NAME

mkdir – make directories

SYNOPSIS

mkdir [-pv] [-m mode] directory\_name ...

DESCRIPTION

The mkdir utility creates the directories named as operands, in the order

specified, using mode “rwxrwxrwx” (0777) as modified by the current

umask(2).

The options are as follows:

-m mode Set the file permission bits of the final created

directory to the specified mode. The mode argument can be

in any of the formats specified to the chmod(1) command.

If a symbolic mode is specified, the operation characters

‘+’ and ‘-’ are interpreted relative to an initial mode of

“a=rwx”.

-p Create intermediate directories as required. If this

option is not specified, the full path prefix of each

operand must already exist. On the other hand, with this

option specified, no error will be reported if a directory

given as an operand already exists. Intermediate

directories are created with permission bits of

“rwxrwxrwx” (0777) as modified by the current umask, plus

write and search permission for the owner.

-v Be verbose when creating directories, listing them as they

are created.

The user must have write permission in the parent directory.

EXIT STATUS

The mkdir utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Create a directory named foobar:

$ mkdir foobar

Create a directory named foobar and set its file mode to 700:

$ mkdir -m 700 foobar

Create a directory named cow/horse/monkey, creating any non-existent

intermediate directories as necessary:

$ mkdir -p cow/horse/monkey

COMPATIBILITY

The -v option is non-standard and its use in scripts is not recommended.

SEE ALSO

rmdir(1)

STANDARDS

The mkdir utility is expected to be IEEE Std 1003.2 (“POSIX.2”)

compatible.

HISTORY

A mkdir command appeared in Version 1 AT&T UNIX.

NAME

echo – write arguments to the standard output

SYNOPSIS

echo [-n] [string ...]

DESCRIPTION

The echo utility writes any specified operands, separated by single blank

(‘ ’) characters and followed by a newline (‘\n’) character, to the

standard output.

The following option is available:

-n Do not print the trailing newline character.

The end-of-options marker -- is not recognized and written literally.

The newline may also be suppressed by appending ‘\c’ to the end of the

string, as is done by iBCS2 compatible systems. Note that the -n option

as well as the effect of ‘\c’ are implementation-defined in IEEE Std

1003.1-2001 (“POSIX.1”) as amended by Cor. 1-2002. For portability, echo

should only be used if the first argument does not start with a hyphen

(‘-’) and does not contain any backslashes (‘\’). If this is not

sufficient, printf(1) should be used.

Most shells provide a builtin echo command which tends to differ from

this utility in the treatment of options and backslashes. Consult the

builtin(1) manual page.

EXIT STATUS

The echo utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

builtin(1), csh(1), printf(1), sh(1)

STANDARDS

The echo utility conforms to IEEE Std 1003.1-2001 (“POSIX.1”) as amended

by Cor. 1-2002.

NAME

cat – concatenate and print files

SYNOPSIS

cat [-belnstuv] [file ...]

DESCRIPTION

The cat utility reads files sequentially, writing them to the standard

output. The file operands are processed in command-line order. If file

is a single dash (‘-’) or absent, cat reads from the standard input. If

file is a UNIX domain socket, cat connects to it and then reads it until

EOF. This complements the UNIX domain binding capability available in

inetd(8).

The options are as follows:

-b Number the non-blank output lines, starting at 1.

-e Display non-printing characters (see the -v option), and display

a dollar sign (‘$’) at the end of each line.

-l Set an exclusive advisory lock on the standard output file

descriptor. This lock is set using fcntl(2) with the F\_SETLKW

command. If the output file is already locked, cat will block

until the lock is acquired.

-n Number the output lines, starting at 1.

-s Squeeze multiple adjacent empty lines, causing the output to be

single spaced.

-t Display non-printing characters (see the -v option), and display

tab characters as ‘^I’.

-u Disable output buffering.

-v Display non-printing characters so they are visible. Control

characters print as ‘^X’ for control-X; the delete character

(octal 0177) prints as ‘^?’. Non-ASCII characters (with the high

bit set) are printed as ‘M-’ (for meta) followed by the character

for the low 7 bits.

EXIT STATUS

The cat utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

The command:

cat file1

will print the contents of file1 to the standard output.

The command:

cat file1 file2 > file3

will sequentially print the contents of file1 and file2 to the file

file3, truncating file3 if it already exists. See the manual page for

your shell (e.g., sh(1)) for more information on redirection.

The command:

cat file1 - file2 - file3

will print the contents of file1, print data it receives from the

standard input until it receives an EOF (‘^D’) character, print the

contents of file2, read and output contents of the standard input again,

then finally output the contents of file3. Note that if the standard

input referred to a file, the second dash on the command-line would have

no effect, since the entire contents of the file would have already been

read and printed by cat when it encountered the first ‘-’ operand.

SEE ALSO

head(1), more(1), pr(1), sh(1), tail(1), vis(1), zcat(1), fcntl(2),

setbuf(3)

Rob Pike, “UNIX Style, or cat -v Considered Harmful”, USENIX Summer

Conference Proceedings, 1983.

STANDARDS

The cat utility is compliant with the IEEE Std 1003.2-1992 (“POSIX.2”)

specification.

The flags [-belnstv] are extensions to the specification.

HISTORY

A cat utility appeared in Version 1 AT&T UNIX. Dennis Ritchie designed

and wrote the first man page. It appears to have been for cat.

BUGS

Because of the shell language mechanism used to perform output

redirection, the command “cat file1 file2 > file1” will cause the

original data in file1 to be destroyed!

The cat utility does not recognize multibyte characters when the -t or -v

option is in effect.

NAME

touch – change file access and modification times

SYNOPSIS

touch [-A [-][[hh]mm]SS] [-achm] [-r file] [-t [[CC]YY]MMDDhhmm[.SS]]

[-d YYYY-MM-DDThh:mm:SS[.frac][tz]] file ...

DESCRIPTION

The touch utility sets the modification and access times of files. If

any file does not exist, it is created with default permissions.

By default, touch changes both modification and access times. The -a and

-m flags may be used to select the access time or the modification time

individually. Selecting both is equivalent to the default. By default,

the timestamps are set to the current time. The -d and -t flags

explicitly specify a different time, and the -r flag specifies to set the

times those of the specified file. The -A flag adjusts the values by a

specified amount.

The following options are available:

-A Adjust the access and modification time stamps for the file by

the specified value. This flag is intended for use in modifying

files with incorrectly set time stamps.

The argument is of the form “[-][[hh]mm]SS” where each pair of

letters represents the following:

- Make the adjustment negative: the new time stamp is

set to be before the old one.

hh The number of hours, from 00 to 99.

mm The number of minutes, from 00 to 59.

SS The number of seconds, from 00 to 59.

The -A flag implies the -c flag: if any file specified does not

exist, it will be silently ignored.

-a Change the access time of the file. The modification time of the

file is not changed unless the -m flag is also specified.

-c Do not create the file if it does not exist. The touch utility

does not treat this as an error. No error messages are displayed

and the exit value is not affected.

-d Change the access and modification times to the specified date

time instead of the current time of day. The argument is of the

form “YYYY-MM-DDThh:mm:SS[.frac][tz]” where the letters represent

the following:

YYYY At least four decimal digits representing the year.

MM, DD, hh, mm, SS

As with -t time.

T The letter T or a space is the time designator.

.frac An optional fraction, consisting of a period or a

comma followed by one or more digits. The number

of significant digits depends on the kernel

configuration and the filesystem, and may be zero.

tz An optional letter Z indicating the time is in UTC.

Otherwise, the time is assumed to be in local time.

Local time is affected by the value of the TZ

environment variable.

-h If the file is a symbolic link, change the times of the link

itself rather than the file that the link points to. Note that

-h implies -c and thus will not create any new files.

-m Change the modification time of the file. The access time of the

file is not changed unless the -a flag is also specified.

-r Use the access and modifications times from the specified file

instead of the current time of day.

-t Change the access and modification times to the specified time

instead of the current time of day. The argument is of the form

“[[CC]YY]MMDDhhmm[.SS]” where each pair of letters represents the

following:

CC The first two digits of the year (the century).

YY The second two digits of the year. If “YY” is

specified, but “CC” is not, a value for “YY”

between 69 and 99 results in a “CC” value of 19.

Otherwise, a “CC” value of 20 is used.

MM The month of the year, from 01 to 12.

DD the day of the month, from 01 to 31.

hh The hour of the day, from 00 to 23.

mm The minute of the hour, from 00 to 59.

SS The second of the minute, from 00 to 60.

If the “CC” and “YY” letter pairs are not specified, the values

default to the current year. If the “SS” letter pair is not

specified, the value defaults to 0.

EXIT STATUS

The touch utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

The obsolescent form of touch, where a time format is specified as the

first argument, is supported. When no -r or -t option is specified,

there are at least two arguments, and the first argument is a string of

digits either eight or ten characters in length, the first argument is

interpreted as a time specification of the form “MMDDhhmm[YY]”.

The “MM”, “DD”, “hh” and “mm” letter pairs are treated as their

counterparts specified to the -t option. If the “YY” letter pair is in

the range 39 to 99, the year is set to 1939 to 1999, otherwise, the year

is set in the 21st century.

SEE ALSO

utimensat(2)

STANDARDS

The touch utility is expected to be a superset of the IEEE Std 1003.2

(“POSIX.2”) specification.

NAME

ls – list directory contents

SYNOPSIS

ls [-ABCFGHILPRSTUWZabcdfghiklmnopqrstuwxy1,] [--color=when] [-D format]

[file ...]

DESCRIPTION

For each operand that names a file of a type other than directory, ls

displays its name as well as any requested, associated information. For

each operand that names a file of type directory, ls displays the names

of files contained within that directory, as well as any requested,

associated information.

If no operands are given, the contents of the current directory are

displayed. If more than one operand is given, non-directory operands are

displayed first; directory and non-directory operands are sorted

separately and in lexicographical order.

The following options are available:

-A Include directory entries whose names begin with a dot (‘.’)

except for . and ... Automatically set for the super-user unless

-I is specified.

-B Force printing of non-printable characters (as defined by

ctype(3) and current locale settings) in file names as \xxx,

where xxx is the numeric value of the character in octal. This

option is not defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-C Force multi-column output; this is the default when output is to

a terminal.

-D format

When printing in the long (-l) format, use format to format the

date and time output. The argument format is a string used by

strftime(3). Depending on the choice of format string, this may

result in a different number of columns in the output. This

option overrides the -T option. This option is not defined in

IEEE Std 1003.1-2008 (“POSIX.1”).

-F Display a slash (‘/’) immediately after each pathname that is a

directory, an asterisk (‘\*’) after each that is executable, an at

sign (‘@’) after each symbolic link, an equals sign (‘=’) after

each socket, a percent sign (‘%’) after each whiteout, and a

vertical bar (‘|’) after each that is a FIFO.

-G Enable colorized output. This option is equivalent to defining

CLICOLOR or COLORTERM in the environment and setting

--color=auto. (See below.) This functionality can be compiled

out by removing the definition of COLORLS. This option is not

defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-H Symbolic links on the command line are followed. This option is

assumed if none of the -F, -d, or -l options are specified.

-I Prevent -A from being automatically set for the super-user. This

option is not defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-L If argument is a symbolic link, list the file or directory the

link references rather than the link itself. This option cancels

the -P option.

-P If argument is a symbolic link, list the link itself rather than

the object the link references. This option cancels the -H and

-L options.

-R Recursively list subdirectories encountered.

-S Sort by size (largest file first) before sorting the operands in

lexicographical order.

-T When printing in the long (-l) format, display complete time

information for the file, including month, day, hour, minute,

second, and year. The -D option gives even more control over the

output format. This option is not defined in IEEE Std

1003.1-2008 (“POSIX.1”).

-U Use time when file was created for sorting or printing. This

option is not defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-W Display whiteouts when scanning directories. This option is not

defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-Z Display each file's MAC label; see maclabel(7). This option is

not defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-a Include directory entries whose names begin with a dot (‘.’).

-b As -B, but use C escape codes whenever possible. This option is

not defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-c Use time when file status was last changed for sorting or

printing.

--color=when

Output colored escape sequences based on when, which may be set

to either always, auto, or never.

always will make ls always output color. If TERM is unset or set

to an invalid terminal, then ls will fall back to explicit ANSI

escape sequences without the help of termcap(5). always is the

default if --color is specified without an argument.

auto will make ls output escape sequences based on termcap(5),

but only if stdout is a tty and either the -G flag is specified

or the COLORTERM environment variable is set and not empty.

never will disable color regardless of environment variables.

never is the default when neither --color nor -G is specified.

For compatibility with GNU coreutils, ls supports yes or force as

equivalent to always, no or none as equivalent to never, and tty

or if-tty as equivalent to auto.

-d Directories are listed as plain files (not searched recursively).

-f Output is not sorted. This option turns on -a. It also negates

the effect of the -r, -S and -t options. As allowed by IEEE Std

1003.1-2008 (“POSIX.1”), this option has no effect on the -d, -l,

-R and -s options.

-g This option has no effect. It is only available for

compatibility with 4.3BSD, where it was used to display the group

name in the long (-l) format output. This option is incompatible

with IEEE Std 1003.1-2008 (“POSIX.1”).

-h When used with the -l option, use unit suffixes: Byte, Kilobyte,

Megabyte, Gigabyte, Terabyte and Petabyte in order to reduce the

number of digits to four or fewer using base 2 for sizes. This

option is not defined in IEEE Std 1003.1-2008 (“POSIX.1”).

-i For each file, print the file's file serial number (inode

number).

-k This has the same effect as setting environment variable

BLOCKSIZE to 1024, except that it also nullifies any -h options

to its left.

-l (The lowercase letter “ell”.) List files in the long format, as

described in the The Long Format subsection below.

-m Stream output format; list files across the page, separated by

commas.

-n Display user and group IDs numerically rather than converting to

a user or group name in a long (-l) output.

-o Include the file flags in a long (-l) output. This option is

incompatible with IEEE Std 1003.1-2008 (“POSIX.1”). See

chflags(1) for a list of file flags and their meanings.

-p Write a slash (‘/’) after each filename if that file is a

directory.

-q Force printing of non-graphic characters in file names as the

character ‘?’; this is the default when output is to a terminal.

-r Reverse the order of the sort.

-s Display the number of blocks used in the file system by each

file. Block sizes and directory totals are handled as described

in The Long Format subsection below, except (if the long format

is not also requested) the directory totals are not output when

the output is in a single column, even if multi-column output is

requested.

-t Sort by descending time modified (most recently modified first).

If two files have the same modification timestamp, sort their

names in ascending lexicographical order. The -r option reverses

both of these sort orders.

Note that these sort orders are contradictory: the time sequence

is in descending order, the lexicographical sort is in ascending

order. This behavior is mandated by IEEE Std 1003.2 (“POSIX.2”).

This feature can cause problems listing files stored with

sequential names on FAT file systems, such as from digital

cameras, where it is possible to have more than one image with

the same timestamp. In such a case, the photos cannot be listed

in the sequence in which they were taken. To ensure the same

sort order for time and for lexicographical sorting, set the

environment variable LS\_SAMESORT or use the -y option. This

causes ls to reverse the lexicographical sort order when sorting

files with the same modification timestamp.

-u Use time of last access, instead of time of last modification of

the file for sorting (-t) or printing (-l).

-w Force raw printing of non-printable characters. This is the

default when output is not to a terminal. This option is not

defined in IEEE Std 1003.1-2001 (“POSIX.1”).

-x The same as -C, except that the multi-column output is produced

with entries sorted across, rather than down, the columns.

-y When the -t option is set, sort the alphabetical output in the

same order as the time output. This has the same effect as

setting LS\_SAMESORT. See the description of the -t option for

more details. This option is not defined in IEEE Std 1003.1-2001

(“POSIX.1”).

-1 (The numeric digit “one”.) Force output to be one entry per line.

This is the default when output is not to a terminal.

-, (Comma) When the -l option is set, print file sizes grouped and

separated by thousands using the non-monetary separator returned

by localeconv(3), typically a comma or period. If no locale is

set, or the locale does not have a non-monetary separator, this

option has no effect. This option is not defined in IEEE Std

1003.1-2001 (“POSIX.1”).

The -1, -C, -x, and -l options all override each other; the last one

specified determines the format used.

The -c, -u, and -U options all override each other; the last one

specified determines the file time used.

The -S and -t options override each other; the last one specified

determines the sort order used.

The -B, -b, -w, and -q options all override each other; the last one

specified determines the format used for non-printable characters.

The -H, -L and -P options all override each other (either partially or

fully); they are applied in the order specified.

By default, ls lists one entry per line to standard output; the

exceptions are to terminals or when the -C or -x options are specified.

File information is displayed with one or more ⟨blank⟩s separating the

information associated with the -i, -s, and -l options.

The Long Format

If the -l option is given, the following information is displayed for

each file: file mode, number of links, owner name, group name, MAC label,

number of bytes in the file, abbreviated month, day-of-month file was

last modified, hour file last modified, minute file last modified, and

the pathname.

If the modification time of the file is more than 6 months in the past or

future, and the -D or -T are not specified, then the year of the last

modification is displayed in place of the hour and minute fields.

