
daemon over the network using the /W feature of the protocol (see NAME/FINGER Protocol).
grep: Grep searches the named input files (or standard input if no files are named, or the file name - is
given) for lines containing a match to the given pattern. By default, grep prints the matching lines.

There are three major variants of grep, controlled by the following options.
-G Interpret pattern as a basic regular expression (see below). This is the default.
-E Interpret pattern as an extended regular expression (see below).

-F
    Interpret pattern as a list of fixed strings, separated by newlines, any of which is to be
    matched. In addition, two variant programs egrep and fgrep are available. Egrep is similiar
    (but not identical) to grep -E, and is compatible with the historical Unix egrep. Fgrep is the
    same as grep -F.

All variants of grep understand the following options: -num Matches will be printed with num lines
of leading and trailing context. However, grep will never print any given line more than once.

-A num
Print num lines of trailing context after matching lines.
-B num
Print num lines of leading context before matching lines.

-C
Equivalent to -2. -V Print the version number of grep to standard error. This version number
should be included in all bug reports (see below).

-b
Print the byte offset within the input file before each line of output.

-c
Suppress normal output; instead print a count of matching lines for each input file. With the -v
option (see below), count non-matching lines. -e pattern Use pattern as the pattern; useful
to protect patterns beginning with -. -f file Obtain the pattern from file.

-h
Suppress the prefixing of filenames on output when multiple files are searched.

-i
Ignore case distinctions in both the pattern and the input files. -L Suppress normal output;
instead print the name of each input file from which no output would normally have been
printed. -l Suppress normal output; instead print the name of each input file from which output
would normally have been printed. -n Prefix each line of output with the line number within its
input file.

-q
Quiet; suppress normal output.

-s
Suppress error messages about nonexistent or unreadable files. -v Invert the sense of
matching, to select non-matching lines. -w Select only those lines containing matches that
form whole words. The test is that the matching substring must either be at the beginning of
the line, or preceded by a non-word constituent character. Similarly, it must be either at the
end of the line or followed by a non-word constituent character. Word-constituent characters
are letters, digits, and the underscore. -x Select only those matches that exactly match the
whole line.
kill: kill sends the TERM (terminate, 15) signal to the processes with the specified pids. If a signal name
or number preceded by `-' is given as first argument, that signal is sent instead of terminate. The
signal names are listed by using the -l option, and are as given in <signal.h>, stripped

of the common
SIG prefix.

The terminate signal will kill processes that do not catch the signal, so `kill -9 ...' is a sure kill, as the
KILL (9) signal cannot be caught. By convention, if process number 0 is specified, all members in
the process group (that is, processes resulting from the current login) are signaled (but beware: this
works only if you use sh(1); not if you use csh(1).) Negative process numbers also have special
meanings; see kill(2V) for details. The killed processes must belong to the current user unless he is
the super-user.

To shut the system down and bring it up single user the super-user may send the initialization
process a TERM (terminate) signal by `kill 1'; see init(8). To force init to close and open terminals
according to what is currently in /etc/ttytab use `kill -HUP 1' (sending a hangup signal to process 1).

The shell reports the process number of an asynchronous process started with `&' (run in the
background). Process numbers can also be found by using ps(1).

kill is built in to csh(1); it allows job specifiers, such as `kill % ...', in place of kill arguments. See
csh(1) for details.
less: Less is a program similar to more (1), but which allows backwards movement in the file as well as
forward movement. Also, less does not have to read the entire input file before starting, so with
large input files it starts up faster than text editors like vi (1). Less uses termcap (or terminfo on
some systems), so it can run on a variety of terminals. There is even limited support for hardcopy
terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are
prefixed with an up-arrow.)

Commands are based on both more and vi. Commands may be preceded by a decimal number,
called N in the descriptions below. The number is used by some commands, as indicated.
In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for example
ESC-v means the two character sequence "ESCAPE", then "v".

H
    Help: display a summary of these commands. If you forget all the other commands, remember
    this one.

SPACE or f or ^F or ^V
Scroll forward N lines, default one window (see option -z below). If N is more than the

screen size,
only the final screenful is displayed. Warning: some systems use ^V as a special literalization
character.

b or ^B or ESC-v
Scroll backward N lines, default one window (see option -z below). If N is more than the screen
size, only the final screenful is displayed.

RETURN or ^N or e or ^E or j or ^J
Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen
size.

y or ^Y or ^P or k or ^K
Scroll backward N lines, default 1. The entire N lines are displayed, even if N is more than the
screen size. Warning: some systems use ^Y as a special job control character.

d or ^D
Scroll forward N lines, default one half of the screen size. If N is specified, it becomes the new
default for subsequent d and u commands.

u or ^U
Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new
default for subsequent d and u commands.

r or ^R or ^L
Repaint the screen.

R
Repaint the screen, discarding any buffered input. Useful if the file is changing while it is being
viewed.

g or < or ESC-<
Go to line N in the file, default 1 (beginning of file). (Warning: this may be slow if N is large.)

G or > or ESC->
Go to line N in the file, default the end of the file. (Warning: this may be slow if N is large, or if N is
not specified and standard input, rather than a file, is being read.)

p or %
Go to a position N percent into the file. N should be between 0 and 100. (This works if standard
input is being read, but only if less has already read to the end of the file. It is always fast, but not
always useful.)

m
Followed by any lowercase letter, marks the current position with that letter.

`
(Single quote.) Followed by any lowercase letter, returns to the position which was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. All marks are lost when a new file is examined.

^X^X Same as single quote.

/pattern
Search forward in the file for the N-th line containing the pattern. N defaults to 1. The pattern is a regular expression, as recognized by ed. The search starts at the second line displayed (but see the -a option, which changes this).

?pattern
Search backward in the file for the N-th line containing the pattern. The search starts at the line immediately before the top line displayed.

/!pattern
Like /, but the search is for the N-th line which does NOT contain the pattern.

?!pattern
Like ?, but the search is for the N-th line which does NOT contain the pattern.

n
Repeat previous search, for N-th line containing the last pattern (or NOT containing the last pattern, if the previous search was /! or ?!).

E [filename]
Examine a new file. If the filename is missing, the "current" file (see the N and P commands below) from the list of files in the command line is re-examined. If the filename is a pound sign (#), the previously examined file is re-examined.

