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 systems which support **poll**(2) you can also use ^X or the character specified by the --intr option. If the input is a pipe and the --exit-follow-on-close option is in effect, **less** will automatically stop waiting for data when the input side of the pipe is closed.

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 bottom 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. By default, searching is case-sensitive (uppercase and lowercase are considered different); the -i option can be used to change this. 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.

^S Followed by a digit N between 1 and 5. Only text which has a non-empty match for the N-th parenthesized SUB-PATTERN will be considered to match the pattern. (Supported only if **less** is built with one of the regular expression libraries **posix**, **pcre**, or **pcre2**.) Multiple ^S modifiers can be specified, to match more than one sub-pattern.

^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. If the ^W modifier is set, the ^E modifier is ignored.

?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.

^S 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 six 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.

shell-command
Similar to the "!" command, except that the command is expanded in the same way as prompt strings. For example, the name of the current file would be given as "%f".

| <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 works only if the input is a pipe, not an ordinary file.

^X When the "Waiting for data" message is displayed, such as while in the F command, pressing ^X will stop **less** from waiting and return to a prompt. This may cause **less** to think that the file ends at the current position, so it may be necessary to use the R or F command to see more data. The --intr option can be used to specify a different character to use instead of ^X. This command works only on systems that support the **poll**(2) function. On systems without **poll**(2), the interrupt character (usually ^C) can be used instead.

OPTIONS
Command line options are described below. Most options may be changed while **less** is running, via the "-" command.

Some 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 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.

-b*n* 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. Lost characters are displayed as question marks.

-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.

-D**x***color* or --color=**x***color*

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.

H Header lines and columns, set via the --header option.

M Mark letters in the status column.

N Line numbers enabled via the -N option.

P Prompts.

R The rscroll character.

S Search results.

1-5 The text in a search result which matches the first through fifth parenthesized sub-pattern. Sub-pattern coloring works only if **less** is built with one of the regular expression libraries **posix**, **pcre**, or **pcre2**.

W The highlight enabled via the -w option.

d Bold text.

k Blinking text.

s Standout text.

u Underlined text.

The uppercase letters and digits 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 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 uppercase 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) 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).

On MS-DOS only, the -Da option may be used to specify strict parsing of ANSI color (SGR) sequences when the -R option is used. Without this option, sequences that change text attributes (bold, underline, etc.) may clear the text color.

-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.

-h*n* 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.

-j*n* 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. 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 character displayed in the status column may be one of:

> The line is chopped with the -S option, and the text that is chopped off beyond the right edge of the screen contains a match for the current search.

< The line is horizontally shifted, and the text that is shifted beyond the left side of the screen contains a match for the current search.

= The line is both chopped and shifted, and there are matches beyond both sides of the screen.

* There are matches in the visible part of the line but none to the right or left of it.

a-z, A-Z
The line has been marked with the corresponding letter via the m command.

-k*filename* or --lesskey-file=*filename*
Causes **less** to open and interpret the named file as a **lesskey**(1) binary 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.

--lesskey-src=*filename*
Causes **less** to open and interpret the named file as a **lesskey**(1) source file. If the LESSKEYIN or LESSKEYIN_SYSTEM environment variable is set, or if a lesskey source file is found in a standard place (see KEY BINDINGS), it is also used as a *lesskey source* file. Prior to version 582, the **lesskey** program needed to be run to convert a *lesskey source* file to a *lesskey binary* file for **less** to use. Newer versions of **less** read the *lesskey source* file directly and ignore the binary file if the source file exists.

-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**(1)), 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**(1).

-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.

-o*filename* 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.

-O*filename* 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**.

-p*pattern* 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.

-P*prompt* 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" (with some exceptions as described under the -U option). 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. See also the --wordwrap option.

-t*tag* 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**.

-T*tagsfile* 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.

See also the --proc-backspace, --proc-tab, and --proc-return options.

-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. If the --status-line option is in effect, the entire line (the width of the screen) is highlighted. Otherwise, only the text in the 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.

-x*n*,... 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.

-y*n* 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.