If the owner or group names are not a known user or group name, or the -n

option is given, the numeric ID's are displayed.

If the file is a character special or block special file, the device

number for the file is displayed in the size field. If the file is a

symbolic link the pathname of the linked-to file is preceded by “->”.

The listing of a directory's contents is preceded by a labeled total

number of blocks used in the file system by the files which are listed as

the directory's contents (which may or may not include . and .. and other

files which start with a dot, depending on other options).

The default block size is 512 bytes. The block size may be set with

option -k or environment variable BLOCKSIZE. Numbers of blocks in the

output will have been rounded up so the numbers of bytes is at least as

many as used by the corresponding file system blocks (which might have a

different size).

The file mode printed under the -l option consists of the entry type and

the permissions. The entry type character describes the type of file, as

follows:

- Regular file.

b Block special file.

c Character special file.

d Directory.

l Symbolic link.

p FIFO.

s Socket.

w Whiteout.

The next three fields are three characters each: owner permissions, group

permissions, and other permissions. Each field has three character

positions:

1. If r, the file is readable; if -, it is not readable.

2. If w, the file is writable; if -, it is not writable.

3. The first of the following that applies:

S If in the owner permissions, the file is not

executable and set-user-ID mode is set. If in the

group permissions, the file is not executable and

set-group-ID mode is set.

s If in the owner permissions, the file is

executable and set-user-ID mode is set. If in the

group permissions, the file is executable and

setgroup-ID mode is set.

x The file is executable or the directory is

searchable.

- The file is neither readable, writable,

executable, nor set-user-ID nor set-group-ID mode,

nor sticky. (See below.)

These next two apply only to the third character in the last

group (other permissions).

T The sticky bit is set (mode 1000), but not execute

or search permission. (See chmod(1) or

sticky(7).)

t The sticky bit is set (mode 1000), and is

searchable or executable. (See chmod(1) or

sticky(7).)

The next field contains a plus (‘+’) character if the file has an ACL, or

a space (‘ ’) if it does not. The ls utility does not show the actual

ACL; use getfacl(1) to do this.

ENVIRONMENT

The following environment variables affect the execution of ls:

BLOCKSIZE If this is set, its value, rounded up to 512 or down

to a multiple of 512, will be used as the block size

in bytes by the -l and -s options. See The Long

Format subsection for more information.

CLICOLOR Use ANSI color sequences to distinguish file types.

See LSCOLORS below. In addition to the file types

mentioned in the -F option some extra attributes

(setuid bit set, etc.) are also displayed. The

colorization is dependent on a terminal type with the

proper termcap(5) capabilities. The default “cons25”

console has the proper capabilities, but to display

the colors in an xterm(1), for example, the TERM

variable must be set to “xterm-color”. Other

terminal types may require similar adjustments.

Colorization is silently disabled if the output is

not directed to a terminal unless the CLICOLOR\_FORCE

variable is defined or --color is set to “always”.

CLICOLOR\_FORCE Color sequences are normally disabled if the output

is not directed to a terminal. This can be

overridden by setting this variable. The TERM

variable still needs to reference a color capable

terminal however otherwise it is not possible to

determine which color sequences to use.

COLORTERM See description for CLICOLOR above.

COLUMNS If this variable contains a string representing a

decimal integer, it is used as the column position

width for displaying multiple-text-column output.

The ls utility calculates how many pathname text

columns to display based on the width provided. (See

-C and -x.)

LANG The locale to use when determining the order of day

and month in the long -l format output. See

environ(7) for more information.

LSCOLORS The value of this variable describes what color to

use for which attribute when colors are enabled with

CLICOLOR or COLORTERM. This string is a

concatenation of pairs of the format fb, where f is

the foreground color and b is the background color.

The color designators are as follows:

a black

b red

c green

d brown

e blue

f magenta

g cyan

h light grey

A bold black, usually shows up as dark grey

B bold red

C bold green

D bold brown, usually shows up as yellow

E bold blue

F bold magenta

G bold cyan

H bold light grey; looks like bright white

x default foreground or background

Note that the above are standard ANSI colors. The

actual display may differ depending on the color

capabilities of the terminal in use.

The order of the attributes are as follows:

1. directory

2. symbolic link

3. socket

4. pipe

5. executable

6. block special

7. character special

8. executable with setuid bit set

9. executable with setgid bit set

10. directory writable to others, with sticky

bit

11. directory writable to others, without

sticky bit

The default is "exfxcxdxbxegedabagacad", i.e., blue

foreground and default background for regular

directories, black foreground and red background for

setuid executables, etc.

LS\_COLWIDTHS If this variable is set, it is considered to be a

colon-delimited list of minimum column widths.

Unreasonable and insufficient widths are ignored

(thus zero signifies a dynamically sized column).

Not all columns have changeable widths. The fields

are, in order: inode, block count, number of links,

user name, group name, flags, file size, file name.

LS\_SAMESORT If this variable is set, the -t option sorts the

names of files with the same modification timestamp

in the same sense as the time sort. See the

description of the -t option for more details.

TERM The CLICOLOR and COLORTERM functionality depends on a

terminal type with color capabilities.

TZ The timezone to use when displaying dates. See

environ(7) for more information.

EXIT STATUS

The ls utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

List the contents of the current working directory in long format:

$ ls -l

In addition to listing the contents of the current working directory in

long format, show inode numbers, file flags (see chflags(1)), and suffix

each filename with a symbol representing its file type:

$ ls -lioF

List the files in /var/log, sorting the output such that the mostly

recently modified entries are printed first:

$ ls -lt /var/log

COMPATIBILITY

The group field is now automatically included in the long listing for

files in order to be compatible with the IEEE Std 1003.2 (“POSIX.2”)

specification.

SEE ALSO

chflags(1), chmod(1), getfacl(1), sort(1), xterm(1), localeconv(3),

strftime(3), strmode(3), termcap(5), maclabel(7), sticky(7), symlink(7),

getfmac(8)

STANDARDS

With the exception of options -g, -n and -o, the ls utility conforms to

IEEE Std 1003.1-2001 (“POSIX.1”) and IEEE Std 1003.1-2008 (“POSIX.1”).

The options -B, -D, -G, -I, -T, -U, -W, -Z, -b, -h, -w, -y and -, are

non-standard extensions.

The ACL support is compatible with IEEE Std 1003.2c (“POSIX.2c”) Draft 17

(withdrawn).

HISTORY

An ls command appeared in Version 1 AT&T UNIX.

BUGS

To maintain backward compatibility, the relationships between the many

options are quite complex.

The exception mentioned in the -s option description might be a feature

that was based on the fact that single-column output usually goes to

something other than a terminal. It is debatable whether this is a

design bug.

IEEE Std 1003.2 (“POSIX.2”) mandates opposite sort orders for files with

the same timestamp when sorting with the -t option.

NAME

pwd – return working directory name

SYNOPSIS

pwd [-L | -P]

DESCRIPTION

The pwd utility writes the absolute pathname of the current working

directory to the standard output.

Some shells may provide a builtin pwd command which is similar or

identical to this utility. Consult the builtin(1) manual page.

The options are as follows:

-L Display the logical current working directory.

-P Display the physical current working directory (all symbolic

links resolved).

If no options are specified, the -P option is assumed.

ENVIRONMENT

Environment variables used by pwd:

PWD Logical current working directory.

EXIT STATUS

The pwd utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Show current working directory with symbolic links resolved:

$ /bin/pwd

/usr/home/fernape

Show the logical current directory. Then use file(1) to inspect the

/home directory:

$ /bin/pwd -L

/home/fernape

$ file /home

/home: symbolic link to usr/home

SEE ALSO

builtin(1), cd(1), csh(1), realpath(1), sh(1), getcwd(3)

STANDARDS

The pwd utility conforms to IEEE Std 1003.1-2001 (“POSIX.1”).

HISTORY

The pwd command appeared in Version 5 AT&T UNIX.

BUGS

In csh(1) the command dirs is always faster because it is built into that

shell. However, it 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.

The -L option does not work unless the PWD environment variable is

exported by the shell.

NAME

builtin, !, %, ., :, @, [, {, }, alias, alloc, bg, bind, bindkey, break,

breaksw, builtins, case, cd, chdir, command, complete, continue, default,

dirs, do, done, echo, echotc, elif, else, end, endif, endsw, esac, eval,

exec, exit, export, false, fc, fg, filetest, fi, for, foreach, getopts,

glob, goto, hash, hashstat, history, hup, if, jobid, jobs, kill, limit,

local, log, login, logout, ls-F, nice, nohup, notify, onintr, popd,

printenv, printf, pushd, pwd, read, readonly, rehash, repeat, return,

sched, set, setenv, settc, setty, setvar, shift, source, stop, suspend,

switch, telltc, test, then, time, times, trap, true, type, ulimit, umask,

unalias, uncomplete, unhash, unlimit, unset, unsetenv, until, wait,

where, which, while – shell built-in commands

SYNOPSIS

See the built-in command description in the appropriate shell manual

page.

DESCRIPTION

Shell builtin commands are commands that can be executed within the

running shell's process. Note that, in the case of csh(1) builtin

commands, the command is executed in a subshell if it occurs as any

component of a pipeline except the last.

If a command specified to the shell contains a slash ‘/’, the shell will

not execute a builtin command, even if the last component of the

specified command matches the name of a builtin command. Thus, while

specifying “echo” causes a builtin command to be executed under shells

that support the echo builtin command, specifying “/bin/echo” or “./echo”

does not.

While some builtin commands may exist in more than one shell, their

operation may be different under each shell which supports them. Below

is a table which lists shell builtin commands, the standard shells that

support them and whether they exist as standalone utilities.

Only builtin commands for the csh(1) and sh(1) shells are listed here.

Consult a shell's manual page for details on the operation its builtin

commands. Beware that the sh(1) manual page, at least, calls some of

these commands “built-in commands” and some of them “reserved words”.

Users of other shells may need to consult an info(1) page or other

sources of documentation.

Commands marked “No\*\*” under External do exist externally, but are

implemented as scripts using a builtin command of the same name.

Command External csh(1) sh(1)

! No No Yes

% No Yes No

. No No Yes

: No Yes Yes

@ No Yes No

[ Yes No Yes

{ No No Yes

} No No Yes

alias No\*\* Yes Yes

alloc No Yes No

bg No\*\* Yes Yes

bind No No Yes

bindkey No Yes No

break No Yes Yes

breaksw No Yes No

builtin No No Yes

builtins No Yes No

case No Yes Yes

cd No\*\* Yes Yes

chdir No Yes Yes

command No\*\* No Yes

complete No Yes No

continue No Yes Yes

default No Yes No

dirs No Yes No

do No No Yes

done No No Yes

echo Yes Yes Yes

echotc No Yes No

elif No No Yes

else No Yes Yes

end No Yes No

endif No Yes No

endsw No Yes No

esac No No Yes

eval No Yes Yes

exec No Yes Yes

exit No Yes Yes

export No No Yes

false Yes No Yes

fc No\*\* No Yes

fg No\*\* Yes Yes

filetest No Yes No

fi No No Yes

for No No Yes

foreach No Yes No

getopts No\*\* No Yes

glob No Yes No

goto No Yes No

hash No\*\* No Yes

hashstat No Yes No

history No Yes No

hup No Yes No

if No Yes Yes

jobid No No Yes

jobs No\*\* Yes Yes

kill Yes Yes Yes

limit No Yes No

local No No Yes

log No Yes No

login Yes Yes No

logout No Yes No

ls-F No Yes No

nice Yes Yes No

nohup Yes Yes No

notify No Yes No

onintr No Yes No

popd No Yes No

printenv Yes Yes No

printf Yes No Yes

pushd No Yes No

pwd Yes No Yes

read No\*\* No Yes

readonly No No Yes

rehash No Yes No

repeat No Yes No

return No No Yes

sched No Yes No

set No Yes Yes

setenv No Yes No

settc No Yes No

setty No Yes No

setvar No No Yes

shift No Yes Yes

source No Yes No

stop No Yes No

suspend No Yes No

switch No Yes No

telltc No Yes No

test Yes No Yes

then No No Yes

time Yes Yes No

times No No Yes

trap No No Yes

true Yes No Yes

type No\*\* No Yes

ulimit No\*\* No Yes

umask No\*\* Yes Yes

unalias No\*\* Yes Yes

uncomplete No Yes No

unhash No Yes No

unlimit No Yes No

unset No Yes Yes

unsetenv No Yes No

until No No Yes

wait No\*\* Yes Yes

where No Yes No

which Yes Yes No

while No Yes Yes

SEE ALSO

csh(1), echo(1), false(1), info(1), kill(1), login(1), nice(1), nohup(1),

printenv(1), printf(1), pwd(1), sh(1), test(1), time(1), true(1),

which(1)

NAME

less - opposite of more

SYNOPSIS

less -?

less --help

less -V

less --version

less [-[+]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]

[-b space] [-h lines] [-j line] [-k keyfile]

[-{oO} logfile] [-p pattern] [-P prompt] [-t tag]

[-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]

[-# shift] [+[+]cmd] [--] [filename]...

(See the OPTIONS section for alternate option syntax with long option

names.)

DESCRIPTION

Less is a program similar to more(1), but which allows backward

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 a caret.)

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.

COMMANDS

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 or H Help: display a summary of these commands. If you forget all

the other commands, remember this one.

SPACE or ^V or f or ^F

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.

z Like SPACE, but if N is specified, it becomes the new window

size.

ESC-SPACE

Like SPACE, but scrolls a full screenful, even if it reaches

end-of-file in the process.

ENTER or 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.

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.

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.

w Like ESC-v, but if N is specified, it becomes the new window

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.

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.

J Like j, but continues to scroll beyond the end of the file.

K or Y Like k, but continues to scroll beyond the beginning of the

file.

ESC-) or RIGHTARROW

Scroll horizontally right N characters, default half the screen

width (see the -# option). If a number N is specified, it

becomes the default for future RIGHTARROW and LEFTARROW

commands. While the text is scrolled, it acts as though the -S

option (chop lines) were in effect.

ESC-( or LEFTARROW

Scroll horizontally left N characters, default half the screen

width (see the -# option). If a number N is specified, it

becomes the default for future RIGHTARROW and LEFTARROW

commands.

ESC-} or ^RIGHTARROW

Scroll horizontally right to show the end of the longest

displayed line.

ESC-{ or ^LEFTARROW

Scroll horizontally left back to the first column.

r or ^R or ^L

Repaint the screen.

R Repaint the screen, discarding any buffered input. That is,

reload the current file. Useful if the file is changing while

it is being viewed.

F Scroll forward, and keep trying to read when the end of file is

reached. Normally this command would be used when already at

the end of the file. It is a way to monitor the tail of a file

which is growing while it is being viewed. (The behavior is

similar to the "tail -f" command.) To stop waiting for more

data, enter the interrupt character (usually ^C). On some

systems you can also use ^X.

ESC-F Like F, but as soon as a line is found which matches the last

search pattern, the terminal bell is rung and forward scrolling

stops.

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.)

ESC-G Same as G, except if no number N is specified and the input is

standard input, goes to the last line which is currently

buffered.

p or % Go to a position N percent into the file. N should be between 0

and 100, and may contain a decimal point.

P Go to the line containing byte offset N in the file.

{ If a left curly bracket appears in the top line displayed on the

screen, the { command will go to the matching right curly

bracket. The matching right curly bracket is positioned on the

bottom line of the screen. If there is more than one left curly

bracket on the top line, a number N may be used to specify the

N-th bracket on the line.

} If a right curly bracket appears in the bottom line displayed on

the screen, the } command will go to the matching left curly

bracket. The matching left curly bracket is positioned on the

top line of the screen. If there is more than one right curly

bracket on the top line, a number N may be used to specify the

N-th bracket on the line.

( Like {, but applies to parentheses rather than curly brackets.

) Like }, but applies to parentheses rather than curly brackets.

[ Like {, but applies to square brackets rather than curly

brackets.

] Like }, but applies to square brackets rather than curly

brackets.

ESC-^F Followed by two characters, acts like {, but uses the two

characters as open and close brackets, respectively. For

example, "ESC ^F < >" could be used to go forward to the > which

matches the < in the top displayed line.

ESC-^B Followed by two characters, acts like }, but uses the two

characters as open and close brackets, respectively. For

example, "ESC ^B < >" could be used to go backward to the <

which matches the > in the bottom displayed line.

m Followed by any lowercase or uppercase letter, marks the first

displayed line with that letter. If the status column is

enabled via the -J option, the status column shows the marked

line.

M Acts like m, except the last displayed line is marked rather

than the first displayed line.

' (Single quote.) Followed by any lowercase or uppercase 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. Followed by a ^ or $, jumps to the beginning or end

of the file respectively. Marks are preserved when a new file

is examined, so the ' command can be used to switch between

input files.

^X^X Same as single quote.

ESC-m Followed by any lowercase or uppercase letter, clears the mark

identified by that letter.

/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 the regular expression library supplied by your

system. The search starts at the first line displayed (but see

the -a and -j options, which change this).