^X^V or :e
Same as E. Warning: some systems use ^V as a special literalization character.

N or :n
Examine the next file (from the list of files given in the command line). If a number N is specified (not to be confused with the command N), the N-th next file is examined.

P or :p
Examine the previous file. If a number N is specified, the N-th previous file is examined.

= or ^G
Prints some information about the file being viewed, including its name and the line number and byte

offset of the bottom line being displayed. If possible, it also prints the length of the file and the
percent of the file above the last displayed line.

-
Followed by one of the command line option letters (see below), this will change the setting
of that option and print a message describing the new setting. If the option letter has a
numeric value (such as -b or -h), or a string value (such as -P or -t), a new value may be
entered after the option letter.

_
(Underscore.) Followed by one of the command line option letters (see below), this will print
a message describing the current setting of that option. The setting of the option is not
changed.

+cmd Causes the specified cmd to be executed each time a new
file is examined. For example, +G causes less to initially display each file starting at the end
rather than the beginning.

V
Prints the version number of less being run.

q or :q or ZZ
Exits less.

The following two commands may or may not be valid, depending on your particular installation.

v
Invokes an editor to edit the current file being viewed. The editor is taken from the
environment variable EDITOR, or defaults to "vi".

! shell-command
Invokes a shell to run the shell-command given. A percent sign in the command is replaced by the
name of the current file. "!!" repeats the last shell command. "!" with no shell command simply
invokes a shell. In all cases, the shell is taken from the environment variable SHELL, or defaults to
"sh".
logout: Built-in commands are executed within the C shell. If a built-in command occurs as any component
of a pipeline except the last, it is executed in a subshell.

:
Null command. This command is interpreted, but performs no action.

alias [name [def]]
Assign def to the alias name. def is a list of words that may contain escaped history substitution

metasyntax. name is not allowed to be alias or unalias. If def is omitted, the alias name is displayed
along with its current definition. If both name and def are omitted, all aliases are displayed.

bg [%job] ...
Run the current or specified jobs in the background.

break
    Resume execution after the end of the nearest enclosing foreach or while loop. The remaining
    commands on the current line are executed. This allows multilevel breaks to be written as a
    list of break commands, all on one line.

breaksw
    Break from a switch, resuming after the endsw.

case label:
A label in a switch statement.

cd [dir]
chdir [dir]
Change the shell's working directory-to-directory dir. If no argument is given, change to the home
directory of the user. If dir is a relative pathname not found in the current directory, check for it in
those directories listed in the cdpath variable. If dir is the name of a shell variable whose value starts
with a /, change to the directory named by that value.

continue Continue execution of the nearest enclosing while or foreach.

default: Labels the default case in a switch statement. The default should come after all case labels.
Any remaining commands on the command line are first executed.

dirs [-l]
Print the directory stack, most recent to the left; the first directory shown is the current directory.
With the -l argument, produce an unabbreviated printout; use of the ~ notation is suppressed.

echo [-n] list
The words in list are written to the shell's standard output, separated by SPACE characters. The
output is terminated with a NEWLINE unless the -n option is used.

eval argument ...
Reads the arguments as input to the shell, and executes the resulting command(s). This is usually
used to execute commands generated as the result of command or variable substitution, since
parsing occurs before these substitutions. See tset(1) for an example of how to use eval.

exec command
Execute command in place of the current shell, which terminates.

exit [(expr)]
The shell exits, either with the value of the status variable, or with the value of the specified by the
expression expr.

fg % [job]
Bring the current or specified job into the foreground.

foreach var (wordlist)
...

end
    The variable var is successively set to each member of wordlist. The sequence of commands
    between this command and the matching end is executed for each new value of var. (Both
    foreach and end must appear alone on separate lines.)

The built-in command continue may be used to continue the loop prematurely and the built-in
command break to terminate it prematurely. When this command is read from the terminal, the loop
is read up once prompting with ? before any statements in the loop are executed.

glob wordlist
Perform filename expansion on wordlist. Like echo, but no \ escapes are recognized. Words are
delimited by null characters in the output.

goto label
The specified label is filename and command expanded to yield a label. The shell rewinds its input
as much as possible and searches for a line of the form label: possibly preceded by SPACE or
TAB characters. Execution continues after the indicated line. It is an error to jump to a label that
occurs between a while or for built-in, and its corresponding end.

hashstat Print a statistics line indicating how effective the internal hash table has been at locating
commands (and avoiding execs). An exec is attempted for each component of the path where the
hash function indicates a possible hit, and in each component that does not begin with a `/'.

history [-hr] [n]
Display the history list; if n is given, display only the n most recent events.

-r
    Reverse the order of printout to be most recent first rather than oldest first. -h Display the
    history list without leading numbers. This is used to produce files suitable for sourcing using

the -h option to source.

if (expr) command
If the specified expression evaluates to true, the single command with arguments is executed.
Variable substitution on command happens early, at the same time it does for the rest of the if
command. command must be a simple command, not a pipeline, a command list, or a parenthesized
command list. Note: I/O redirection occurs even if expr is false, when command is not executed
(this is a bug).

if (expr) then
...
else if (expr2) then ...
else
...

endif
If expr is true, commands up to the first else are executed. Otherwise, if expr2 is true, the
commands between the else if and the second else are executed. Otherwise, commands
between the else and the endif are executed. Any number of else if pairs are allowed, but only
one else. Only one endif is needed, but it is required. The words else and endif must be the
first nonwhite characters on a line. The if must appear alone on its input line or after an else.)

jobs[-l]
List the active jobs under job control.

-l
List process IDs, in addition to the normal information.

kill [-sig] [pid] [%job] ...
kill -l Send the TERM (terminate) signal, by default, or the signal specified, to the specified process
ID, the job indicated, or the current job. Signals are either given by number or by name. There is no
default. Typing kill does not send a signal to the current job. If the signal being sent is TERM
(terminate) or HUP (hangup), then the job or process is sent a CONT (continue) signal as well.

-l
List the signal names that can be sent.

limit [-h] [resource [max-use]] Limit the consumption by the current process or any process it
spawns, each not to exceed max-use on the specified resource. If max-use is omitted, print the
current limit; if resource is omitted, display all limits.