Certain characters are special if entered at the beginning of

the pattern; they modify the type of search rather than become

part of the pattern:

^N or !

Search for lines which do NOT match the pattern.

^E or \*

Search multiple files. That is, if the search reaches

the END of the current file without finding a match, the

search continues in the next file in the command line

list.

^F or @

Begin the search at the first line of the FIRST file in

the command line list, regardless of what is currently

displayed on the screen or the settings of the -a or -j

options.

^K Highlight any text which matches the pattern on the

current screen, but don't move to the first match (KEEP

current position).

^R Don't interpret regular expression metacharacters; that

is, do a simple textual comparison.

^W WRAP around the current file. That is, if the search

reaches the end of the current file without finding a

match, the search continues from the first line of the

current file up to the line where it started.

?pattern

Search backward in the file for the N-th line containing the

pattern. The search starts at the last line displayed (but see

the -a and -j options, which change this).

Certain characters are special as in the / command:

^N or !

Search for lines which do NOT match the pattern.

^E or \*

Search multiple files. That is, if the search reaches

the beginning of the current file without finding a

match, the search continues in the previous file in the

command line list.

^F or @

Begin the search at the last line of the last file in the

command line list, regardless of what is currently

displayed on the screen or the settings of the -a or -j

options.

^K As in forward searches.

^R As in forward searches.

^W WRAP around the current file. That is, if the search

reaches the beginning of the current file without finding

a match, the search continues from the last line of the

current file up to the line where it started.

ESC-/pattern

Same as "/\*".

ESC-?pattern

Same as "?\*".

n Repeat previous search, for N-th line containing the last

pattern. If the previous search was modified by ^N, the search

is made for the N-th line NOT containing the pattern. If the

previous search was modified by ^E, the search continues in the

next (or previous) file if not satisfied in the current file.

If the previous search was modified by ^R, the search is done

without using regular expressions. There is no effect if the

previous search was modified by ^F or ^K.

N Repeat previous search, but in the reverse direction.

ESC-n Repeat previous search, but crossing file boundaries. The

effect is as if the previous search were modified by \*.

ESC-N Repeat previous search, but in the reverse direction and

crossing file boundaries.

ESC-u Undo search highlighting. Turn off highlighting of strings

matching the current search pattern. If highlighting is already

off because of a previous ESC-u command, turn highlighting back

on. Any search command will also turn highlighting back on.

(Highlighting can also be disabled by toggling the -G option; in

that case search commands do not turn highlighting back on.)

ESC-U Like ESC-u but also clears the saved search pattern. If the

status column is enabled via the -J option, this clears all

search matches marked in the status column.

&pattern

Display only lines which match the pattern; lines which do not

match the pattern are not displayed. If pattern is empty (if

you type & immediately followed by ENTER), any filtering is

turned off, and all lines are displayed. While filtering is in

effect, an ampersand is displayed at the beginning of the

prompt, as a reminder that some lines in the file may be hidden.

Multiple & commands may be entered, in which case only lines

which match all of the patterns will be displayed.

Certain characters are special as in the / command:

^N or !

Display only lines which do NOT match the pattern.

^R Don't interpret regular expression metacharacters; that

is, do a simple textual comparison.

: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. A percent sign (%) in the

filename is replaced by the name of the current file. A pound

sign (#) is replaced by the name of the previously examined

file. However, two consecutive percent signs are simply

replaced with a single percent sign. This allows you to enter a

filename that contains a percent sign in the name. Similarly,

two consecutive pound signs are replaced with a single pound

sign. The filename is inserted into the command line list of

files so that it can be seen by subsequent :n and :p commands.

If the filename consists of several files, they are all inserted

into the list of files and the first one is examined. If the

filename contains one or more spaces, the entire filename should

be enclosed in double quotes (also see the -" option).

^X^V or E

Same as :e. Warning: some systems use ^V as a special

literalization character. On such systems, you may not be able

to use ^V.

:n Examine the next file (from the list of files given in the

command line). If a number N is specified, the N-th next file

is examined.

:p Examine the previous file in the command line list. If a number

N is specified, the N-th previous file is examined.

:x Examine the first file in the command line list. If a number N

is specified, the N-th file in the list is examined.

:d Remove the current file from the list of files.

t Go to the next tag, if there were more than one matches for the

current tag. See the -t option for more details about tags.

T Go to the previous tag, if there were more than one matches for

the current tag.

= or ^G or :f

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, the number of lines in the file and the percent of the

file above the last displayed line.

- Followed by one of the command line option letters (see OPTIONS

below), this will change the setting of that option and print a

message describing the new setting. If a ^P (CONTROL-P) is

entered immediately after the dash, the setting of the option is

changed but no message is printed. 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. If

no new value is entered, a message describing the current

setting is printed and nothing is changed.

-- Like the - command, but takes a long option name (see OPTIONS

below) rather than a single option letter. You must press ENTER

or RETURN after typing the option name. A ^P immediately after

the second dash suppresses printing of a message describing the

new setting, as in the - command.

-+ Followed by one of the command line option letters this will

reset the option to its default setting and print a message

describing the new setting. (The "-+X" command does the same

thing as "-+X" on the command line.) This does not work for

string-valued options.

--+ Like the -+ command, but takes a long option name rather than a

single option letter.

-! Followed by one of the command line option letters, this will

reset the option to the "opposite" of its default setting and

print a message describing the new setting. This does not work

for numeric or string-valued options.

--! Like the -! command, but takes a long option name rather than a

single option letter.

\_ (Underscore.) Followed by one of the command line option

letters, this will print a message describing the current

setting of that option. The setting of the option is not

changed.

\_\_ (Double underscore.) Like the \_ (underscore) command, but takes

a long option name rather than a single option letter. You must

press ENTER or RETURN after typing the option name.

+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 :q or :Q or ZZ

Exits less.

The following four 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 VISUAL if defined,

or EDITOR if VISUAL is not defined, or defaults to "vi" if

neither VISUAL nor EDITOR is defined. See also the discussion

of LESSEDIT under the section on PROMPTS below.

! 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.

A pound sign (#) is replaced by the name of the previously

examined file. "!!" repeats the last shell command. "!" with

no shell command simply invokes a shell. On Unix systems, the

shell is taken from the environment variable SHELL, or defaults

to "sh". On MS-DOS and OS/2 systems, the shell is the normal

command processor.

| <m> shell-command

<m> represents any mark letter. Pipes a section of the input

file to the given shell command. The section of the file to be

piped is between the position marked by the letter and the

current screen. The entire current screen is included,

regardless of whether the marked position is before or after the

current screen. <m> may also be ^ or $ to indicate beginning or

end of file respectively. If <m> is . or newline, the current

screen is piped.

s filename

Save the input to a file. This only works if the input is a

pipe, not an ordinary file.

OPTIONS

Command line options are described below. Most options may be changed

while less is running, via the "-" command.

Most options may be given in one of two forms: either a dash followed

by a single letter, or two dashes followed by a long option name. A

long option name may be abbreviated as long as the abbreviation is

unambiguous. For example, --quit-at-eof may be abbreviated --quit, but

not --qui, since both --quit-at-eof and --quiet begin with --qui. Some

long option names are in uppercase, such as --QUIT-AT-EOF, as distinct

from --quit-at-eof. Such option names need only have their first

letter capitalized; the remainder of the name may be in either case.

For example, --Quit-at-eof is equivalent to --QUIT-AT-EOF.

Options are also taken from the environment variable "LESS". For

example, to avoid typing "less -options ..." each time less is invoked,

you might tell csh:

setenv LESS "-options"

or if you use sh:

LESS="-options"; export LESS

On MS-DOS, you don't need the quotes, but you should replace any

percent signs in the options string by double percent signs.

The environment variable is parsed before the command line, so command

line options override the LESS environment variable. If an option

appears in the LESS variable, it can be reset to its default value on

the command line by beginning the command line option with "-+".

Some options like -k or -D require a string to follow the option

letter. The string for that option is considered to end when a dollar

sign ($) is found. For example, you can set two -D options on MS-DOS

like this:

LESS="Dn9.1$Ds4.1"

If the --use-backslash option appears earlier in the options, then a

dollar sign or backslash may be included literally in an option string

by preceding it with a backslash. If the --use-backslash option is not

in effect, then backslashes are not treated specially, and there is no

way to include a dollar sign in the option string.

-? or --help

This option displays a summary of the commands accepted by less

(the same as the h command). (Depending on how your shell

interprets the question mark, it may be necessary to quote the

question mark, thus: "-\?".)

-a or --search-skip-screen

By default, forward searches start at the top of the displayed

screen and backwards searches start at the bottom of the

displayed screen (except for repeated searches invoked by the n

or N commands, which start after or before the "target" line

respectively; see the -j option for more about the target line).

The -a option causes forward searches to instead start at the

bottom of the screen and backward searches to start at the top

of the screen, thus skipping all lines displayed on the screen.

-A or --SEARCH-SKIP-SCREEN

Causes all forward searches (not just non-repeated searches) to

start just after the target line, and all backward searches to

start just before the target line. Thus, forward searches will

skip part of the displayed screen (from the first line up to and

including the target line). Similarly backwards searches will

skip the displayed screen from the last line up to and including

the target line. This was the default behavior in less versions

prior to 441.

-bn or --buffers=n

Specifies the amount of buffer space less will use for each

file, in units of kilobytes (1024 bytes). By default 64 KB of

buffer space is used for each file (unless the file is a pipe;

see the -B option). The -b option specifies instead that n

kilobytes of buffer space should be used for each file. If n is

-1, buffer space is unlimited; that is, the entire file can be

read into memory.

-B or --auto-buffers

By default, when data is read from a pipe, buffers are allocated

automatically as needed. If a large amount of data is read from

the pipe, this can cause a large amount of memory to be

allocated. The -B option disables this automatic allocation of

buffers for pipes, so that only 64 KB (or the amount of space

specified by the -b option) is used for the pipe. Warning: use

of -B can result in erroneous display, since only the most

recently viewed part of the piped data is kept in memory; any

earlier data is lost.

-c or --clear-screen

Causes full screen repaints to be painted from the top line

down. By default, full screen repaints are done by scrolling

from the bottom of the screen.

-C or --CLEAR-SCREEN

Same as -c, for compatibility with older versions of less.

-d or --dumb

The -d option suppresses the error message normally displayed if

the terminal is dumb; that is, lacks some important capability,

such as the ability to clear the screen or scroll backward. The

-d option does not otherwise change the behavior of less on a

dumb terminal.

-Dxcolor or --color=xcolor

Changes the color of different parts of the displayed text. x

is a single character which selects the type of text whose color

is being set:

B Binary characters.

C Control characters.

E Errors and informational messages.

M Mark letters in the status column.

N Line numbers enabled via the -N option.

P Prompts.

R The rscroll character.

S Search results.

W The highlight enabled via the -w option.

d Bold text.

k Blinking text.

s Standout text.

u Underlined text.

The uppercase letters can be used only when the --use-color

option is enabled. When text color is specified by both an

uppercase letter and a lowercase letter, the uppercase letter

takes precedence. For example, error messages are normally

displayed as standout text. So if both "s" and "E" are given a

color, the "E" color applies to error messages, and the "s"

color applies to other standout text. The "d" and "u" letters

refer to bold and underline text formed by overstriking with

backspaces (see the -u option), not to text using ANSI escape

sequences with the -R option.

A lowercase letter may be followed by a + to indicate that both

the normal format change and the specified color should both be

used. For example, -Dug displays underlined text as green

without underlining; the green color has replaced the usual

underline formatting. But -Du+g displays underlined text as

both green and in underlined format.

color is either a 4-bit color string or an 8-bit color string:

A 4-bit color string is zero, one or two characters, where the

first character specifies the foreground color and the second

specifies the background color as follows:

b Blue

c Cyan

g Green

k Black

m Magenta

r Red

w White

y Yellow

The corresponding upper-case letter denotes a brighter shade of

the color. For example, -DNGk displays line numbers as bright

green text on a black background, and -DEbR displays error

messages as blue text on a bright red background. If either

character is a "-" or is omitted, the corresponding color is set

to that of normal text.

An 8-bit color string is one or two decimal integers separated

by a dot, where the first integer specifies the foreground color

and the second specifies the background color. Each integer is

a value between 0 and 255 inclusive which selects a "CSI 38;5"

color value (see

https://en.wikipedia.org/wiki/ANSI\_escape\_code#SGR\_parameters)

If either integer is a "-" or is omitted, the corresponding

color is set to that of normal text. On MS-DOS versions of

less, 8-bit color is not supported; instead, decimal values are

interpreted as 4-bit CHAR\_INFO.Attributes values (see

https://docs.microsoft.com/en-us/windows/console/char-info-str).

-e or --quit-at-eof

Causes less to automatically exit the second time it reaches

end-of-file. By default, the only way to exit less is via the

"q" command.

-E or --QUIT-AT-EOF

Causes less to automatically exit the first time it reaches end-

of-file.

-f or --force

Forces non-regular files to be opened. (A non-regular file is a

directory or a device special file.) Also suppresses the

warning message when a binary file is opened. By default, less

will refuse to open non-regular files. Note that some operating

systems will not allow directories to be read, even if -f is

set.

-F or --quit-if-one-screen

Causes less to automatically exit if the entire file can be

displayed on the first screen.

-g or --hilite-search

Normally, less will highlight ALL strings which match the last

search command. The -g option changes this behavior to

highlight only the particular string which was found by the last

search command. This can cause less to run somewhat faster than

the default.

-G or --HILITE-SEARCH

The -G option suppresses all highlighting of strings found by

search commands.

-hn or --max-back-scroll=n

Specifies a maximum number of lines to scroll backward. If it

is necessary to scroll backward more than n lines, the screen is

repainted in a forward direction instead. (If the terminal does

not have the ability to scroll backward, -h0 is implied.)

-i or --ignore-case

Causes searches to ignore case; that is, uppercase and lowercase

are considered identical. This option is ignored if any

uppercase letters appear in the search pattern; in other words,

if a pattern contains uppercase letters, then that search does

not ignore case.

-I or --IGNORE-CASE

Like -i, but searches ignore case even if the pattern contains

uppercase letters.

-jn or --jump-target=n

Specifies a line on the screen where the "target" line is to be

positioned. The target line is the line specified by any

command to search for a pattern, jump to a line number, jump to

a file percentage or jump to a tag. The screen line may be

specified by a number: the top line on the screen is 1, the next

is 2, and so on. The number may be negative to specify a line

relative to the bottom of the screen: the bottom line on the

screen is -1, the second to the bottom is -2, and so on.

Alternately, the screen line may be specified as a fraction of

the height of the screen, starting with a decimal point: .5 is

in the middle of the screen, .3 is three tenths down from the

first line, and so on. If the line is specified as a fraction,

the actual line number is recalculated if the terminal window is

resized, so that the target line remains at the specified

fraction of the screen height. If any form of the -j option is

used, repeated forward searches (invoked with "n" or "N") begin

at the line immediately after the target line, and repeated

backward searches begin at the target line, unless changed by -a

or -A. For example, if "-j4" is used, the target line is the

fourth line on the screen, so forward searches begin at the

fifth line on the screen. However nonrepeated searches (invoked

with "/" or "?") always begin at the start or end of the current

screen respectively.

-J or --status-column

Displays a status column at the left edge of the screen. The

status column shows the lines that matched the current search,

and any lines that are marked (via the m or M command).

-kfilename or --lesskey-file=filename

Causes less to open and interpret the named file as a lesskey(1)

file. Multiple -k options may be specified. If the LESSKEY or

LESSKEY\_SYSTEM environment variable is set, or if a lesskey file

is found in a standard place (see KEY BINDINGS), it is also used

as a lesskey file.

-K or --quit-on-intr

Causes less to exit immediately (with status 2) when an

interrupt character (usually ^C) is typed. Normally, an

interrupt character causes less to stop whatever it is doing and

return to its command prompt. Note that use of this option

makes it impossible to return to the command prompt from the "F"

command.

-L or --no-lessopen

Ignore the LESSOPEN environment variable (see the INPUT

PREPROCESSOR section below). This option can be set from within

less, but it will apply only to files opened subsequently, not

to the file which is currently open.

-m or --long-prompt

Causes less to prompt verbosely (like more), with the percent

into the file. By default, less prompts with a colon.

-M or --LONG-PROMPT

Causes less to prompt even more verbosely than more.

-n or --line-numbers

Suppresses line numbers. The default (to use line numbers) may

cause less to run more slowly in some cases, especially with a

very large input file. Suppressing line numbers with the -n

option will avoid this problem. Using line numbers means: the

line number will be displayed in the verbose prompt and in the =

command, and the v command will pass the current line number to

the editor (see also the discussion of LESSEDIT in PROMPTS

below).

-N or --LINE-NUMBERS

Causes a line number to be displayed at the beginning of each

line in the display.

-ofilename or --log-file=filename

Causes less to copy its input to the named file as it is being

viewed. This applies only when the input file is a pipe, not an

ordinary file. If the file already exists, less will ask for

confirmation before overwriting it.

-Ofilename or --LOG-FILE=filename

The -O option is like -o, but it will overwrite an existing file

without asking for confirmation.