-h
    Use hard limits instead of the current limits. Hard limits impose a ceiling on the values of the
    current limits. Only the super-user may raise the hard limits.

resource is one of:

cputime
    Maximum CPU seconds per process.

filesize
    Largest single file allowed.

datasize
    Maximum data size (including stack) for the process.

stacksize
    Maximum stack size for the

process.
coredumpsize Maximum size of a core dump (file).
descriptors Maximum value for a file descriptor.

max-use is a number, with an optional scaling factor, as follows:

nh
    Hours (for cputime).

nk
    n kilobytes. This is the default for all but cputime.

nm
    n megabytes or minutes (for cputime).

mm:ss
    Minutes and seconds (for cputime).

login [username|-p]
Terminate a login shell and invoke login(1). The .logout file is not processed. If username is omitted,
login prompts for the name of a user.

-p
    Preserve the current environment (variables).

logout
    Terminate a login shell.

nice [+n|-n] [command]
Increment the process priority value for the shell or for command by n. The higher the priority
value, the lower the priority of a process, and the slower it runs. When given, command is always
run in a subshell, and the restrictions placed on commands in simple if commands apply. If
command is omitted, nice increments the value for the current shell. If no increment is

specified, nice
sets the nice value to 4. The range of nice values is from -20 through 19. Values of n outside this
range set the value to the lower, or to the higher boundary, respectively.

+n
    Increment the process priority value by n.

-n
    Decrement by n. This argument can be used only by the super-user.

nohup [command]
Run command with HUPs ignored. With no arguments, ignore HUPs throughout the remainder of a
script. When given, command is always run in a subshell, and the restrictions placed on commands
in simple if commands apply. All processes detached with & are effectively nohup'd.

notify [%job] ...
Notify the user asynchronously when the status of the current, or of specified jobs, changes.

onintr [- | label]
Control the action of the shell on interrupts. With no arguments, onintr restores the default action of
the shell on interrupts. (The shell terminates shell scripts and returns to the terminal command input
level). With the - argument, the shell ignores all interrupts. With a label argument, the shell executes
a goto label when an interrupt is received or a child process terminates because it was interrupted.

popd [+n]
Pop the directory stack, and cds to the new top directory. The elements of the directory stack are
numbered from 0 starting at the top.

+n
    Discard the n'th entry in the stack.

pushd [+n | dir]
Push a directory onto the directory stack. With no arguments, exchange the top two elements.

+n
    Rotate the n'th entry to the top of the stack and cd to it. dir Push the current working
    directory onto the stack and change to dir.

rehash
    Recompute the internal hash table of the contents of directories listed in the path variable to
    account for new commands added.

repeat count command
Repeat command count times command is subject to the same restrictions as with the

one-line if
statement.

set [var [= value]]
set var[n] = word
With no arguments, set displays the values of all shell variables. Multiword values are displayed as a
parenthesized list. With the var argument alone, set assigns an empty (null) value to the variable var.
With arguments of the form var = value set assigns value to var, where value is one of:

word
A single word (or quoted string). (wordlist) A space-separated list of words

enclosed in parentheses.

Values are command and filename expanded before being assigned. The form set var[n] = word
replaces the n'th word in a multiword value with word.

setenv [VAR [word]]
With no arguments, setenv displays all environment variables. With the VAR argument sets the
environment variable VAR to have an empty (null) value. (By convention, environment variables are
normally given upper-case names.) With both VAR and word arguments setenv sets the
environment variable NAME to the value word, which must be either a single word or a quoted
string. The most commonly used environment variables, USER, TERM, and PATH, are automatically imported to and exported from the csh variables user, term, and path; there is no need
to use setenv for these. In addition, the shell sets the PWD environment variable from the csh
variable cwd whenever the latter changes.

shift [variable]
The components of argv, or variable, if supplied, are shifted to the left, discarding the first
component. It is an error for the variable not to be set, or to have a null value.

source [-h] name
Reads commands from name. source commands may be nested, but if they are nested too deeply
the shell may run out of file descriptors. An error in a sourced file at any level terminates all nested
source commands.

-h
Place commands from the file name on the history list without executing them.

stop [%job] ...
Stop the current or specified background job.

suspend Stop the shell in its tracks, much as if it had been sent a stop signal with ^Z.

This is most
often used to stop shells started by su.

switch (string)
case label:
...
breaksw
... default:
...
breaksw

endsw
Each label is successively matched, against the specified string, which is first command and
filename expanded. The file metacharacters *, ? and [...] may be used in the case labels,
which are variable expanded. If none of the labels match before a "default" label is found,
execution begins after the default label. Each case statement and the default statement must
appear at the beginning of a line. The command breaksw continues execution after the endsw.
Otherwise control falls through subsequent case and default statements as with C. If no label
matches and there is no default, execution continues after the endsw.

time [command]
With no argument, print a summary of time used by this C shell and its children. With an optional
command, execute command and print a summary of the time it uses.

umask [value]
Display the file creation mask. With value set the file creation mask. value is given in octal, and is
XORed with the permissions of 666 for files and 777 for directories to arrive at the permissions for
new files. Common values include 002, giving complete access to the group, and read (and
directory search) access to others, or 022, giving read (and directory search) but not write
permission to the group and others.

unalias pattern
Discard aliases that match (filename substitution) pattern. All aliases are removed by unalias *.

unhash
Disable the internal hash table.

unlimit [-h] [resource]
Remove a limitation on resource. If no resource is specified, then all resource limitations are
removed. See the description of the limit command for the list of resource names.

-h
Remove corresponding hard limits. Only the super-user may do this.

unset pattern
Remove variables whose names match (filename substitution) pattern. All variables are removed by
`unset *'; this has noticeably distasteful side effects.

unsetenv variable
Remove variable from the environment. Pattern matching, as with unset is not performed.

wait
    Wait for background jobs to finish (or for an interrupt) before prompting.

while (expr)
...

end
    While expr is true (evaluates to non-zero), repeat commands between the while and the
    matching end statement. break and continue may be used to terminate or continue the loop
    prematurely. The while and end must appear alone on their input lines. If the shell's input is a
    terminal, it prompts for commands with a question-mark until the end command is entered
    and then performs the commands in the loop.

%[job] [&]
Bring the current or indicated job to the foreground. With the ampersand, continue running job in
the background.

@ [var =expr]
@ [var[n] =expr]
With no arguments, display the values for all shell variables. With arguments, the variable var, or the
n'th word in the value of var, to the value that expr evaluates to. (If [n] is supplied, both var and its
n'th component must already exist.)

If the expression contains the characters >, <, & or |, then at least this part of expr must be placed
within parentheses.

The operators *=, +=, etc., are available as in C. The space separating the name from the
assignment operator is optional. Spaces are, however, mandatory in separating components of expr
that would otherwise be single words.

Special postfix operators, ++ and -- increment or decrement name, respectively.
lpq: lpq displays the contents of a printer queue. It reports the status of jobs specified by job#, or all
jobs owned by the user specified by username. lpq reports on all jobs in the default printer queue
when invoked with no arguments.

For each print job in the queue, lpq reports the user's name, current position, the names of input files
comprising the job, the job number (by which it is referred to when using lprm(1)) and the total size
in bytes. Normally, only as much information as will fit on one line is displayed. Jobs are normally
queued on a first-in-first-out basis. Filenames comprising a job may be unavailable, such as when lpr
is used at the end of a pipeline; in such cases the filename field indicates ``(standard input)''.

If lpq warns that there is no daemon present (that is, due to some malfunction), the lpc(8) command
can be used to restart a printer daemon.
-P printer
    Display information about the queue for the specified printer. In the absence of the -P
    option, the queue to the printer specified by the PRINTER variable in the environment is
    used. If the PRINTER variable isn't set, the queue for the default printer is used.

-l
    Display queue information in long format; includes the name of the host from which the job
    originated.

+[interval]
    Display the spool queue periodically until it empties. This option clears the terminal screen
    before reporting on the queue. If an interval is supplied, lpq sleeps that number of seconds in
    between reports.
lpr: lpr creates a printer job in a spooling area for subsequent printing as facilities become available.
Each printer job consists of a control file and one or more data files. The data files are copies of (or,
with -s , symbolic links to) each filename you specify. The spool area is managed by the line printer
daemon, lpd(8). Jobs that specify a printer on a remote machine are forwarded by lpd.

lpr reads from the standard input if no files are specified.
-Printer
    Send output to the named printer. Otherwise send output to the printer named in the
    PRINTER environment variable, or to the default printer, lp.

-#copies
    Produce the number of copies indicated for each named file. For example:

example% lpr -#3 index.c lookup.c

produces three copies of index.c, followed by three copies of lookup.c. On the other hand,

example% cat index.c lookup.c | lpr -#3

generates three copies of the concatenation of the files.

-Cclass
Print class as the job classification on the burst page. For example,

example% lpr -C Operations new.index.c

replaces the system name (the name returned by hostname) with "Operations" on the burst page,
and prints the file new.index.c.

-Jjob
Print job as the job name on the burst page. Normally, lpr uses the first file's name.

-Ttitle
Use title instead of the file name for the

title used by pr(1V).

-i[indent]
Indent output indent SPACE characters. Eight SPACE characters is the default. The indent
is passed to the input filter. If no input filter is present, this option is ignored. -1 font
-2 font
-3 font

-4 font
Mount the specified font on font position 1, 2, 3 or 4. The daemon will construct a .railmag
file in the spool directory that indicates the mount by referencing /usr/lib/vfont/font.

-wcols
Use cols as the page width for pr.

-r
Remove the file upon completion of spooling, or upon completion of printing with the -s
option.

-m
Send mail upon completion.

-h
Suppress printing the burst page.

-s
Create a symbolic link from the spool area to the data files rather than trying to copy them (so
large files can be printed). This means the data files should not be modified or removed until
they have been printed. This option can be used to avoid truncating files larger than the
maximum given in the mx capability of the printcap(5) entry. -s only prevents copies of local
files from being made. Jobs from remote hosts are copied anyway. -s only works

with named
data files; if the lpr command is at the end of a pipeline, the data is copied to the spool.

filter-option The following single letter options notify the line printer spooler that the files are not
standard text files. The spooling daemon will use the appropriate filters to print the data accordingly.

-p
Use pr to format the files (lpr -p is very much like `pr | lpr'). -l Print control characters and
suppress page breaks. -t The files contain troff(1) (cat phototypesetter) binary data. -n The
files contain data from ditroff (device independent troff). -d The files contain data from tex
(DVI format from Stanford). -g The files contain standard plot data as produced by the
plot(3X) routines (see also plot(1G) for the filters used by the printer spooler). -v The files
contain a raster image, see rasterfile(5). The printer must support an appropriate imaging
model such as PostScript in order to print the image. -c The files contain data produced by
cifplot. -f Interpret the first character of each line as a standard FORTRAN carriage control
character.

If no filter-option is given (and the printer can interpret PostScript), the string `%!' as the first two
characters of a file indicates that it contains PostScript commands.

These filter options offer a standard user interface, and all options may not be available for, nor
applicable to, all printers.
lprm: lprm removes a job or jobs from a printer's spooling queue. Since the spool directory is protected
from users, using lprm is normally the only method by which a user can remove a job.

Without any arguments, lprm deletes the job that is currently active, provided that the user who
invoked lprm owns that job.

When the super-user specifies a username, lprm removes all jobs belonging to that user.

You can remove a specific job by supplying its job number as an argument, which you can obtain
using lpq(1). For example:

example% lpq -Phost
host is ready and printing
Rank Owner Job Files Total Size active wendy 385 standard input 35501 bytes example
% lprm
-Phost 385

lprm reports the names of any files it removes, and is silent if there are no applicable jobs to remove.

lprm kills the active printer daemon, if necessary, before removing spooled jobs; it restarts the
daemon when through.
-Pprinter
    Specify the queue associated with a specific printer. Otherwise the value of the PRINTER
    variable in the environment is used. If this variable is unset, the queue for the default printer is
    used.

-
    Remove all jobs owned by you. If invoked by the super-user, all jobs in the spool are
    removed. (Job ownership is determined by the user's login name and host name on the
    machine where the lpr command was invoked).
ls: -a, --all
List all files in directories, including all files that start with `.'.