If no log file has been specified, the -o and -O options can be

used from within less to specify a log file. Without a file

name, they will simply report the name of the log file. The "s"

command is equivalent to specifying -o from within less.

-ppattern or --pattern=pattern

The -p option on the command line is equivalent to specifying

+/pattern; that is, it tells less to start at the first

occurrence of pattern in the file.

-Pprompt or --prompt=prompt

Provides a way to tailor the three prompt styles to your own

preference. This option would normally be put in the LESS

environment variable, rather than being typed in with each less

command. Such an option must either be the last option in the

LESS variable, or be terminated by a dollar sign.

-Ps followed by a string changes the default (short) prompt to

that string.

-Pm changes the medium (-m) prompt.

-PM changes the long (-M) prompt.

-Ph changes the prompt for the help screen.

-P= changes the message printed by the = command.

-Pw changes the message printed while waiting for data (in the

F command).

All prompt strings consist of a sequence of letters and special

escape sequences. See the section on PROMPTS for more details.

-q or --quiet or --silent

Causes moderately "quiet" operation: the terminal bell is not

rung if an attempt is made to scroll past the end of the file or

before the beginning of the file. If the terminal has a "visual

bell", it is used instead. The bell will be rung on certain

other errors, such as typing an invalid character. The default

is to ring the terminal bell in all such cases.

-Q or --QUIET or --SILENT

Causes totally "quiet" operation: the terminal bell is never

rung. If the terminal has a "visual bell", it is used in all

cases where the terminal bell would have been rung.

-r or --raw-control-chars

Causes "raw" control characters to be displayed. The default is

to display control characters using the caret notation; for

example, a control-A (octal 001) is displayed as "^A". Warning:

when the -r option is used, less cannot keep track of the actual

appearance of the screen (since this depends on how the screen

responds to each type of control character). Thus, various

display problems may result, such as long lines being split in

the wrong place.

USE OF THE -r OPTION IS NOT RECOMMENDED.

-R or --RAW-CONTROL-CHARS

Like -r, but only ANSI "color" escape sequences and OSC 8

hyperlink sequences are output in "raw" form. Unlike -r, the

screen appearance is maintained correctly, provided that there

are no escape sequences in the file other than these types of

escape sequences. Color escape sequences are only supported

when the color is changed within one line, not across lines. In

other words, the beginning of each line is assumed to be normal

(non-colored), regardless of any escape sequences in previous

lines. For the purpose of keeping track of screen appearance,

these escape sequences are assumed to not move the cursor.

OSC 8 hyperlinks are sequences of the form:

ESC ] 8 ; ... \7

The terminating sequence may be either a BEL character (\7) or

the two-character sequence "ESC \".

ANSI color escape sequences are sequences of the form:

ESC [ ... m

where the "..." is zero or more color specification characters.

You can make less think that characters other than "m" can end

ANSI color escape sequences by setting the environment variable

LESSANSIENDCHARS to the list of characters which can end a color

escape sequence. And you can make less think that characters

other than the standard ones may appear between the ESC and the

m by setting the environment variable LESSANSIMIDCHARS to the

list of characters which can appear.

-s or --squeeze-blank-lines

Causes consecutive blank lines to be squeezed into a single

blank line. This is useful when viewing nroff output.

-S or --chop-long-lines

Causes lines longer than the screen width to be chopped

(truncated) rather than wrapped. That is, the portion of a long

line that does not fit in the screen width is not displayed

until you press RIGHT-ARROW. The default is to wrap long lines;

that is, display the remainder on the next line.

-ttag or --tag=tag

The -t option, followed immediately by a TAG, will edit the file

containing that tag. For this to work, tag information must be

available; for example, there may be a file in the current

directory called "tags", which was previously built by ctags(1)

or an equivalent command. If the environment variable

LESSGLOBALTAGS is set, it is taken to be the name of a command

compatible with global(1), and that command is executed to find

the tag. (See http://www.gnu.org/software/global/global.html).

The -t option may also be specified from within less (using the

- command) as a way of examining a new file. The command ":t"

is equivalent to specifying -t from within less.

-Ttagsfile or --tag-file=tagsfile

Specifies a tags file to be used instead of "tags".

-u or --underline-special

Causes backspaces and carriage returns to be treated as

printable characters; that is, they are sent to the terminal

when they appear in the input.

-U or --UNDERLINE-SPECIAL

Causes backspaces, tabs, carriage returns and "formatting

characters" (as defined by Unicode) to be treated as control

characters; that is, they are handled as specified by the -r

option.

By default, if neither -u nor -U is given, backspaces which

appear adjacent to an underscore character are treated

specially: the underlined text is displayed using the terminal's

hardware underlining capability. Also, backspaces which appear

between two identical characters are treated specially: the

overstruck text is printed using the terminal's hardware

boldface capability. Other backspaces are deleted, along with

the preceding character. Carriage returns immediately followed

by a newline are deleted. Other carriage returns are handled as

specified by the -r option. Unicode formatting characters, such

as the Byte Order Mark, are sent to the terminal. Text which is

overstruck or underlined can be searched for if neither -u nor

-U is in effect.

-V or --version

Displays the version number of less.

-w or --hilite-unread

Temporarily highlights the first "new" line after a forward

movement of a full page. The first "new" line is the line

immediately following the line previously at the bottom of the

screen. Also highlights the target line after a g or p command.

The highlight is removed at the next command which causes

movement. The entire line is highlighted, unless the -J option

is in effect, in which case only the status column is

highlighted.

-W or --HILITE-UNREAD

Like -w, but temporarily highlights the first new line after any

forward movement command larger than one line.

-xn,... or --tabs=n,...

Sets tab stops. If only one n is specified, tab stops are set

at multiples of n. If multiple values separated by commas are

specified, tab stops are set at those positions, and then

continue with the same spacing as the last two. For example,

-x9,17 will set tabs at positions 9, 17, 25, 33, etc. The

default for n is 8.

-X or --no-init

Disables sending the termcap initialization and deinitialization

strings to the terminal. This is sometimes desirable if the

deinitialization string does something unnecessary, like

clearing the screen.

-yn or --max-forw-scroll=n

Specifies a maximum number of lines to scroll forward. If it is

necessary to scroll forward more than n lines, the screen is

repainted instead. The -c or -C option may be used to repaint

from the top of the screen if desired. By default, any forward

movement causes scrolling.

-zn or --window=n or -n

Changes the default scrolling window size to n lines. The

default is one screenful. The z and w commands can also be used

to change the window size. The "z" may be omitted for

compatibility with some versions of more. If the number n is

negative, it indicates n lines less than the current screen

size. For example, if the screen is 24 lines, -z-4 sets the

scrolling window to 20 lines. If the screen is resized to 40

lines, the scrolling window automatically changes to 36 lines.

-"cc or --quotes=cc

Changes the filename quoting character. This may be necessary

if you are trying to name a file which contains both spaces and

quote characters. Followed by a single character, this changes

the quote character to that character. Filenames containing a

space should then be surrounded by that character rather than by

double quotes. Followed by two characters, changes the open

quote to the first character, and the close quote to the second

character. Filenames containing a space should then be preceded

by the open quote character and followed by the close quote

character. Note that even after the quote characters are

changed, this option remains -" (a dash followed by a double

quote).

-~ or --tilde

Normally lines after end of file are displayed as a single tilde

(~). This option causes lines after end of file to be displayed

as blank lines.

-# or --shift

Specifies the default number of positions to scroll horizontally

in the RIGHTARROW and LEFTARROW commands. If the number

specified is zero, it sets the default number of positions to

one half of the screen width. Alternately, the number may be

specified as a fraction of the width of the screen, starting

with a decimal point: .5 is half of the screen width, .3 is

three tenths of the screen width, and so on. If the number is

specified as a fraction, the actual number of scroll positions

is recalculated if the terminal window is resized, so that the

actual scroll remains at the specified fraction of the screen

width.

--follow-name

Normally, if the input file is renamed while an F command is

executing, less will continue to display the contents of the

original file despite its name change. If --follow-name is

specified, during an F command less will periodically attempt to

reopen the file by name. If the reopen succeeds and the file is

a different file from the original (which means that a new file

has been created with the same name as the original (now

renamed) file), less will display the contents of that new file.

--incsearch

Subsequent search commands will be "incremental"; that is, less

will advance to the next line containing the search pattern as

each character of the pattern is typed in.

--line-num-width

Sets the minimum width of the line number field when the -N

option is in effect. The default is 7 characters.

--mouse

Enables mouse input: scrolling the mouse wheel down moves

forward in the file, scrolling the mouse wheel up moves

backwards in the file, and clicking the mouse sets the "#" mark

to the line where the mouse is clicked. The number of lines to

scroll when the wheel is moved can be set by the --wheel-lines

option. Mouse input works only on terminals which support X11

mouse reporting, and on the Windows version of less.

--MOUSE

Like --mouse, except the direction scrolled on mouse wheel

movement is reversed.

--no-keypad

Disables sending the keypad initialization and deinitialization

strings to the terminal. This is sometimes useful if the keypad

strings make the numeric keypad behave in an undesirable manner.

--no-histdups

This option changes the behavior so that if a search string or

file name is typed in, and the same string is already in the

history list, the existing copy is removed from the history list

before the new one is added. Thus, a given string will appear

only once in the history list. Normally, a string may appear

multiple times.

--rscroll

This option changes the character used to mark truncated lines.

It may begin with a two-character attribute indicator like

LESSBINFMT does. If there is no attribute indicator, standout

is used. If set to "-", truncated lines are not marked.

--save-marks

Save marks in the history file, so marks are retained across

different invocations of less.

--status-col-width

Sets the width of the status column when the -J option is in

effect. The default is 2 characters.

--use-backslash

This option changes the interpretations of options which follow

this one. After the --use-backslash option, any backslash in an

option string is removed and the following character is taken

literally. This allows a dollar sign to be included in option

strings.

--use-color

Enables the colored text in various places. The -D option can

be used to change the colors. Colored text works only if the

terminal supports ANSI color escape sequences (as defined in

ECMA-48 SGR; see

https://www.ecma-international.org/publications-and-

standards/standards/ecma-48).

--wheel-lines=n

Set the number of lines to scroll when the mouse wheel is

scrolled and the --mouse or --MOUSE option is in effect. The

default is 1 line.

-- A command line argument of "--" marks the end of option

arguments. Any arguments following this are interpreted as

filenames. This can be useful when viewing a file whose name

begins with a "-" or "+".

+ If a command line option begins with +, the remainder of that

option is taken to be an initial command to less. For example,

+G tells less to start at the end of the file rather than the

beginning, and +/xyz tells it to start at the first occurrence

of "xyz" in the file. As a special case, +<number> acts like

+<number>g; that is, it starts the display at the specified line

number (however, see the caveat under the "g" command above).

If the option starts with ++, the initial command applies to

every file being viewed, not just the first one. The + command

described previously may also be used to set (or change) an

initial command for every file.

LINE EDITING

When entering a command line at the bottom of the screen (for example,

a filename for the :e command, or the pattern for a search command),

certain keys can be used to manipulate the command line. Most commands

have an alternate form in [ brackets ] which can be used if a key does

not exist on a particular keyboard. (Note that the forms beginning

with ESC do not work in some MS-DOS and Windows systems because ESC is

the line erase character.) Any of these special keys may be entered

literally by preceding it with the "literal" character, either ^V or

^A. A backslash itself may also be entered literally by entering two

backslashes.

LEFTARROW [ ESC-h ]

Move the cursor one space to the left.

RIGHTARROW [ ESC-l ]

Move the cursor one space to the right.

^LEFTARROW [ ESC-b or ESC-LEFTARROW ]

(That is, CONTROL and LEFTARROW simultaneously.) Move the

cursor one word to the left.

^RIGHTARROW [ ESC-w or ESC-RIGHTARROW ]

(That is, CONTROL and RIGHTARROW simultaneously.) Move the

cursor one word to the right.

HOME [ ESC-0 ]

Move the cursor to the beginning of the line.

END [ ESC-$ ]

Move the cursor to the end of the line.

BACKSPACE

Delete the character to the left of the cursor, or cancel the

command if the command line is empty.

DELETE or [ ESC-x ]

Delete the character under the cursor.

^BACKSPACE [ ESC-BACKSPACE ]

(That is, CONTROL and BACKSPACE simultaneously.) Delete the

word to the left of the cursor.

^DELETE [ ESC-X or ESC-DELETE ]

(That is, CONTROL and DELETE simultaneously.) Delete the word

under the cursor.

UPARROW [ ESC-k ]

Retrieve the previous command line. If you first enter some

text and then press UPARROW, it will retrieve the previous

command which begins with that text.

DOWNARROW [ ESC-j ]

Retrieve the next command line. If you first enter some text

and then press DOWNARROW, it will retrieve the next command

which begins with that text.

TAB Complete the partial filename to the left of the cursor. If it

matches more than one filename, the first match is entered into

the command line. Repeated TABs will cycle thru the other

matching filenames. If the completed filename is a directory, a

"/" is appended to the filename. (On MS-DOS systems, a "\" is

appended.) The environment variable LESSSEPARATOR can be used

to specify a different character to append to a directory name.

BACKTAB [ ESC-TAB ]

Like, TAB, but cycles in the reverse direction thru the matching

filenames.

^L Complete the partial filename to the left of the cursor. If it

matches more than one filename, all matches are entered into the

command line (if they fit).

^U (Unix and OS/2) or ESC (MS-DOS)

Delete the entire command line, or cancel the command if the

command line is empty. If you have changed your line-kill

character in Unix to something other than ^U, that character is

used instead of ^U.

^G Delete the entire command line and return to the main prompt.

KEY BINDINGS

You may define your own less commands by using the program lesskey(1)

to create a lesskey file. This file specifies a set of command keys

and an action associated with each key. You may also use lesskey to

change the line-editing keys (see LINE EDITING), and to set environment

variables. If the environment variable LESSKEY is set, less uses that

as the name of the lesskey file. Otherwise, less looks in a standard

place for the lesskey file: On Unix systems, less looks for a lesskey

file called "$HOME/.less". On MS-DOS and Windows systems, less looks

for a lesskey file called "$HOME/\_less", and if it is not found there,

then looks for a lesskey file called "\_less" in any directory specified

in the PATH environment variable. On OS/2 systems, less looks for a

lesskey file called "$HOME/less.ini", and if it is not found, then

looks for a lesskey file called "less.ini" in any directory specified

in the INIT environment variable, and if it not found there, then looks

for a lesskey file called "less.ini" in any directory specified in the

PATH environment variable. See the lesskey manual page for more

details.

A system-wide lesskey file may also be set up to provide key bindings.

If a key is defined in both a local lesskey file and in the system-wide

file, key bindings in the local file take precedence over those in the

system-wide file. If the environment variable LESSKEY\_SYSTEM is set,

less uses that as the name of the system-wide lesskey file. Otherwise,

less looks in a standard place for the system-wide lesskey file: On

Unix systems, the system-wide lesskey file is /usr/local/etc/sysless.

(However, if less was built with a different sysconf directory than

/usr/local/etc, that directory is where the sysless file is found.) On

MS-DOS and Windows systems, the system-wide lesskey file is

c:\\_sysless. On OS/2 systems, the system-wide lesskey file is

c:\sysless.ini.

INPUT PREPROCESSOR

You may define an "input preprocessor" for less. Before less opens a

file, it first gives your input preprocessor a chance to modify the way

the contents of the file are displayed. An input preprocessor is

simply an executable program (or shell script), which writes the

contents of the file to a different file, called the replacement file.

The contents of the replacement file are then displayed in place of the

contents of the original file. However, it will appear to the user as

if the original file is opened; that is, less will display the original

filename as the name of the current file.

An input preprocessor receives one command line argument, the original

filename, as entered by the user. It should create the replacement

file, and when finished, print the name of the replacement file to its

standard output. If the input preprocessor does not output a

replacement filename, less uses the original file, as normal. The

input preprocessor is not called when viewing standard input. To set

up an input preprocessor, set the LESSOPEN environment variable to a

command line which will invoke your input preprocessor. This command

line should include one occurrence of the string "%s", which will be

replaced by the filename when the input preprocessor command is

invoked.

When less closes a file opened in such a way, it will call another

program, called the input postprocessor, which may perform any desired

clean-up action (such as deleting the replacement file created by

LESSOPEN). This program receives two command line arguments, the

original filename as entered by the user, and the name of the

replacement file. To set up an input postprocessor, set the LESSCLOSE

environment variable to a command line which will invoke your input

postprocessor. It may include two occurrences of the string "%s"; the

first is replaced with the original name of the file and the second

with the name of the replacement file, which was output by LESSOPEN.

For example, on many Unix systems, these two scripts will allow you to

keep files in compressed format, but still let less view them directly:

lessopen.sh:

#! /bin/sh

case "$1" in

\*.Z) TEMPFILE=$(mktemp)

uncompress -c $1 >$TEMPFILE 2>/dev/null

if [ -s $TEMPFILE ]; then

echo $TEMPFILE

else

rm -f $TEMPFILE

fi

;;

esac

lessclose.sh:

#! /bin/sh

rm $2

To use these scripts, put them both where they can be executed and set

LESSOPEN="lessopen.sh %s", and LESSCLOSE="lessclose.sh %s %s". More

complex LESSOPEN and LESSCLOSE scripts may be written to accept other

types of compressed files, and so on.