-b, --escape
    Quote nongraphic characters in file names using alphabetic and octal backslash sequences
    like those used in C.

-c, --time=ctime, --time=status
    Sort directory contents according to the files' status change time instead of the modification
    time. If the long listing format is being used, print the status change time instead of the
    modification time.

-d, --directory
    List directories like other files, rather than listing their contents.

-f
    Do not sort directory contents; list them in whatever order they are stored on the disk. The
    same as enabling -a and -U and disabling -l, -s, and -t.

--full-time
    List times in full, rather than using the standard abbreviation heuristics.

-g
    Ignored; for Unix compatibility.

-i, --inode
    Print the index number of each file to the left of the file name.

-k, --kilobytes
    If file sizes are being listed, print them in kilobytes. This overrides the environment variable
    POSIXLY_CORRECT.

-l, --format=long, --format=verbose

In addition to the name of each file, print the file type, permissions, number of hard links,
owner name, group name, size in bytes, and timestamp (the modification time unless other
times are selected). For files with a time that is more than 6 months old or more than 1 hour
into the future, the timestamp contains the year instead of the time of day.

-m, --format=commas
List files horizontally, with as many as will fit on each line, separated by commas.

-n, --numeric-uid-gid
List the numeric UID and GID instead of the names.

-p
Append a character to each file name indicating the file type.

-q, --hide-control-chars
Print question marks instead of nongraphic characters in file names.

-r, --reverse
Sort directory contents in reverse order.

-s, --size
Print the size of each file in 1K blocks to the left of the file name. If the environment variable
POSIXLY_CORRECT is set, 512-byte blocks are used instead.

-t, --sort=time

Sort directory contents by timestamp instead of alphabetically, with the newest files listed first.

-u, --time=atime, --time=access, --time=use
Sort directory contents according to the files' last access time instead of the modification time.
If the long listing format is being used, print the last access time instead of the modification
time.

-x, --format=across, --format=horizontal
List the files in columns, sorted horizontally.

-A, --almost-all
List all files in directories, except for `.' and `..'.

-B, --ignore-backups
Do not list files that end with `~', unless they are given on the command line.

-C, --format=vertical
List files in columns, sorted vertically.

-F, --classify
Append a character to each file name indicating the file type. For regular files that are
executable, append a `*'. The file type indicators are `/' for directories, `@' for

symbolic
links, `|' for FIFOs, `=' for sockets, and nothing for regular files.

-G, --no-group
Inhibit display of group information in a long format directory listing.

-L, --dereference
List the files linked to by symbolic links instead of listing the contents of the links.

-N, --literal
Do not quote file names.

-Q, --quote-name
Enclose file names in double quotes and quote nongraphic characters as in C.

-R, --recursive
List the contents of all directories recursively.

-S, --sort=size
Sort directory contents by file size instead of alphabetically, with the largest files listed first.

-U, --sort=none

Do not sort directory contents; list them in whatever order they are stored on the disk. This option is
not called -f because the Unix ls -f option also enables -a and disables -l, -s, and -t. It seems
useless and ugly to group those unrelated things together in one option. Since this option doesn't do
that, it has a different name.

-X, --sort=extension
Sort directory contents alphabetically by file extension (characters after the last `.'); files with
no extension are sorted first.

-1, --format=single-column
List one file per line.

-w, --width cols
Assume the screen is cols columns wide. The default is taken from the terminal driver if
possible; otherwise the environment variable COLUMNS is used if it is set; otherwise the
default is 80.

-T, --tabsize cols
Assume that each tabstop is cols columns wide. The default is 8.

-I, --ignore pattern
Do not list files whose names match the shell pattern unless they are given on the
command line. As in the shell, an initial `.' in a filename does not match a wildcard at the start
of pattern.

--color, --colour, --color=yes, --colour=yes
    Colorize the names of files depending on the type of file. See DISPLAY COLORIZATION
    below.

--color=tty, --colour=tty
    Same as --color but only if standard output is a terminal. This is very useful for shell scripts
    and command aliases, especially if your favourite pager does not support color control codes.

--color=no, --colour=no
    Disables colorization. This is the default. Provided to override a previous color option.

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

--version
    Print version information on standard output then exit successfully.

DISPLAY COLORIZATION
When using the --color option, this version of ls will colorize the file names printed according to the
name and type of file. By default, this colorization is by type only, and the codes used are ISO 6429
(ANSI) compliant.

You can override the default colors by defining the environment variable LS_COLORS (or
LS_COLOURS). The format of this variable is reminiscent of the termcap(5) file format; a
colon-separated list of expressions of the form "xx=string", where "xx" is a two-character variable
name. The variables with their associated defaults are:

no
    0 Normal (non-filename) text

fi
    0 Regular file di 32 Directory

ln
    36 Symbolic link

pi
    31 Named pipe (FIFO) so 33 Socket

bd
    44;37 Block device

cd
    44;37 Character device

ex
    35 Executable file

mi
    (none) Missing file (defaults to fi)

or
    (none) Orphaned symbolic link (defaults to ln)

lc
    \e[Left code

rc
    m Right code

ec
    (none) End code (replaces lc+no+rc)

You only need to include the variables you want to change from the default.

File names can also be colorized based on filename extension. This is specified in the LS_COLORS
variable using the syntax "*ext=string". For example, using ISO 6429 codes, to color all C-language
source files blue you would specify "*.c=34". This would color all files ending in .c in blue (34)
color.

Control characters can be written either in C-style \escaped notation, or in stty-like ^-notation. The
C-style notation adds \e for Escape, _ for a normal space character, and \? for Delete. In addition,
the \ escape character can be used to override the default interpretation of \, ^, : and =.

Each file will be written as <lc> <color code> <rc> <filename> <ec>. If the <ec> code is
undefined, the sequence <lc> <no> <rc> will be used instead. This is generally more convenient to
use, but less general. The left, right and end codes are provided so you don't have to type common
parts over and over again and to support weird terminals; you will generally not need to change
them at all unless your terminal does not use ISO 6429 color sequences but a different system.