It is also possible to set up an input preprocessor to pipe the file

data directly to less, rather than putting the data into a replacement

file. This avoids the need to decompress the entire file before

starting to view it. An input preprocessor that works this way is

called an input pipe. An input pipe, instead of writing the name of a

replacement file on its standard output, writes the entire contents of

the replacement file on its standard output. If the input pipe does

not write any characters on its standard output, then there is no

replacement file and less uses the original file, as normal. To use an

input pipe, make the first character in the LESSOPEN environment

variable a vertical bar (|) to signify that the input preprocessor is

an input pipe. As with non-pipe input preprocessors, the command

string must contain one occurrence of %s, which is replaced with the

filename of the input file.

For example, on many Unix systems, this script will work like the

previous example scripts:

lesspipe.sh:

#! /bin/sh

case "$1" in

\*.Z) uncompress -c $1 2>/dev/null

;;

\*) exit 1

;;

esac

exit $?

To use this script, put it where it can be executed and set

LESSOPEN="|lesspipe.sh %s".

Note that a preprocessor cannot output an empty file, since that is

interpreted as meaning there is no replacement, and the original file

is used. To avoid this, if LESSOPEN starts with two vertical bars, the

exit status of the script becomes meaningful. If the exit status is

zero, the output is considered to be replacement text, even if it is

empty. If the exit status is nonzero, any output is ignored and the

original file is used. For compatibility with previous versions of

less, if LESSOPEN starts with only one vertical bar, the exit status of

the preprocessor is ignored.

When an input pipe is used, a LESSCLOSE postprocessor can be used, but

it is usually not necessary since there is no replacement file to clean

up. In this case, the replacement file name passed to the LESSCLOSE

postprocessor is "-".

For compatibility with previous versions of less, the input

preprocessor or pipe is not used if less is viewing standard input.

However, if the first character of LESSOPEN is a dash (-), the input

preprocessor is used on standard input as well as other files. In this

case, the dash is not considered to be part of the preprocessor

command. If standard input is being viewed, the input preprocessor is

passed a file name consisting of a single dash. Similarly, if the

first two characters of LESSOPEN are vertical bar and dash (|-) or two

vertical bars and a dash (||-), the input pipe is used on standard

input as well as other files. Again, in this case the dash is not

considered to be part of the input pipe command.

NATIONAL CHARACTER SETS

There are three types of characters in the input file:

normal characters

can be displayed directly to the screen.

control characters

should not be displayed directly, but are expected to be found

in ordinary text files (such as backspace and tab).

binary characters

should not be displayed directly and are not expected to be

found in text files.

A "character set" is simply a description of which characters are to be

considered normal, control, and binary. The LESSCHARSET environment

variable may be used to select a character set. Possible values for

LESSCHARSET are:

ascii BS, TAB, NL, CR, and formfeed are control characters, all chars

with values between 32 and 126 are normal, and all others are

binary.

iso8859

Selects an ISO 8859 character set. This is the same as ASCII,

except characters between 160 and 255 are treated as normal

characters.

latin1 Same as iso8859.

latin9 Same as iso8859.

dos Selects a character set appropriate for MS-DOS.

ebcdic Selects an EBCDIC character set.

IBM-1047

Selects an EBCDIC character set used by OS/390 Unix Services.

This is the EBCDIC analogue of latin1. You get similar results

by setting either LESSCHARSET=IBM-1047 or LC\_CTYPE=en\_US in your

environment.

koi8-r Selects a Russian character set.

next Selects a character set appropriate for NeXT computers.

utf-8 Selects the UTF-8 encoding of the ISO 10646 character set.

UTF-8 is special in that it supports multi-byte characters in

the input file. It is the only character set that supports

multi-byte characters.

windows

Selects a character set appropriate for Microsoft Windows (cp

1251).

In rare cases, it may be desired to tailor less to use a character set

other than the ones definable by LESSCHARSET. In this case, the

environment variable LESSCHARDEF can be used to define a character set.

It should be set to a string where each character in the string

represents one character in the character set. The character "." is

used for a normal character, "c" for control, and "b" for binary. A

decimal number may be used for repetition. For example, "bccc4b."

would mean character 0 is binary, 1, 2 and 3 are control, 4, 5, 6 and 7

are binary, and 8 is normal. All characters after the last are taken

to be the same as the last, so characters 9 through 255 would be

normal. (This is an example, and does not necessarily represent any

real character set.)

This table shows the value of LESSCHARDEF which is equivalent to each

of the possible values for LESSCHARSET:

ascii 8bcccbcc18b95.b

dos 8bcccbcc12bc5b95.b.

ebcdic 5bc6bcc7bcc41b.9b7.9b5.b..8b6.10b6.b9.7b

9.8b8.17b3.3b9.7b9.8b8.6b10.b.b.b.

IBM-1047 4cbcbc3b9cbccbccbb4c6bcc5b3cbbc4bc4bccbc

191.b

iso8859 8bcccbcc18b95.33b.

koi8-r 8bcccbcc18b95.b128.

latin1 8bcccbcc18b95.33b.

next 8bcccbcc18b95.bb125.bb

If neither LESSCHARSET nor LESSCHARDEF is set, but any of the strings

"UTF-8", "UTF8", "utf-8" or "utf8" is found in the LC\_ALL, LC\_CTYPE or

LANG environment variables, then the default character set is utf-8.

If that string is not found, but your system supports the setlocale

interface, less will use setlocale to determine the character set.

setlocale is controlled by setting the LANG or LC\_CTYPE environment

variables.

Finally, if the setlocale interface is also not available, the default

character set is latin1.

Control and binary characters are displayed in standout (reverse

video). Each such character is displayed in caret notation if possible

(e.g. ^A for control-A). Caret notation is used only if inverting the

0100 bit results in a normal printable character. Otherwise, the

character is displayed as a hex number in angle brackets. This format

can be changed by setting the LESSBINFMT environment variable.

LESSBINFMT may begin with a "\*" and one character to select the display

attribute: "\*k" is blinking, "\*d" is bold, "\*u" is underlined, "\*s" is

standout, and "\*n" is normal. If LESSBINFMT does not begin with a "\*",

normal attribute is assumed. The remainder of LESSBINFMT is a string

which may include one printf-style escape sequence (a % followed by x,

X, o, d, etc.). For example, if LESSBINFMT is "\*u[%x]", binary

characters are displayed in underlined hexadecimal surrounded by

brackets. The default if no LESSBINFMT is specified is "\*s<%02X>".

Warning: the result of expanding the character via LESSBINFMT must be

less than 31 characters.

When the character set is utf-8, the LESSUTFBINFMT environment variable

acts similarly to LESSBINFMT but it applies to Unicode code points that

were successfully decoded but are unsuitable for display (e.g.,

unassigned code points). Its default value is "<U+%04lX>". Note that

LESSUTFBINFMT and LESSBINFMT share their display attribute setting

("\*x") so specifying one will affect both; LESSUTFBINFMT is read after

LESSBINFMT so its setting, if any, will have priority. Problematic

octets in a UTF-8 file (octets of a truncated sequence, octets of a

complete but non-shortest form sequence, invalid octets, and stray

trailing octets) are displayed individually using LESSBINFMT so as to

facilitate diagnostic of how the UTF-8 file is ill-formed.

PROMPTS

The -P option allows you to tailor the prompt to your preference. The

string given to the -P option replaces the specified prompt string.

Certain characters in the string are interpreted specially. The prompt

mechanism is rather complicated to provide flexibility, but the

ordinary user need not understand the details of constructing

personalized prompt strings.

A percent sign followed by a single character is expanded according to

what the following character is:

%bX Replaced by the byte offset into the current input file. The b

is followed by a single character (shown as X above) which

specifies the line whose byte offset is to be used. If the

character is a "t", the byte offset of the top line in the

display is used, an "m" means use the middle line, a "b" means

use the bottom line, a "B" means use the line just after the

bottom line, and a "j" means use the "target" line, as specified

by the -j option.

%B Replaced by the size of the current input file.

%c Replaced by the column number of the text appearing in the first

column of the screen.

%dX Replaced by the page number of a line in the input file. The

line to be used is determined by the X, as with the %b option.

%D Replaced by the number of pages in the input file, or

equivalently, the page number of the last line in the input

file.

%E Replaced by the name of the editor (from the VISUAL environment

variable, or the EDITOR environment variable if VISUAL is not

defined). See the discussion of the LESSEDIT feature below.

%f Replaced by the name of the current input file.

%F Replaced by the last component of the name of the current input

file.

%g Replaced by the shell-escaped name of the current input file.

This is useful when the expanded string will be used in a shell

command, such as in LESSEDIT.

%i Replaced by the index of the current file in the list of input

files.

%lX Replaced by the line number of a line in the input file. The

line to be used is determined by the X, as with the %b option.

%L Replaced by the line number of the last line in the input file.

%m Replaced by the total number of input files.

%pX Replaced by the percent into the current input file, based on

byte offsets. The line used is determined by the X as with the

%b option.

%PX Replaced by the percent into the current input file, based on

line numbers. The line used is determined by the X as with the

%b option.

%s Same as %B.

%t Causes any trailing spaces to be removed. Usually used at the

end of the string, but may appear anywhere.

%T Normally expands to the word "file". However if viewing files

via a tags list using the -t option, it expands to the word

"tag".

%x Replaced by the name of the next input file in the list.

If any item is unknown (for example, the file size if input is a pipe),

a question mark is printed instead.

The format of the prompt string can be changed depending on certain

conditions. A question mark followed by a single character acts like

an "IF": depending on the following character, a condition is

evaluated. If the condition is true, any characters following the

question mark and condition character, up to a period, are included in

the prompt. If the condition is false, such characters are not

included. A colon appearing between the question mark and the period

can be used to establish an "ELSE": any characters between the colon

and the period are included in the string if and only if the IF

condition is false. Condition characters (which follow a question

mark) may be:

?a True if any characters have been included in the prompt so far.

?bX True if the byte offset of the specified line is known.

?B True if the size of current input file is known.

?c True if the text is horizontally shifted (%c is not zero).

?dX True if the page number of the specified line is known.

?e True if at end-of-file.

?f True if there is an input filename (that is, if input is not a

pipe).

?lX True if the line number of the specified line is known.

?L True if the line number of the last line in the file is known.

?m True if there is more than one input file.

?n True if this is the first prompt in a new input file.

?pX True if the percent into the current input file, based on byte

offsets, of the specified line is known.

?PX True if the percent into the current input file, based on line

numbers, of the specified line is known.

?s Same as "?B".

?x True if there is a next input file (that is, if the current

input file is not the last one).

Any characters other than the special ones (question mark, colon,

period, percent, and backslash) become literally part of the prompt.

Any of the special characters may be included in the prompt literally

by preceding it with a backslash.

Some examples:

?f%f:Standard input.

This prompt prints the filename, if known; otherwise the string

"Standard input".

?f%f .?ltLine %lt:?pt%pt\%:?btByte %bt:-...

This prompt would print the filename, if known. The filename is

followed by the line number, if known, otherwise the percent if known,

otherwise the byte offset if known. Otherwise, a dash is printed.

Notice how each question mark has a matching period, and how the %

after the %pt is included literally by escaping it with a backslash.

?n?f%f .?m(%T %i of %m) ..?e(END) ?x- Next\: %x..%t";

This prints the filename if this is the first prompt in a file,

followed by the "file N of N" message if there is more than one input

file. Then, if we are at end-of-file, the string "(END)" is printed

followed by the name of the next file, if there is one. Finally, any

trailing spaces are truncated. This is the default prompt. For

reference, here are the defaults for the other two prompts (-m and -M

respectively). Each is broken into two lines here for readability

only.

?n?f%f .?m(%T %i of %m) ..?e(END) ?x- Next\: %x.:

?pB%pB\%:byte %bB?s/%s...%t

?f%f .?n?m(%T %i of %m) ..?ltlines %lt-%lb?L/%L. :

byte %bB?s/%s. .?e(END) ?x- Next\: %x.:?pB%pB\%..%t

And here is the default message produced by the = command:

?f%f .?m(%T %i of %m) .?ltlines %lt-%lb?L/%L. .

byte %bB?s/%s. ?e(END) :?pB%pB\%..%t

The prompt expansion features are also used for another purpose: if an

environment variable LESSEDIT is defined, it is used as the command to

be executed when the v command is invoked. The LESSEDIT string is

expanded in the same way as the prompt strings. The default value for

LESSEDIT is:

%E ?lm+%lm. %g

Note that this expands to the editor name, followed by a + and the line

number, followed by the shell-escaped file name. If your editor does

not accept the "+linenumber" syntax, or has other differences in

invocation syntax, the LESSEDIT variable can be changed to modify this

default.

SECURITY

When the environment variable LESSSECURE is set to 1, less runs in a

"secure" mode. This means these features are disabled:

! the shell command

| the pipe command

:e the examine command.

v the editing command

s -o log files

-k use of lesskey files

-t use of tags files

metacharacters in filenames, such as \*

filename completion (TAB, ^L)

Less can also be compiled to be permanently in "secure" mode.

COMPATIBILITY WITH MORE

If the environment variable LESS\_IS\_MORE is set to 1, or if the program

is invoked via a file link named "more", less behaves (mostly) in

conformance with the POSIX "more" command specification. In this mode,

less behaves differently in these ways:

The -e option works differently. If the -e option is not set, less

behaves as if the -e option were set. If the -e option is set, less

behaves as if the -E option were set.

The -m option works differently. If the -m option is not set, the

medium prompt is used, and it is prefixed with the string "--More--".

If the -m option is set, the short prompt is used.

The -n option acts like the -z option. The normal behavior of the -n

option is unavailable in this mode.

The parameter to the -p option is taken to be a less command rather

than a search pattern.

The LESS environment variable is ignored, and the MORE environment

variable is used in its place.

ENVIRONMENT VARIABLES

Environment variables may be specified either in the system environment

as usual, or in a lesskey(1) file. If environment variables are

defined in more than one place, variables defined in a local lesskey

file take precedence over variables defined in the system environment,

which take precedence over variables defined in the system-wide lesskey

file.

COLUMNS

Sets the number of columns on the screen. Takes precedence over

the number of columns specified by the TERM variable. (But if

you have a windowing system which supports TIOCGWINSZ or

WIOCGETD, the window system's idea of the screen size takes

precedence over the LINES and COLUMNS environment variables.)

EDITOR The name of the editor (used for the v command).

HOME Name of the user's home directory (used to find a lesskey file

on Unix and OS/2 systems).

HOMEDRIVE, HOMEPATH

Concatenation of the HOMEDRIVE and HOMEPATH environment

variables is the name of the user's home directory if the HOME

variable is not set (only in the Windows version).

INIT Name of the user's init directory (used to find a lesskey file

on OS/2 systems).

LANG Language for determining the character set.

LC\_CTYPE

Language for determining the character set.

LESS Options which are passed to less automatically.

LESSANSIENDCHARS

Characters which may end an ANSI color escape sequence (default

"m").

LESSANSIMIDCHARS

Characters which may appear between the ESC character and the

end character in an ANSI color escape sequence (default

"0123456789:;[?!"'#%()\*+ ".

LESSBINFMT

Format for displaying non-printable, non-control characters.

LESSCHARDEF

Defines a character set.

LESSCHARSET

Selects a predefined character set.

LESSCLOSE

Command line to invoke the (optional) input-postprocessor.

LESSECHO

Name of the lessecho program (default "lessecho"). The lessecho

program is needed to expand metacharacters, such as \* and ?, in

filenames on Unix systems.

LESSEDIT

Editor prototype string (used for the v command). See

discussion under PROMPTS.

LESSGLOBALTAGS

Name of the command used by the -t option to find global tags.

Normally should be set to "global" if your system has the

global(1) command. If not set, global tags are not used.

LESSHISTFILE

Name of the history file used to remember search commands and

shell commands between invocations of less. If set to "-" or

"/dev/null", a history file is not used. The default is

"$HOME/.lesshst" on Unix systems, "$HOME/\_lesshst" on DOS and

Windows systems, or "$HOME/lesshst.ini" or "$INIT/lesshst.ini"

on OS/2 systems.

LESSHISTSIZE

The maximum number of commands to save in the history file. The

default is 100.

LESSKEY

Name of the default lesskey(1) file.

LESSKEY\_SYSTEM

Name of the default system-wide lesskey(1) file.

LESSMETACHARS

List of characters which are considered "metacharacters" by the

shell.

LESSMETAESCAPE

Prefix which less will add before each metacharacter in a

command sent to the shell. If LESSMETAESCAPE is an empty

string, commands containing metacharacters will not be passed to

the shell.

LESSOPEN

Command line to invoke the (optional) input-preprocessor.

LESSSECURE

Runs less in "secure" mode. See discussion under SECURITY.

LESSSEPARATOR

String to be appended to a directory name in filename

completion.

LESSUTFBINFMT

Format for displaying non-printable Unicode code points.

LESS\_IS\_MORE

Emulate the more(1) command.

LINES Sets the number of lines on the screen. Takes precedence over

the number of lines specified by the TERM variable. (But if you

have a windowing system which supports TIOCGWINSZ or WIOCGETD,

the window system's idea of the screen size takes precedence

over the LINES and COLUMNS environment variables.)

MORE Options which are passed to less automatically when running in

more compatible mode.

PATH User's search path (used to find a lesskey file on MS-DOS and

OS/2 systems).

SHELL The shell used to execute the ! command, as well as to expand

filenames.

TERM The type of terminal on which less is being run.

VISUAL The name of the editor (used for the v command).

SEE ALSO

lesskey(1)

COPYRIGHT

Copyright (C) 1984-2021 Mark Nudelman

less is part of the GNU project and is free software. You can

redistribute it and/or modify it under the terms of either (1) the GNU

General Public License as published by the Free Software Foundation; or

(2) the Less License. See the file README in the less distribution for

more details regarding redistribution. You should have received a copy

of the GNU General Public License along with the source for less; see

the file COPYING. If not, write to the Free Software Foundation, 59

Temple Place, Suite 330, Boston, MA 02111-1307, USA. You should also

have received a copy of the Less License; see the file LICENSE.

less is distributed in the hope that it will be useful, but WITHOUT ANY

WARRANTY; without even the implied warranty of MERCHANTABILITY or

FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License

for more details.

NAME

cp – copy files

SYNOPSIS

cp [-R [-H | -L | -P]] [-f | -i | -n] [-alpsvx] source\_file target\_file

cp [-R [-H | -L | -P]] [-f | -i | -n] [-alpsvx]

source\_file ... target\_directory

DESCRIPTION

In the first synopsis form, the cp utility copies the contents of the

source\_file to the target\_file. In the second synopsis form, the

contents of each named source\_file is copied to the destination

target\_directory. The names of the files themselves are not changed. If

cp detects an attempt to copy a file to itself, the copy will fail.

The following options are available:

-H If the -R option is specified, symbolic links on the command line

are followed. (Symbolic links encountered in the tree traversal

are not followed.)

-L If the -R option is specified, all symbolic links are followed.

-P If the -R option is specified, no symbolic links are followed.

This is the default.

-R If source\_file designates a directory, cp copies the directory and

the entire subtree connected at that point. If the source\_file

ends in a /, the contents of the directory are copied rather than

the directory itself. This option also causes symbolic links to be

copied, rather than indirected through, and for cp to create

special files rather than copying them as normal files. Created

directories have the same mode as the corresponding source

directory, unmodified by the process' umask.

Note that cp copies hard linked files as separate files. If you

need to preserve hard links, consider using tar(1), cpio(1), or

pax(1) instead.

-a Archive mode. Same as -RpP.

-f For each existing destination pathname, remove it and create a new

file, without prompting for confirmation regardless of its

permissions. (The -f option overrides any previous -i or -n

options.)

-i Cause cp to write a prompt to the standard error output before

copying a file that would overwrite an existing file. If the

response from the standard input begins with the character ‘y’ or

‘Y’, the file copy is attempted. (The -i option overrides any

previous -f or -n options.)

-l Create hard links to regular files in a hierarchy instead of

copying.

-n Do not overwrite an existing file. (The -n option overrides any

previous -f or -i options.)

-p Cause cp to preserve the following attributes of each source file

in the copy: modification time, access time, file flags, file mode,

ACL, user ID, and group ID, as allowed by permissions.

If the user ID and group ID cannot be preserved, no error message

is displayed and the exit value is not altered.

If the source file has its set-user-ID bit on and the user ID

cannot be preserved, the set-user-ID bit is not preserved in the

copy's permissions. If the source file has its set-group-ID bit on

and the group ID cannot be preserved, the set-group-ID bit is not

preserved in the copy's permissions. If the source file has both

its set-user-ID and set-group-ID bits on, and either the user ID or

group ID cannot be preserved, neither the set-user-ID nor set-

group-ID bits are preserved in the copy's permissions.

-s Create symbolic links to regular files in a hierarchy instead of

copying.

-v Cause cp to be verbose, showing files as they are copied.

-x File system mount points are not traversed.

For each destination file that already exists, its contents are

overwritten if permissions allow. Its mode, user ID, and group ID are

unchanged unless the -p option was specified.

In the second synopsis form, target\_directory must exist unless there is

only one named source\_file which is a directory and the -R flag is

specified.

If the destination file does not exist, the mode of the source file is

used as modified by the file mode creation mask (umask, see csh(1)). If

the source file has its set-user-ID bit on, that bit is removed unless

both the source file and the destination file are owned by the same user.

If the source file has its set-group-ID bit on, that bit is removed

unless both the source file and the destination file are in the same

group and the user is a member of that group. If both the set-user-ID

and set-group-ID bits are set, all of the above conditions must be

fulfilled or both bits are removed.

Appropriate permissions are required for file creation or overwriting.

Symbolic links are always followed unless the -R flag is set, in which

case symbolic links are not followed, by default. The -H or -L flags (in

conjunction with the -R flag) cause symbolic links to be followed as

described above. The -H, -L and -P options are ignored unless the -R

option is specified. In addition, these options override each other and

the command's actions are determined by the last one specified.

If cp receives a SIGINFO (see the status argument for stty(1)) signal,

the current input and output file and the percentage complete will be

written to the standard output.

EXIT STATUS

The cp utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Make a copy of file foo named bar:

$ cp foo bar

Copy a group of files to the /tmp directory:

$ cp \*.txt /tmp

Copy the directory junk and all of its contents (including any

subdirectories) to the /tmp directory:

$ cp -R junk /tmp

COMPATIBILITY

Historic versions of the cp utility had a -r option. This implementation

supports that option, however, its behavior is different from historical

FreeBSD behavior. Use of this option is strongly discouraged as the

behavior is implementation-dependent. In FreeBSD, -r is a synonym for

-RL and works the same unless modified by other flags. Historical

implementations of -r differ as they copy special files as normal files

while recreating a hierarchy.

The -l, -s, -v, -x and -n options are non-standard and their use in

scripts is not recommended.

SEE ALSO

mv(1), rcp(1), umask(2), fts(3), symlink(7)

STANDARDS

The cp command is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

rm, unlink – remove directory entries

SYNOPSIS

rm [-f | -i] [-dIRrvWx] file ...

unlink [--] file

DESCRIPTION

The rm utility attempts to remove the non-directory type files specified

on the command line. If the permissions of the file do not permit

writing, and the standard input device is a terminal, the user is

prompted (on the standard error output) for confirmation.

The options are as follows:

-d Attempt to remove directories as well as other types of files.

-f Attempt to remove the files without prompting for confirmation,

regardless of the file's permissions. If the file does not

exist, do not display a diagnostic message or modify the exit

status to reflect an error. The -f option overrides any previous

-i options.

-i Request confirmation before attempting to remove each file,

regardless of the file's permissions, or whether or not the

standard input device is a terminal. The -i option overrides any

previous -f options.

-I Request confirmation once if more than three files are being

removed or if a directory is being recursively removed. This is

a far less intrusive option than -i yet provides almost the same

level of protection against mistakes.

-P This flag has no effect. It is kept only for backwards

compatibility with 4.4BSD-Lite2.

-R Attempt to remove the file hierarchy rooted in each file

argument. The -R option implies the -d option. If the -i option

is specified, the user is prompted for confirmation before each

directory's contents are processed (as well as before the attempt

is made to remove the directory). If the user does not respond

affirmatively, the file hierarchy rooted in that directory is

skipped.

-r Equivalent to -R.

-v Be verbose when deleting files, showing them as they are removed.

-W Attempt to undelete the named files. Currently, this option can

only be used to recover files covered by whiteouts in a union

file system (see undelete(2)).

-x When removing a hierarchy, do not cross mount points.

The rm utility removes symbolic links, not the files referenced by the

links.

It is an error to attempt to remove the files /, . or ...

When the utility is called as unlink, only one argument, which must not

be a directory, may be supplied. No options may be supplied in this

simple mode of operation, which performs an unlink(2) operation on the

passed argument. However, the usual option-end delimiter, --, may

optionally precede the argument.

EXIT STATUS

The rm utility exits 0 if all of the named files or file hierarchies were

removed, or if the -f option was specified and all of the existing files

or file hierarchies were removed. If an error occurs, rm exits with a

value >0.

NOTES

The rm command uses getopt(3) to parse its arguments, which allows it to

accept the ‘--’ option which will cause it to stop processing flag

options at that point. This will allow the removal of file names that

begin with a dash (‘-’). For example:

rm -- -filename

The same behavior can be obtained by using an absolute or relative path

reference. For example:

rm /home/user/-filename

rm ./-filename

EXAMPLES

Recursively remove all files contained within the foobar directory

hierarchy:

$ rm -rf foobar

Any of these commands will remove the file -f:

$ rm -- -f

$ rm ./-f

$ unlink -f

COMPATIBILITY

The rm utility differs from historical implementations in that the -f

option only masks attempts to remove non-existent files instead of

masking a large variety of errors. The -v option is non-standard and its

use in scripts is not recommended.

Also, historical BSD implementations prompted on the standard output, not

the standard error output.

The -P option does not have any effect as of FreeBSD 13 and may be

removed in the future.

SEE ALSO

chflags(1), rmdir(1), undelete(2), unlink(2), fts(3), getopt(3),

symlink(7)

STANDARDS

The rm command conforms to IEEE Std 1003.1-2008, 2013 Edition

(“POSIX.1”).

The simplified unlink command conforms to Version 2 of the Single UNIX

Specification (“SUSv2”).

NAME

rmdir – remove directories

SYNOPSIS

rmdir [-pv] directory ...

DESCRIPTION

The rmdir utility removes the directory entry specified by each directory

argument, provided it is empty.

Arguments are processed in the order given. In order to remove both a

parent directory and a subdirectory of that parent, the subdirectory must

be specified first so the parent directory is empty when rmdir tries to

remove it.

The following option is available:

-p Each directory argument is treated as a pathname of which all

components will be removed, if they are empty, starting with the

last most component. (See rm(1) for fully non-discriminant

recursive removal.)

-v Be verbose, listing each directory as it is removed.

EXIT STATUS

The rmdir utility exits with one of the following values:

0 Each directory entry specified by a directory operand referred to

an empty directory and was removed successfully.

>0 An error occurred.

EXAMPLES

Remove the directory foobar, if it is empty:

$ rmdir foobar

Remove all directories up to and including cow, stopping at the first

non-empty directory (if any):

$ rmdir -p cow/horse/monkey

SEE ALSO

rm(1)

STANDARDS

The rmdir utility is expected to be IEEE Std 1003.2 (“POSIX.2”)

compatible.

NAME

mv – move files

SYNOPSIS

mv [-f | -i | -n] [-hv] source target

mv [-f | -i | -n] [-v] source ... directory

DESCRIPTION

In its first form, the mv utility renames the file named by the source

operand to the destination path named by the target operand. This form

is assumed when the last operand does not name an already existing

directory.

In its second form, mv moves each file named by a source operand to a

destination file in the existing directory named by the directory

operand. The destination path for each operand is the pathname produced

by the concatenation of the last operand, a slash, and the final pathname

component of the named file.

The following options are available:

-f Do not prompt for confirmation before overwriting the destination

path. (The -f option overrides any previous -i or -n options.)

-h If the target operand is a symbolic link to a directory, do not

follow it. This causes the mv utility to rename the file source

to the destination path target rather than moving source into the

directory referenced by target.

-i Cause mv to write a prompt to standard error before moving a file

that would overwrite an existing file. If the response from the

standard input begins with the character ‘y’ or ‘Y’, the move is

attempted. (The -i option overrides any previous -f or -n

options.)

-n Do not overwrite an existing file. (The -n option overrides any

previous -f or -i options.)

-v Cause mv to be verbose, showing files after they are moved.

It is an error for the source operand to specify a directory if the

target exists and is not a directory.

If the destination path does not have a mode which permits writing, mv

prompts the user for confirmation as specified for the -i option.

As the rename(2) call does not work across file systems, mv uses cp(1)

and rm(1) to accomplish the move. The effect is equivalent to:

rm -f destination\_path && \

cp -pRP source\_file destination && \

rm -rf source\_file

EXIT STATUS

The mv utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Rename file foo to bar, overwriting bar if it already exists:

$ mv -f foo bar

COMPATIBILITY

The -h, -n, and -v options are non-standard and their use in scripts is

not recommended.

SEE ALSO

cp(1), rm(1), symlink(7)

STANDARDS

The mv utility is expected to be IEEE Std 1003.2 (“POSIX.2”) compatible.

NAME

chmod – change file modes

SYNOPSIS

chmod [-fhv] [-R [-H | -L | -P]] mode file ...

DESCRIPTION

The chmod utility modifies the file mode bits of the listed files as

specified by the mode operand.

The options are as follows:

-f Do not display a diagnostic message if chmod could not modify the

mode for file, nor modify the exit status to reflect such

failures.

-H If the -R option is specified, symbolic links on the command line

are followed and hence unaffected by the command. (Symbolic

links encountered during tree traversal are not followed.)

-h If the file is a symbolic link, change the mode of the link

itself rather than the file that the link points to.

-L If the -R option is specified, all symbolic links are followed.

-P If the -R option is specified, no symbolic links are followed.

This is the default.

-R Change the modes of the file hierarchies rooted in the files,

instead of just the files themselves. Beware of unintentionally

matching the “..” hard link to the parent directory when using

wildcards like “.\*”.

-v Cause chmod to be verbose, showing filenames as the mode is

modified. If the -v flag is specified more than once, the old

and new modes of the file will also be printed, in both octal and

symbolic notation.

The -H, -L and -P options are ignored unless the -R option is specified.

In addition, these options override each other and the command's actions

are determined by the last one specified.

If chmod receives a SIGINFO signal (see the status argument for stty(1)),

then the current filename as well as the old and new modes are displayed.

Only the owner of a file or the super-user is permitted to change the

mode of a file.

EXIT STATUS

The chmod utility exits 0 on success, and >0 if an error occurs.

MODES

Modes may be absolute or symbolic. An absolute mode is an octal number

constructed from the sum of one or more of the following values:

4000 (the setuid bit). Executable files with this bit set will

run with effective uid set to the uid of the file owner.

Directories with this bit set will force all files and sub-

directories created in them to be owned by the directory

owner and not by the uid of the creating process, if the

underlying file system supports this feature: see chmod(2)

and the suiddir option to mount(8).

2000 (the setgid bit). Executable files with this bit set will

run with effective gid set to the gid of the file owner.

1000 (the sticky bit). See chmod(2) and sticky(7).

0400 Allow read by owner.

0200 Allow write by owner.

0100 For files, allow execution by owner. For directories,

allow the owner to search in the directory.

0040 Allow read by group members.

0020 Allow write by group members.

0010 For files, allow execution by group members. For

directories, allow group members to search in the

directory.

0004 Allow read by others.

0002 Allow write by others.

0001 For files, allow execution by others. For directories

allow others to search in the directory.

For example, the absolute mode that permits read, write and execute by

the owner, read and execute by group members, read and execute by others,

and no set-uid or set-gid behaviour is 755 (400+200+100+040+010+004+001).

The symbolic mode is described by the following grammar:

mode ::= clause [, clause ...]

clause ::= [who ...] [action ...] action

action ::= op [perm ...]

who ::= a | u | g | o

op ::= + | - | =

perm ::= r | s | t | w | x | X | u | g | o

The who symbols ``u'', ``g'', and ``o'' specify the user, group, and

other parts of the mode bits, respectively. The who symbol ``a'' is

equivalent to ``ugo''.

The perm symbols represent the portions of the mode bits as follows:

r The read bits.

s The set-user-ID-on-execution and set-group-ID-on-execution

bits.

t The sticky bit.

w The write bits.

x The execute/search bits.

X The execute/search bits if the file is a directory or any

of the execute/search bits are set in the original

(unmodified) mode. Operations with the perm symbol ``X''

are only meaningful in conjunction with the op symbol

``+'', and are ignored in all other cases.

u The user permission bits in the original mode of the file.

g The group permission bits in the original mode of the file.

o The other permission bits in the original mode of the file.

The op symbols represent the operation performed, as follows:

+ If no value is supplied for perm, the ``+'' operation has no

effect. If no value is supplied for who, each permission bit

specified in perm, for which the corresponding bit in the file mode

creation mask (see umask(2)) is clear, is set. Otherwise, the mode

bits represented by the specified who and perm values are set.

- If no value is supplied for perm, the ``-'' operation has no

effect. If no value is supplied for who, each permission bit

specified in perm, for which the corresponding bit in the file mode

creation mask is set, is cleared. Otherwise, the mode bits

represented by the specified who and perm values are cleared.

= The mode bits specified by the who value are cleared, or, if no who

value is specified, the owner, group and other mode bits are

cleared. Then, if no value is supplied for who, each permission

bit specified in perm, for which the corresponding bit in the file

mode creation mask is clear, is set. Otherwise, the mode bits

represented by the specified who and perm values are set.