If your terminal does use ISO 6429 color codes, you can compose the type codes (i.e. all except
the lc, rc, and ec codes) from numerical commands separated by semicolons. The most common
commands are:

0
    to restore default color

1
    for brighter colors

4
: for underlined text

5
: for flashing text

30
: for black foreground

31
: for red foreground

32
: for green foreground

33
: for yellow (or brown) foreground

34
: for blue foreground

35
: for purple foreground

36
: for cyan foreground

37
: for white (or gray) foreground

40
: for black background

41
: for red background

42
: for green background

43
: for yellow (or brown) background

44
: for blue background

45
: for purple background

46
: for cyan background

47
: for white (or gray) background

Not all commands will work on all systems or display devices.

A few terminal programs do not recognize the default end code properly. If all text gets colorized
after you do a directory listing, try changing the no and fi codes from 0 to the numerical codes for
your standard fore- and background colors.
mail: mail is a comfortable, flexible, interactive program for composing, sending and receiving electronic
messages. While reading messages, mail provides you with commands to browse, display, save,
delete, and respond to messages. While sending mail, mail allows editing and reviewing of messages
being composed, and the inclusion of text from files or other messages.

Incoming mail is stored in the system mailbox for each user. This is a file named after the user in
/var/spool/mail. mail normally looks in this file for incoming messages, but you can use the MAIL
environment variable to have it look in a different file. When you read a message, it is marked to be
moved to a secondary file for storage. This secondary file, called the mbox, is normally the file
mbox in your home directory. This file can also be changed by setting the MBOX environment
variable. Messages remain in the mbox file until deliberately removed.
If no recipient is specified, mail attempts to read messages from the system mailbox.

-d
    Turn on debugging output. (Neither particularly interesting nor recommended.)

-e
    Test for presence of mail. If there is no mail, mail prints nothing and exits (with a successful
    return code).

-F
    Record the message in a file named after the first recipient. Override the record variable, if
    set.

-H
    Print header summary only.

-i
    Ignore interrupts (as with the ignore variable).

-n
    Do not initialize from the system default Mail.rc file.

-N
    Do not print initial header summary.

-U
    Convert uucp style addresses to Internet standards. Overrides the conv environment variable.

-v

Pass the -v flag to sendmail(8).

-f [filename] Read messages from filename instead of
system mailbox. If no filename is specified, the mbox is used.

-f +folder
Use the file folder in the folder directory (same as the folder command). The name of this
directory is listed in the folder variable.

-h number
The number of network "hops" made so far. This is provided for network software to avoid
infinite delivery loops.

-r address
Pass address to network delivery software. All tilde (~) commands are disabled.

-s subject
Set the Subject header field to subject.

-T file
Print the contents of the article-id fields of all messages that were read or deleted on file (for
the use of network news programs if available).

-u user
Read user's system mailbox. This is only effective if user's system mailbox is not read
protected.
man: man displays information from the reference manuals. It can display complete manual pages that you
select by title, or one-line summaries selected either by keyword (-k), or by the name of an
associated file (-f).

A section, when given, applies to the titles that follow it on the command line (up to the next
section, if any). man looks in the indicated section of the manual for those titles. section is either a
digit (perhaps followed by a single letter indicating the type of manual page), or one of the words
new, local, old, or public. The abbreviations n, l, o and p are also allowed. If section is omitted,
man searches all reference sections (giving preference to commands over functions) and prints the
first manual page it finds. If no manual page is located, man prints an error message.

The reference page sources are typically located in the /usr/man/man? directories. Since these
directories are optionally installed, they may not reside on your host; you may have to mount
/usr/man from a host on which they do reside. If there are preformatted, up-to-date versions in
corresponding cat? or fmt? directories, man simply displays or prints those versions. If the

preformatted version of interest is out of date or missing, man reformats it prior to display. If
directories for the preformatted versions are not provided, man reformats a page whenever it is
requested; it uses a temporary file to store the formatted text during display.

If the standard output is not a terminal, or if the `-' flag is given, man pipes its output through
cat(1V). Otherwise, man pipes its output through more(1) to handle paging and underlining on the
screen.
-t man arranges for the specified manual pages to be troffed to a suitable raster output device (see
troff(1) or vtroff(1)). If both the - and -t flags are given, man updates the troffed versions of each
named title (if necessary), but does not display them.

-M path
    Change the search path for manual pages. path is a colon-separated list of directories that
    contain manual page directory subtrees. For example, /usr/man/u_man:/usr/man/a_man
    makes man search in the standard System V locations. When used with the -k or -f options,
    the -M option must appear first. Each directory in the path is assumed to contain
    subdirectories of the form man[1-8l-p].

-T macro-package
    man uses macro-package rather than the standard -man macros defined in
    /usr/lib/tmac/tmac.an for formatting manual pages.

-k keyword ...
    man prints out one-line summaries from the whatis database (table of contents) that contain
    any of the given keywords. The whatis database is created using the catman(8) command
    with the -w option.

-f filename ...
    man attempts to locate manual pages related to any of the given filenames. It strips the
    leading pathname components from each filename, and then prints one-line summaries
    containing the resulting basename or names. This option also uses the whatis database.
mkdir: mkdir creates directories. Standard entries, `.', for the directory itself, and `..' for its parent, are
made automatically.

The -p flag allows missing parent directories to be created as needed.

With the exception of the set-gid bit, the current umask(2V) setting determines the mode in which
directories are created. The new directory inherits the set-gid bit of the parent directory. Modes
may be modified after creation by using chmod(1V).

mkdir requires write permission in the parent directory.
more: more is a filter that displays the contents of a text file on the terminal, one screenful at a time. It
normally pauses after each screenful, and prints --More-- at the bottom of the screen. more
provides a two-line overlap between screens for continuity. If more is reading from a file rather than
a pipe, the percentage of characters displayed so far is also shown.

more scrolls up to display one more line in response to a RETURN character; it displays another
screenful in response to a SPACE character. Other commands are listed below.

page clears the screen before displaying the next screenful of text; it only provides a one-line overlap
between screens.

more sets the terminal to noecho mode, so that the output can be continuous. Commands that you
type do not normally show up on your terminal, except for the / and ! commands.

If the standard output is not a terminal, more acts just like cat(1V), except that a header is printed
before each file in a series.
-c Clear before displaying. Redrawing the screen instead of scrolling for faster displays. This option
is ignored if the terminal does not have the ability to clear to the end of a line.

-d
    Display error messages rather than ringing the terminal bell if an unrecognized command is
    used. This is helpful for inexperienced users.

-f
    Do not fold long lines. This is useful when lines contain nonprinting characters or escape
    sequences, such as those generated when nroff(1) output is piped through ul(1).

-l
    Do not treat FORMFEED characters (CTRL-D) as "page breaks." If -l is not used, more
    pauses to accept commands after any line containing a ^L character (CTRL-D). Also, if a file
    begins with a FORMFEED, the screen is cleared before the file is printed.

-s
    Squeeze. Replace multiple blank lines with a single blank line. This is helpful when viewing
    nroff(1) output, on the screen.

-u
    Suppress generation of underlining escape sequences. Normally, more handles underlining,
    such as that produced by nroff(1), in a manner appropriate to the terminal. If the terminal can

perform underlining or has a stand-out mode, more supplies appropriate escape sequences as
called for in the text file.

-lines
Display the indicated number of lines in each screenful, rather than the default (the number of
lines in the terminal screen less two).

+linenumber
Start up at linenumber.

+/pattern
Start up two lines above the line containing the regular expression pattern. Note: unlike
editors, this construct should not end with a `/'. If it does, then the trailing slash is taken as a
character in the search pattern.

mv: mv moves files and directories around in the file system. A side effect of mv is to rename a file or
directory. The three major forms of mv are shown in the synopsis above.

The first form of mv moves (changes the name of) filename1 to filename2. If filename2 already
exists, it is removed before filename1 is moved. If filename2 has a mode which forbids writing, mv
prints the mode (see chmod(2V)) and reads the standard input to obtain a line; if the line begins with
y, the move takes place, otherwise mv exits.

The second form of mv moves (changes the name of) directory1 to directory2, only if directory2
does not already exist if it does, the third form applies.

The third form of mv moves one or more filenames (may also be directories) with their original
names, into the last directory in the list.

mv refuses to move a file or directory onto itself.

-
Interpret all the following arguments to mv as file names. This allows file names starting with
minus.

-f
Force. Override any mode restrictions and the -i option. The -f option also suppresses any
warning messages about modes which would potentially restrict overwriting.

-i
Interactive mode. mv displays the name of the file or directory followed by a question mark
whenever a move would replace an existing file or directory. If you type a line starting with y,
mv moves the specified file or directory, otherwise mv does nothing with that file or directory.

passwd: passwd changes (or installs) a password, login shell (-s option), or full name (-f option) associated
with the user username (your own by default). chsh is equivalent to passwd with the -s option, and
chfn is equivalent to passwd with the -f option.

Use `passwd -y' or yppasswd(1) to change your password in the Network Information Service
(NIS). This will not affect your local password, or your password on any remote machines on which
you have accounts. passwd calls yppasswd automatically if you do not have an entry in the local
passwd file, and the -l option is not specified.

When changing a password, passwd prompts for the old password and then for the new one. You
must supply both, and the new password must be typed twice to forestall mistakes.

If password aging is enabled, the first time an ordinary user enters the new password passwd
checks to see if the old password has "aged" sufficiently. Password "aging" is the amount of time
(usually a certain number of days) that must elapse between password changes. If "aging" is
insufficient the new password is rejected and passwd terminates.

New passwords should be at least five characters long, if they combine upper-case and lower-case
letters, or at least six characters long if in monocase. Users that persist in entering shorter passwords
are compromising their own security. The number of significant characters in a password is eight,
although longer passwords will be accepted.

Only the owner of the name or the super-user may change a password; the owner must prove he
knows the old password. The super-user can change any password and is not forced to comply
with password aging requirements.

When changing a login shell, passwd displays the current login shell and then prompts for the new
one. The new login shell must be one of the approved shells listed in /etc/shells unless you are the
super-user. If /etc/shells does not exist, the only shells that may be specified are /bin/sh and /bin/csh.

The super-user may change anyone's login shell; normal users may only change their own login shell.

When changing a full name, passwd displays the current full name, enclosed between brackets, and
prompts for a new full name. If you type a RETURN, the full name is not changed. If the full name is
to be made blank, you must type the word "none".

The super-user may change anyone's full name; normal users may only change their own.
-a Display the name and aging information for all users. Can only be invoked by the super-user.

-f
    Change the full name.

-l
    Change the local password, login shell, or full name. If username exists in the local passwd
    file, this is the default.

-s
    Change the login shell.

-y
    Change passwd, login shell, or full name in the NIS database.

-d [username]
    Display the name and aging information for the caller or the user specified if the invoker has
    the right privileges.

-e username
    Expire the password for the user name specified. Can only be invoked by the super-user.

-F filename
    Treat filename as the password file.

-n numdays username
    Set the maturity time of the password for username. Passwords that have not "aged" enough
    cannot be changed. Can only be set by the super-user.

-x numdays username

Set the expiration time of the password for username. Can only be set by the super-user.
ps: ps displays information about processes. Normally, only those processes that are running with your
effective user ID and are attached to a controlling terminal (see termio(4)) are shown. Additional
categories of processes can be added to the display using various options. In particular, the -a
option allows you to include processes that are not owned by you (that do not have your user ID),
and the -x option allows you to include processes without control terminals. When you specify both
-a and -x, you get processes owned by anyone, with or without a control terminal. The -r option
restricts the list of processes printed to "running" processes: runnable processes, those in page wait,
or those in short-term non-interruptible waits.

ps displays the process ID, under PID; the control terminal (if any), under TT; the cpu time used by
the process so far, including both user and system time), under TIME; the state of the process,
under STAT; and finally, an indication of the COMMAND that is running.

The state is given by a sequence of four letters, for example, `RWNA'.

First letter
    indicates the run ability of the process:

R
    Runnable processes.

T
    Stopped processes.

P
    Processes in page wait.

D
    Processes in non-interruptible waits; typically short-term waits for disk or NFS I/O.

S
    Processes sleeping for less than about 20 seconds.

I
    Processes that are idle (sleeping longer than about 20 seconds).