Each clause specifies one or more operations to be performed on the mode

bits, and each operation is applied to the mode bits in the order

specified.

Operations upon the other permissions only (specified by the symbol ``o''

by itself), in combination with the perm symbols ``s'' or ``t'', are

ignored.

The ``w'' permission on directories will permit file creation,

relocation, and copy into that directory. Files created within the

directory itself will inherit its group ID.

EXAMPLES

644 make a file readable by anyone and writable by the owner

only.

go-w deny write permission to group and others.

=rw,+X set the read and write permissions to the usual defaults,

but retain any execute permissions that are currently set.

+X make a directory or file searchable/executable by everyone

if it is already searchable/executable by anyone.

755

u=rwx,go=rx

u=rwx,go=u-w make a file readable/executable by everyone and writable by

the owner only.

go= clear all mode bits for group and others.

g=u-w set the group bits equal to the user bits, but clear the

group write bit.

COMPATIBILITY

The -v option is non-standard and its use in scripts is not recommended.

SEE ALSO

chflags(1), install(1), setfacl(1), chmod(2), stat(2), umask(2), fts(3),

setmode(3), sticky(7), symlink(7), chown(8), mount(8)

STANDARDS

The chmod utility is expected to be IEEE Std 1003.2 (“POSIX.2”)

compatible with the exception of the perm symbol “t” which is not

included in that standard.

HISTORY

A chmod command appeared in Version 1 AT&T UNIX.

BUGS

There is no perm option for the naughty bits of a horse.

NAME

ln, link – link files

SYNOPSIS

ln [-L | -P | -s [-F]] [-f | -iw] [-hnv] source\_file [target\_file]

ln [-L | -P | -s [-F]] [-f | -iw] [-hnv] source\_file ... target\_dir

link source\_file target\_file

DESCRIPTION

The ln utility creates a new directory entry (linked file) for the file

name specified by target\_file. The target\_file will be created with the

same file modes as the source\_file. It is useful for maintaining

multiple copies of a file in many places at once without using up storage

for the “copies”; instead, a link “points” to the original copy. There

are two types of links; hard links and symbolic links. How a link

“points” to a file is one of the differences between a hard and symbolic

link.

The options are as follows:

-F If the target file already exists and is a directory, then remove

it so that the link may occur. The -F option should be used with

either -f or -i options. If neither -f nor -i is specified, -f is

implied. The -F option is a no-op unless -s is specified.

-L When creating a hard link to a symbolic link, create a hard link to

the target of the symbolic link. This is the default. This option

cancels the -P option.

-P When creating a hard link to a symbolic link, create a hard link to

the symbolic link itself. This option cancels the -L option.

-f If the target file already exists, then unlink it so that the link

may occur. (The -f option overrides any previous -i and -w

options.)

-h If the target\_file or target\_dir is a symbolic link, do not follow

it. This is most useful with the -f option, to replace a symlink

which may point to a directory.

-i Cause ln to write a prompt to standard error if the target file

exists. If the response from the standard input begins with the

character ‘y’ or ‘Y’, then unlink the target file so that the link

may occur. Otherwise, do not attempt the link. (The -i option

overrides any previous -f options.)

-n Same as -h, for compatibility with other ln implementations.

-s Create a symbolic link.

-v Cause ln to be verbose, showing files as they are processed.

-w Warn if the source of a symbolic link does not currently exist.

By default, ln makes hard links. A hard link to a file is

indistinguishable from the original directory entry; any changes to a

file are effectively independent of the name used to reference the file.

Directories may not be hardlinked, and hard links may not span file

systems.

A symbolic link contains the name of the file to which it is linked. The

referenced file is used when an open(2) operation is performed on the

link. A stat(2) on a symbolic link will return the linked-to file; an

lstat(2) must be done to obtain information about the link. The

readlink(2) call may be used to read the contents of a symbolic link.

Symbolic links may span file systems and may refer to directories.

Given one or two arguments, ln creates a link to an existing file

source\_file. If target\_file is given, the link has that name;

target\_file may also be a directory in which to place the link; otherwise

it is placed in the current directory. If only the directory is

specified, the link will be made to the last component of source\_file.

Given more than two arguments, ln makes links in target\_dir to all the

named source files. The links made will have the same name as the files

being linked to.

When the utility is called as link, exactly two arguments must be

supplied, neither of which may specify a directory. No options may be

supplied in this simple mode of operation, which performs a link(2)

operation using the two passed arguments.

EXAMPLES

Create a symbolic link named /home/src and point it to /usr/src:

# ln -s /usr/src /home/src

Hard link /usr/local/bin/fooprog to file /usr/local/bin/fooprog-1.0:

# ln /usr/local/bin/fooprog-1.0 /usr/local/bin/fooprog

As an exercise, try the following commands:

# ls -i /bin/[

11553 /bin/[

# ls -i /bin/test

11553 /bin/test

Note that both files have the same inode; that is, /bin/[ is essentially

an alias for the test(1) command. This hard link exists so test(1) may

be invoked from shell scripts, for example, using the if [ ] construct.

In the next example, the second call to ln removes the original foo and

creates a replacement pointing to baz:

# mkdir bar baz

# ln -s bar foo

# ln -shf baz foo

Without the -h option, this would instead leave foo pointing to bar and

inside foo create a new symlink baz pointing to itself. This results

from directory-walking.

An easy rule to remember is that the argument order for ln is the same as

for cp(1): The first argument needs to exist, the second one is created.

COMPATIBILITY

The -h, -i, -n, -v and -w options are non-standard and their use in

scripts is not recommended. They are provided solely for compatibility

with other ln implementations.

The -F option is a FreeBSD extension and should not be used in portable

scripts.

SEE ALSO

link(2), lstat(2), readlink(2), stat(2), symlink(2), symlink(7)

STANDARDS

The ln utility conforms to IEEE Std 1003.2-1992 (“POSIX.2”).

The simplified link command conforms to Version 2 of the Single UNIX

Specification (“SUSv2”).

NAME

wc – word, line, character, and byte count

SYNOPSIS

wc [--libxo] [-Lclmw] [file ...]

DESCRIPTION

The wc utility displays the number of lines, words, and bytes contained

in each input file, or standard input (if no file is specified) to the

standard output. A line is defined as a string of characters delimited

by a ⟨newline⟩ character. Characters beyond the final ⟨newline⟩

character will not be included in the line count.

A word is defined as a string of characters delimited by white space

characters. White space characters are the set of characters for which

the iswspace(3) function returns true. If more than one input file is

specified, a line of cumulative counts for all the files is displayed on

a separate line after the output for the last file.

The following options are available:

--libxo

Generate output via libxo(3) in a selection of different human

and machine readable formats. See xo\_parse\_args(3) for details

on command line arguments.

-L Write the length of the line containing the most bytes (default)

or characters (when -m is provided) to standard output. When

more than one file argument is specified, the longest input line

of all files is reported as the value of the final “total”.

-c The number of bytes in each input file is written to the standard

output. This will cancel out any prior usage of the -m option.

-l The number of lines in each input file is written to the standard

output.

-m The number of characters in each input file is written to the

standard output. If the current locale does not support

multibyte characters, this is equivalent to the -c option. This

will cancel out any prior usage of the -c option.

-w The number of words in each input file is written to the standard

output.

When an option is specified, wc only reports the information requested by

that option. The order of output always takes the form of line, word,

byte, and file name. The default action is equivalent to specifying the

-c, -l and -w options.

If no files are specified, the standard input is used and no file name is

displayed. The prompt will accept input until receiving EOF, or [^D] in

most environments.

If wc receives a SIGINFO (see the status argument for stty(1)) signal,

the interim data will be written to the standard error output in the same

format as the standard completion message.

ENVIRONMENT

The LANG, LC\_ALL and LC\_CTYPE environment variables affect the execution

of wc as described in environ(7).

EXIT STATUS

The wc utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Count the number of characters, words and lines in each of the files

report1 and report2 as well as the totals for both:

wc -mlw report1 report2

Find the longest line in a list of files:

wc -L file1 file2 file3 | fgrep total

COMPATIBILITY

Historically, the wc utility was documented to define a word as a

“maximal string of characters delimited by <space>, <tab> or <newline>

characters”. The implementation, however, did not handle non-printing

characters correctly so that “  ^D^E  ” counted as 6 spaces, while

“foo^D^Ebar” counted as 8 characters. 4BSD systems after 4.3BSD modified

the implementation to be consistent with the documentation. This

implementation defines a “word” in terms of the iswspace(3) function, as

required by IEEE Std 1003.2 (“POSIX.2”).

The -L option is a non-standard FreeBSD extension, compatible with the -L

option of the GNU wc utility.

SEE ALSO

iswspace(3), libxo(3), xo\_parse\_args(3)

STANDARDS

The wc utility conforms to IEEE Std 1003.1-2001 (“POSIX.1”).

NAME

head – display first lines of a file

SYNOPSIS

head [-n count | -c bytes] [file ...]

DESCRIPTION

This filter displays the first count lines or bytes of each of the

specified files, or of the standard input if no files are specified. If

count is omitted it defaults to 10.

The following options are available:

-c bytes, --bytes=bytes

Print bytes of each of the specified files.

-n count, --lines=count

Print count lines of each of the specified files.

If more than a single file is specified, each file is preceded by a

header consisting of the string “==> XXX <==” where “XXX” is the name of

the file.

EXIT STATUS

The head utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

To display the first 500 lines of the file foo:

$ head -n 500 foo

head can be used in conjunction with tail(1) in the following way to, for

example, display only line 500 from the file foo:

$ head -n 500 foo | tail -n 1

NAME

tail – display the last part of a file

SYNOPSIS

[-F | -f | -r] [-q] [-b number | -c number | -n number] [file ...]

DESCRIPTION

The tail utility displays the contents of file or, by default, its

standard input, to the standard output.

The display begins at a byte, line or 512-byte block location in the

input. Numbers having a leading plus (‘+’) sign are relative to the

beginning of the input, for example, “-c +2” starts the display at the

second byte of the input. Numbers having a leading minus (‘-’) sign or

no explicit sign are relative to the end of the input, for example, “-n

2” displays the last two lines of the input. The default starting

location is “-n 10”, or the last 10 lines of the input.

The options are as follows:

-b number, --blocks=number

The location is number 512-byte blocks.

-c number, --bytes=number

The location is number bytes.

-f The -f option causes tail to not stop when end of file is

reached, but rather to wait for additional data to be appended to

the input. The -f option is ignored if the standard input is a

pipe, but not if it is a FIFO.

-F The -F option implies the -f option, but tail will also check to

see if the file being followed has been renamed or rotated. The

file is closed and reopened when tail detects that the filename

being read from has a new inode number.

If the file being followed does not (yet) exist or if it is

removed, tail will keep looking and will display the file from

the beginning if and when it is created.

The -F option is the same as the -f option if reading from

standard input rather than a file.

-n number, --lines=number

The location is number lines.

-q Suppresses printing of headers when multiple files are being

examined.

-r The -r option causes the input to be displayed in reverse order,

by line. Additionally, this option changes the meaning of the

-b, -c and -n options. When the -r option is specified, these

options specify the number of bytes, lines or 512-byte blocks to

display, instead of the bytes, lines or blocks from the beginning

or end of the input from which to begin the display. The default

for the -r option is to display all of the input.

If more than a single file is specified, each file is preceded by a

header consisting of the string “==> XXX <==” where XXX is the name of

the file unless -q flag is specified.

EXIT STATUS

The tail utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

To display the last 500 lines of the file foo:

$ tail -n 500 foo

Keep /var/log/messages open, displaying to the standard output anything

appended to the file:

$ tail -F /var/log/messages

SEE ALSO

cat(1), head(1), sed(1)

STANDARDS

The tail utility is expected to be a superset of the IEEE Std 1003.2-1992

(“POSIX.2”) specification. In particular, the -F, -b and -r options are

extensions to that standard.

The historic command line syntax of tail is supported by this

implementation. The only difference between this implementation and

historic versions of tail, once the command line syntax translation has

been done, is that the -b, -c and -n options modify the -r option, i.e.,

“-r -c 4” displays the last 4 characters of the last line of the input,

while the historic tail (using the historic syntax “-4cr”) would ignore

the -c option and display the last 4 lines of the input.

NAME

sort – sort or merge records (lines) of text and binary files

SYNOPSIS

sort [-bcCdfghiRMmnrsuVz] [-k field1[,field2]] [-S memsize] [-T dir] [-t

char] [-o output] [file ...]

sort --help

sort --version

DESCRIPTION

The sort utility sorts text and binary files by lines. A line is a

record separated from the subsequent record by a newline (default) or NUL

´\0´ character (-z option). A record can contain any printable or

unprintable characters. Comparisons are based on one or more sort keys

extracted from each line of input, and are performed lexicographically,

according to the current locale's collating rules and the specified

command-line options that can tune the actual sorting behavior. By

default, if keys are not given, sort uses entire lines for comparison.

The command line options are as follows:

-c, --check, -C, --check=silent|quiet

Check that the single input file is sorted. If the file is not

sorted, sort produces the appropriate error messages and exits

with code 1, otherwise returns 0. If -C or --check=silent is

specified, sort produces no output. This is a "silent" version

of -c.

-m, --merge

Merge only. The input files are assumed to be pre-sorted. If

they are not sorted the output order is undefined.

-o output, --output=output

Print the output to the output file instead of the standard

output.

-S size, --buffer-size=size

Use size for the maximum size of the memory buffer. Size

modifiers %,b,K,M,G,T,P,E,Z,Y can be used. If a memory limit is

not explicitly specified, sort takes up to about 90% of available

memory. If the file size is too big to fit into the memory

buffer, the temporary disk files are used to perform the sorting.

-T dir, --temporary-directory=dir

Store temporary files in the directory dir. The default path is

the value of the environment variable TMPDIR or /var/tmp if

TMPDIR is not defined.

-u, --unique

Unique keys. Suppress all lines that have a key that is equal to

an already processed one. This option, similarly to -s, implies

a stable sort. If used with -c or -C, sort also checks that

there are no lines with duplicate keys.

-s Stable sort. This option maintains the original record order of

records that have an equal key. This is a non-standard feature,

but it is widely accepted and used.

--version

Print the version and silently exits.

--help Print the help text and silently exits.

The following options override the default ordering rules. When ordering

options appear independently of key field specifications, they apply

globally to all sort keys. When attached to a specific key (see -k), the

ordering options override all global ordering options for the key they

are attached to.

-b, --ignore-leading-blanks

Ignore leading blank characters when comparing lines.

-d, --dictionary-order

Consider only blank spaces and alphanumeric characters in

comparisons.

-f, --ignore-case

Convert all lowercase characters to their uppercase equivalent

before comparison, that is, perform case-independent sorting.

-g, --general-numeric-sort, --sort=general-numeric

Sort by general numerical value. As opposed to -n, this option

handles general floating points. It has a more permissive format

than that allowed by -n but it has a significant performance

drawback.

-h, --human-numeric-sort, --sort=human-numeric

Sort by numerical value, but take into account the SI suffix, if

present. Sort first by numeric sign (negative, zero, or

positive); then by SI suffix (either empty, or `k' or `K', or one

of `MGTPEZY', in that order); and finally by numeric value. The

SI suffix must immediately follow the number. For example,

'12345K' sorts before '1M', because M is "larger" than K. This

sort option is useful for sorting the output of a single

invocation of 'df' command with -h or -H options (human-

readable).

-i, --ignore-nonprinting

Ignore all non-printable characters.

-M, --month-sort, --sort=month

Sort by month abbreviations. Unknown strings are considered

smaller than the month names.

-n, --numeric-sort, --sort=numeric

Sort fields numerically by arithmetic value. Fields are supposed

to have optional blanks in the beginning, an optional minus sign,

zero or more digits (including decimal point and possible

thousand separators).

-R, --random-sort, --sort=random

Sort by a random order. This is a random permutation of the

inputs except that the equal keys sort together. It is

implemented by hashing the input keys and sorting the hash

values. The hash function is chosen randomly. The hash function

is randomized by /dev/random content, or by file content if it is

specified by --random-source. Even if multiple sort fields are

specified, the same random hash function is used for all of them.

-r, --reverse

Sort in reverse order.

-V, --version-sort

Sort version numbers. The input lines are treated as file names

in form PREFIX VERSION SUFFIX, where SUFFIX matches the regular

expression "(.([A-Za-z~][A-Za-z0-9~]\*)?)\*". The files are

compared by their prefixes and versions (leading zeros are

ignored in version numbers, see example below). If an input

string does not match the pattern, then it is compared using the

byte compare function. All string comparisons are performed in C

locale, the locale environment setting is ignored.

Example:

$ ls sort\* | sort -V

sort-1.022.tgz

sort-1.23.tgz

sort-1.23.1.tgz

sort-1.024.tgz

sort-1.024.003.

sort-1.024.003.tgz

sort-1.024.07.tgz

sort-1.024.009.tgz

The treatment of field separators can be altered using these options:

-b, --ignore-leading-blanks

Ignore leading blank space when determining the start and end of

a restricted sort key (see -k). If -b is specified before the

first -k option, it applies globally to all key specifications.

Otherwise, -b can be attached independently to each field

argument of the key specifications. -b.

-k field1[,field2], --key=field1[,field2]

Define a restricted sort key that has the starting position

field1, and optional ending position field2 of a key field. The

-k option may be specified multiple times, in which case

subsequent keys are compared when earlier keys compare equal.

The -k option replaces the obsolete options +pos1 and -pos2, but

the old notation is also supported.

-t char, --field-separator=char

Use char as a field separator character. The initial char is not

considered to be part of a field when determining key offsets.

Each occurrence of char is significant (for example, “charchar”

delimits an empty field). If -t is not specified, the default

field separator is a sequence of blank space characters, and

consecutive blank spaces do not delimit an empty field, however,

the initial blank space is considered part of a field when

determining key offsets. To use NUL as field separator, use -t

´\0´.

-z, --zero-terminated

Use NUL as record separator. By default, records in the files

are supposed to be separated by the newline characters. With

this option, NUL (´\0´) is used as a record separator character.

Other options:

--batch-size=num

Specify maximum number of files that can be opened by sort at

once. This option affects behavior when having many input files

or using temporary files. The default value is 16.

--compress-program=PROGRAM

Use PROGRAM to compress temporary files. PROGRAM must compress

standard input to standard output, when called without arguments.

When called with argument -d it must decompress standard input to

standard output. If PROGRAM fails, sort must exit with error.

An example of PROGRAM that can be used here is bzip2.

--random-source=filename

In random sort, the file content is used as the source of the

'seed' data for the hash function choice. Two invocations of

random sort with the same seed data will use the same hash

function and will produce the same result if the input is also

identical. By default, file /dev/random is used.

--debug

Print some extra information about the sorting process to the

standard output.

--files0-from=filename

Take the input file list from the file filename. The file names

must be separated by NUL (like the output produced by the command

"find ... -print0").

--radixsort

Try to use radix sort, if the sort specifications allow. The

radix sort can only be used for trivial locales (C and POSIX),

and it cannot be used for numeric or month sort. Radix sort is

very fast and stable.

--mergesort

Use mergesort. This is a universal algorithm that can always be

used, but it is not always the fastest.

--qsort

Try to use quick sort, if the sort specifications allow. This

sort algorithm cannot be used with -u and -s.

--heapsort

Try to use heap sort, if the sort specifications allow. This

sort algorithm cannot be used with -u and -s.

--mmap Try to use file memory mapping system call. It may increase

speed in some cases.

The following operands are available:

file The pathname of a file to be sorted, merged, or checked. If no

file operands are specified, or if a file operand is -, the

standard input is used.

A field is defined as a maximal sequence of characters other than the

field separator and record separator (newline by default). Initial blank

spaces are included in the field unless -b has been specified; the first

blank space of a sequence of blank spaces acts as the field separator and

is included in the field (unless -t is specified). For example, all

blank spaces at the beginning of a line are considered to be part of the

first field.

Fields are specified by the -k field1[,field2] command-line option. If

field2 is missing, the end of the key defaults to the end of the line.

The arguments field1 and field2 have the form m.n (m,n > 0) and can be

followed by one or more of the modifiers b, d, f, i, n, g, M and r, which

correspond to the options discussed above. When b is specified it

applies only to field1 or field2 where it is specified while the rest of

the modifiers apply to the whole key field regardless if they are

specified only with field1 or field2 or both. A field1 position

specified by m.n is interpreted as the nth character from the beginning

of the mth field. A missing .n in field1 means ‘.1’, indicating the

first character of the mth field; if the -b option is in effect, n is

counted from the first non-blank character in the mth field; m.1b refers

to the first non-blank character in the mth field. 1.n refers to the nth

character from the beginning of the line; if n is greater than the length

of the line, the field is taken to be empty.

nth positions are always counted from the field beginning, even if the

field is shorter than the number of specified positions. Thus, the key

can really start from a position in a subsequent field.

A field2 position specified by m.n is interpreted as the nth character

(including separators) from the beginning of the mth field. A missing .n

indicates the last character of the mth field; m = 0 designates the end

of a line. Thus the option -k v.x,w.y is synonymous with the obsolete

option +v-1.x-1 -w-1.y; when y is omitted, -k v.x,w is synonymous with

+v-1.x-1 -w.0. The obsolete +pos1 -pos2 option is still supported,

except for -w.0b, which has no -k equivalent.

ENVIRONMENT

LC\_COLLATE Locale settings to be used to determine the collation for

sorting records.

LC\_CTYPE Locale settings to be used to case conversion and

classification of characters, that is, which characters are

considered whitespaces, etc.

LC\_MESSAGES

Locale settings that determine the language of output

messages that sort prints out.

LC\_NUMERIC Locale settings that determine the number format used in

numeric sort.

LC\_TIME Locale settings that determine the month format used in month

sort.

LC\_ALL Locale settings that override all of the above locale

settings. This environment variable can be used to set all

these settings to the same value at once.

LANG Used as a last resort to determine different kinds of locale-

specific behavior if neither the respective environment

variable, nor LC\_ALL are set.

NLSPATH Path to NLS catalogs.

TMPDIR Path to the directory in which temporary files will be

stored. Note that TMPDIR may be overridden by the -T option.

GNUSORT\_NUMERIC\_COMPATIBILITY

If defined -t will not override the locale numeric symbols,

that is, thousand separators and decimal separators. By

default, if we specify -t with the same symbol as the

thousand separator or decimal point, the symbol will be

treated as the field separator. Older behavior was less

definite; the symbol was treated as both field separator and

numeric separator, simultaneously. This environment variable

enables the old behavior.

FILES

/var/tmp/.bsdsort.PID.\* Temporary files.

/dev/random Default seed file for the random sort.

EXIT STATUS

The sort utility shall exit with one of the following values:

0 Successfully sorted the input files or if used with -c or -C, the

input file already met the sorting criteria.

1 On disorder (or non-uniqueness) with the -c or -C options.

2 An error occurred.

SEE ALSO

comm(1), join(1), uniq(1)

STANDARDS

The sort utility is compliant with the IEEE Std 1003.1-2008 (“POSIX.1”)

specification.

The flags [-ghRMSsTVz] are extensions to the POSIX specification.

All long options are extensions to the specification, some of them are

provided for compatibility with GNU versions and some of them are own

extensions.

The old key notations +pos1 and -pos2 come from older versions of sort

and are still supported but their use is highly discouraged.

HISTORY

A sort command first appeared in Version 1 AT&T UNIX.

AUTHORS

Gabor Kovesdan <gabor@FreeBSD.org>,

Oleg Moskalenko <mom040267@gmail.com>

NOTES

This implementation of sort has no limits on input line length (other

than imposed by available memory) or any restrictions on bytes allowed

within lines.

The performance depends highly on locale settings, efficient choice of

sort keys and key complexity. The fastest sort is with locale C, on

whole lines, with option -s. In general, locale C is the fastest, then

single-byte locales follow and multi-byte locales as the slowest but the

correct collation order is always respected. As for the key

specification, the simpler to process the lines the faster the search

will be.

When sorting by arithmetic value, using -n results in much better

performance than -g so its use is encouraged whenever possible.

NAME

grep, egrep, fgrep, rgrep

– file pattern searcher

SYNOPSIS

grep [-abcdDEFGHhIiLlmnOopqRSsUVvwxz] [-A num] [-B num] [-C[num]]

[-e pattern] [-f file] [--binary-files=value] [--color[=when]]

[--colour[=when]] [--context[=num]] [--label] [--line-buffered]

[--null] [pattern] [file ...]

DESCRIPTION

The grep utility searches any given input files, selecting lines that

match one or more patterns. By default, a pattern matches an input line

if the regular expression (RE) in the pattern matches the input line

without its trailing newline. An empty expression matches every line.

Each input line that matches at least one of the patterns is written to

the standard output.

grep is used for simple patterns and basic regular expressions (BREs);

egrep can handle extended regular expressions (EREs). See re\_format(7)

for more information on regular expressions. fgrep is quicker than both

grep and egrep, but can only handle fixed patterns (i.e., it does not

interpret regular expressions). Patterns may consist of one or more

lines, allowing any of the pattern lines to match a portion of the input.

The following options are available:

-A num, --after-context=num

Print num lines of trailing context after each match. See also

the -B and -C options.

-a, --text

Treat all files as ASCII text. Normally grep will simply print

“Binary file ... matches” if files contain binary characters.

Use of this option forces grep to output lines matching the

specified pattern.

-B num, --before-context=num

Print num lines of leading context before each match. See also

the -A and -C options.

-b, --byte-offset

The offset in bytes of a matched pattern is displayed in front of

the respective matched line.

-C[num], --context[=num]

Print num lines of leading and trailing context surrounding each

match. The default value of num is “2” and is equivalent to “-A

2 -B 2”. Note: no whitespace may be given between the option and

its argument.

-c, --count

Only a count of selected lines is written to standard output.

--colour=[when], --color=[when]

Mark up the matching text with the expression stored in the

GREP\_COLOR environment variable. The possible values of when are

“never”, “always” and “auto”.

-D action, --devices=action

Specify the demanded action for devices, FIFOs and sockets. The

default action is “read”, which means, that they are read as if

they were normal files. If the action is set to “skip”, devices

are silently skipped.

-d action, --directories=action

Specify the demanded action for directories. It is “read” by

default, which means that the directories are read in the same

manner as normal files. Other possible values are “skip” to

silently ignore the directories, and “recurse” to read them

recursively, which has the same effect as the -R and -r option.

-E, --extended-regexp

Interpret pattern as an extended regular expression (i.e., force

grep to behave as egrep).

-e pattern, --regexp=pattern

Specify a pattern used during the search of the input: an input

line is selected if it matches any of the specified patterns.

This option is most useful when multiple -e options are used to

specify multiple patterns, or when a pattern begins with a dash

(‘-’).

--exclude pattern

If specified, it excludes files matching the given filename

pattern from the search. Note that --exclude and --include

patterns are processed in the order given. If a name matches

multiple patterns, the latest matching rule wins. If no

--include pattern is specified, all files are searched that are

not excluded. Patterns are matched to the full path specified,

not only to the filename component.

--exclude-dir pattern

If -R is specified, it excludes directories matching the given

filename pattern from the search. Note that --exclude-dir and

--include-dir patterns are processed in the order given. If a

name matches multiple patterns, the latest matching rule wins.

If no --include-dir pattern is specified, all directories are

searched that are not excluded.

-F, --fixed-strings

Interpret pattern as a set of fixed strings (i.e., force grep to

behave as fgrep).

-f file, --file=file

Read one or more newline separated patterns from file. Empty

pattern lines match every input line. Newlines are not

considered part of a pattern. If file is empty, nothing is

matched.

-G, --basic-regexp

Interpret pattern as a basic regular expression (i.e., force grep

to behave as traditional grep).

-H Always print filename headers with output lines.

-h, --no-filename

Never print filename headers (i.e., filenames) with output lines.

--help Print a brief help message.

-I Ignore binary files. This option is equivalent to the

“--binary-file=without-match” option.

-i, --ignore-case

Perform case insensitive matching. By default, grep is case

sensitive.

--include pattern

If specified, only files matching the given filename pattern are

searched. Note that --include and --exclude patterns are

processed in the order given. If a name matches multiple

patterns, the latest matching rule wins. Patterns are matched to

the full path specified, not only to the filename component.

--include-dir pattern

If -R is specified, only directories matching the given filename

pattern are searched. Note that --include-dir and --exclude-dir

patterns are processed in the order given. If a name matches

multiple patterns, the latest matching rule wins.

-L, --files-without-match

Only the names of files not containing selected lines are written

to standard output. Pathnames are listed once per file searched.

If the standard input is searched, the string “(standard input)”

is written unless a --label is specified.

-l, --files-with-matches

Only the names of files containing selected lines are written to

standard output. grep will only search a file until a match has

been found, making searches potentially less expensive.

Pathnames are listed once per file searched. If the standard

input is searched, the string “(standard input)” is written

unless a --label is specified.

--label

Label to use in place of “(standard input)” for a file name where

a file name would normally be printed. This option applies to

-H, -L, and -l.

--mmap Use mmap(2) instead of read(2) to read input, which can result in

better performance under some circumstances but can cause

undefined behaviour.

-m num, --max-count=num

Stop reading the file after num matches.

-n, --line-number

Each output line is preceded by its relative line number in the

file, starting at line 1. The line number counter is reset for

each file processed. This option is ignored if -c, -L, -l, or -q

is specified.

--null Prints a zero-byte after the file name.

-O If -R is specified, follow symbolic links only if they were

explicitly listed on the command line. The default is not to

follow symbolic links.

-o, --only-matching

Prints only the matching part of the lines.

-p If -R is specified, no symbolic links are followed. This is the

default.

-q, --quiet, --silent

Quiet mode: suppress normal output. grep will only search a file

until a match has been found, making searches potentially less

expensive.

-R, -r, --recursive

Recursively search subdirectories listed. (i.e., force grep to

behave as rgrep).

-S If -R is specified, all symbolic links are followed. The default

is not to follow symbolic links.

-s, --no-messages

Silent mode. Nonexistent and unreadable files are ignored (i.e.,

their error messages are suppressed).

-U, --binary

Search binary files, but do not attempt to print them.

-u This option has no effect and is provided only for compatibility

with GNU grep.

-V, --version

Display version information and exit.

-v, --invert-match

Selected lines are those not matching any of the specified

patterns.

-w, --word-regexp

The expression is searched for as a word (as if surrounded by

‘[[:<:]]’ and ‘[[:>:]]’; see re\_format(7)). This option has no

effect if -x is also specified.

-x, --line-regexp

Only input lines selected against an entire fixed string or

regular expression are considered to be matching lines.

-y Equivalent to -i. Obsoleted.

-z, --null-data

Treat input and output data as sequences of lines terminated by a

zero-byte instead of a newline.

--binary-files=value

Controls searching and printing of binary files. Options are:

binary (default) Search binary files but do not print them.

without-match Do not search binary files.

text Treat all files as text.

--line-buffered

Force output to be line buffered. By default, output is line

buffered when standard output is a terminal and block buffered

otherwise.

If no file arguments are specified, the standard input is used.

Additionally, “-” may be used in place of a file name, anywhere that a

file name is accepted, to read from standard input. This includes both

-f and file arguments.

EXIT STATUS

The grep utility exits with one of the following values:

0 One or more lines were selected.

1 No lines were selected.

>1 An error occurred.

EXAMPLES

- Find all occurrences of the pattern ‘patricia’ in a file:

$ grep 'patricia' myfile

- Same as above but looking only for complete words:

$ grep -w 'patricia' myfile

- Count occurrences of the exact pattern ‘FOO’ :

$ grep -c FOO myfile

- Same as above but ignoring case:

$ grep -c -i FOO myfile

- Find all occurrences of the pattern ‘.Pp’ at the beginning of a line:

$ grep '^\.Pp' myfile

The apostrophes ensure the entire expression is evaluated by grep

instead of by the user's shell. The caret ‘^’ matches the null

string at the beginning of a line, and the ‘\’ escapes the ‘.’, which

would otherwise match any character.

- Find all lines in a file which do not contain the words ‘foo’ or

‘bar’:

$ grep -v -e 'foo' -e 'bar' myfile

- Peruse the file ‘calendar’ looking for either 19, 20, or 25 using

extended regular expressions:

$ egrep '19|20|25' calendar

- Show matching lines and the name of the ‘\*.h’ files which contain the

pattern ‘FIXME’. Do the search recursively from the /usr/src/sys/arm

directory

$ grep -H -R FIXME --include=\*.h /usr/src/sys/arm/

- Same as above but show only the name of the matching file:

$ grep -l -R FIXME --include=\*.h /usr/src/sys/arm/

- Show lines containing the text ‘foo’. The matching part of the

output is colored and every line is prefixed with the line number and

the offset in the file for those lines that matched.

$ grep -b --colour -n foo myfile

- Show lines that match the extended regular expression patterns read

from the standard input:

$ echo -e 'Free\nBSD\nAll.\*reserved' | grep -E -f - myfile

- Show lines from the output of the pciconf(8) command matching the

specified extended regular expression along with three lines of

leading context and one line of trailing context:

$ pciconf -lv | grep -B3 -A1 -E 'class.\*=.\*storage'

- Suppress any output and use the exit status to show an appropriate

message:

$ grep -q foo myfile && echo File matches

SEE ALSO

ed(1), ex(1), sed(1), zgrep(1), re\_format(7)

STANDARDS

The grep utility is compliant with the IEEE Std 1003.1-2008 (“POSIX.1”)

specification.

The flags [-AaBbCDdGHhILmoPRSUVw] are extensions to that specification,

and the behaviour of the -f flag when used with an empty pattern file is

left undefined.

All long options are provided for compatibility with GNU versions of this

utility.

Historic versions of the grep utility also supported the flags [-ruy].

This implementation supports those options; however, their use is

strongly discouraged.

HISTORY

The grep command first appeared in Version 6 AT&T UNIX.