Z
    Processes that have terminated and that are waiting for their parent process to do a wait(2V)
    ("zombie" processes).

Second letter indicates whether a process is swapped out; blank Represented as a SPACE
character, in this position indicates that the process is loaded (in memory).

W
    Process is swapped out.

>
    Process has specified a soft limit on memory requirements and has exceeded that limit; such a
    process is (necessarily) not swapped.

Third letter indicates whether a process is running with altered CPU scheduling priority (nice(1)):
blank Represented as a SPACE character, in this position indicates that the process is running
without special treatment.

N
    The process priority is reduced,

<

The process priority has been raised artificially.

Fourth letter indicates any special treatment of the process for virtual memory replacement. The
letters correspond to options to the advise(2) system call. Currently the possibilities are:
blank Represented as a SPACE character, in this position stands for VA_NORM.

A
Stands for VA_ANOM. An A typically represents a program which is doing garbage collection.

S
Stands for VA_SEQL. An S is typical of large image processing programs that are using
virtual memory to sequentially address voluminous data.

kernel-name specifies the location of the system namelist. If the -k option is given, c-dump-file tells
ps where to look for the core dump. Otherwise, the core dump is located in the file /vmcore and this
argument is ignored. swap-file gives the location of a swap file other than the default, /dev/drum.
pwd: pwd prints the pathname of the working (current) directory.

If you are using csh(1), you can use the dirs built-in command to do the same job more quickly; but
dirs can give a different answer in the rare case that the current directory or a containing directory
was moved after the shell descended into it. This is because pwd searches back up the directory
tree to report the true pathname, whereas dirs remembers the pathname from the last cd(1)
command. The example below illustrates the differences.

example% cd /usr/wendy/january/reports example% pwd
/usr/wendy/january/reports
example% dirs
~/january/reports
example% mv ~/january ~/february
example% pwd
/usr/wendy/february/reports
example% dirs
~/january/reports
example%

pwd and dirs also give different answers when you change directory through a symbolic link. For
example: example% cd /usr/wendy/january/reports example% pwd
/usr/wendy/january/reports
example% dirs
~/january/reports
example% ls -l /usr/wendy/january
lrwxrwxrwx 1 wendy 17 Jan 30 1983 /usr/wendy/january -> /usr/wendy/1984/jan/
example% cd
/usr/wendy/january

example% pwd
/usr/wendy/1984/jan
example% dirs
/usr/wendy/january

The pathnames of files mounted with the Automounter can also change if the file is not used for a
certain time interval (the default is five minutes).
rm: rm removes (directory entries for) one or more files. If an entry was the last link to the file, the
contents of that file are lost. See ln(1V) for more information about multiple links to files.

To remove a file, you must have write permission in its directory; but you do not need read or write
permission on the file itself. If you do not have write permission on the file and the standard input is a
terminal, rm displays the file's permissions and waits for you to type in a response. If your response
begins with y the file is deleted; otherwise the file is left alone.

rmdir removes each named directory. rmdir only removes empty directories.
-
    Treat the following arguments as filenames `-' so that you can specify filenames starting with a
    minus.

-f
    Force files to be removed without displaying permissions, asking questions or reporting
    errors.

-i
    Ask whether to delete each file, and, under -r, whether to examine each directory.
    Sometimes called the interactive option.

-r
    Recursively delete the contents of a directory, its subdirectories, and the directory itself.
rmdir: rm removes (directory entries for) one or more files. If an entry was the last link to the file, the
contents of that file are lost. See ln(1V) for more information about multiple links to files.

To remove a file, you must have write permission in its directory; but you do not need read or write
permission on the file itself. If you do not have write permission on the file and the standard input is a
terminal, rm displays the file's permissions and waits for you to type in a response. If your response
begins with y the file is deleted; otherwise the file is left alone.

rmdir removes each named directory. rmdir only removes empty directories.
-
    Treat the following arguments as filenames `-' so that you can specify filenames starting with a

minus.

-f
Force files to be removed without displaying permissions, asking questions or reporting
errors.

-i
Ask whether to delete each file, and, under -r, whether to examine each directory.
Sometimes called the interactive option.

-r
Recursively delete the contents of a directory, its subdirectories, and the directory itself.
spell: spell collects words from the named files, and looks them up in a hashed spelling list. Words that do
not appear in the list, or cannot be derived from those that do appear by applying certain inflections,
prefixes or suffixes, are displayed on the standard output.

If there are no filename arguments, words to check are collected from the standard input. spell
ignores most troff(1), tbl(1), and eqn(1) constructs. Copies of all output words are accumulated in
the history file, and a stop list filters out misspellings (for example, their=thy-y+ier) that would
otherwise pass.

By default, spell (like deroff(1)) follows chains of included files (.so and .nx troff(1) requests),
unless the names of such included files begin with /usr/lib.

If a +local_file argument is specified, words found in local_file are removed from spell's output.
local_file is the name of a user-provided file that contains a sorted list of words, one per line. With
this option, the user can specify a set of words that are correct spellings (in addition to spell's own
spelling list) for each job.

The standard spelling list is based on many sources, and while more haphazard than an ordinary
dictionary, is also more effective in respect to proper names and popular technical words. Coverage
of the specialized vocabularies of biology, medicine and chemistry is light.

Three programs help maintain and check the hash lists used by spell:

hashmake
Reads a list of words from the standard input and writes the corresponding nine-digit hash

code on the standard output.

spellin
Reads n hash codes from the standard input and

writes a compressed spelling list on the standard output.

hashcheck Reads a compressed spelling_list and recreates the nine-digit hash codes for all the
words in it; it writes these codes on the standard output.

-b
Check British spelling. Besides preferring "centre", "colour", "programme", "speciality",
"travelled", and so on, this option insists upon -ise in words like standardize, despite what
Fowler and the OED say.

-l
Follow the chains of all included files.

-v
Print all words not literally in the spelling list, as well as plausible derivations from spelling list
words.

-x
Print every plausible stem with `=' for each word.

-d hlist
Use the file hlist as the hashed spelling list.

-h spellhist
Place misspelled words with a user/date stamp in file spellhist.

-s hstop
Use hstop as the hashed stop list.