-z*n* 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**(1). 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.

--exit-follow-on-close

When using the "F" command on a pipe, **less** will automatically stop waiting for more data when the input side of the pipe is closed.

--file-size

If --file-size is specified, **less** will determine the size of the file immediately after opening the file. Then the "=" command will display the number of lines in the file. Normally this is not done, because it can be slow if the input file is non-seekable (such as a pipe) and is large.

--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.

--header=*N[,M]*

Sets the number of header lines and columns displayed on the screen. The value may be of the form "N,M" where N and M are integers, to set the header lines to N and the header columns to M, or it may be a single integer "N" which sets the header lines to N and the header columns to zero, or it may be ",M" which sets the header columns to M and the header lines to zero. When N is nonzero, the first N lines at the top of the screen are replaced with the first N lines of the file, regardless of what part of the file are being viewed. When M is nonzero, the characters displayed at the beginning of each line are replaced with the first M characters of the line, even if the rest of the line is scrolled horizontally. If either N or M is zero, **less** stops displaying header lines or columns, respectively. (Note that it may be necessary to change the setting of the -j option to ensure that the target line is not obscured by the header line(s).)

--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.

--intr=*c*

Use the character *c* instead of ^X to interrupt a read when the "Waiting for data" message is displayed. *c* must be an ASCII character; that is, one with a value between 1 and 127 inclusive. A caret followed by a single character can be used to specify a control character.

--line-num-width=*n*

Sets the minimum width of the line number field when the -N option is in effect to *n* characters. The default is 7.

--modelines=*n*

Before displaying a file, **less** will read the first *n* lines to try to find a vim-compatible *modeline*. If *n* is zero, **less** does not try to find modelines. By using a modeline, the file itself can specify the tab stops that should be used when viewing it.

A modeline contains, anywhere in the line, a program name ("vi", "vim", "ex", or "less"), followed by a colon, possibly followed by the word "set", and finally followed by zero or more option settings. If the word "set" is used, option settings are separated by spaces, and end at the first colon. If the word "set" is not used, option settings may be separated by either spaces or colons. The word "set" is required if the program name is "less" but optional if any of the other three names are used. If any option setting is of the form "tabstop=*n*" or "ts=*n*", then tab stops are automatically set as if --tabs=*n* had been given. See the --tabs description for acceptable values of *n*.

--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.

--no-number-headers

Header lines (defined via the --header option) are not assigned line numbers. Line number 1 is assigned to the first line after any header lines.

--no-search-headers

Searches do not include header lines or header columns.

--no-vbell
Disables the terminal's visual bell.

--proc-backspace
If set, backspaces are handled as if neither the -u option nor the -U option were set. That is, a backspace adjacent to an underscore causes text to be displayed in underline mode, and a backspace between identical characters cause text to be displayed in boldface mode. This option overrides the -u and -U options, so that display of backspaces can be controlled separate from tabs and carriage returns. If not set, backspace display is controlled by the -u and -U options.

--PROC-BACKSPACE
If set, backspaces are handled as if the -U option were set; that is backspaces are treated as control characters.

--proc-return
If set, carriage returns are handled as if neither the -u option nor the -U option were set. That is, a carriage return immediately before a newline is deleted. This option overrides the -u and -U options, so that display of carriage returns can be controlled separate from that of backspaces and tabs. If not set, carriage return display is controlled by the -u and -U options.

--PROC-RETURN
If set, carriage returns are handled as if the -U option were set; that is carriage returns are treated as control characters.

--proc-tab
If set, tabs are handled as if the -U option were not set. That is, tabs are expanded to spaces. This option overrides the -U option, so that display of tabs can be controlled separate from that of backspaces and carriage returns. If not set, tab display is controlled by the -U options.

--PROC-TAB
If set, tabs are handled as if the -U option were set; that is tabs are treated as control characters.

--redraw-on-quit
When quitting, after sending the terminal deinitialization string, redraws the entire last screen. On terminals whose terminal deinitialization string causes the terminal to switch from an alternate screen, this makes the last screenful of the current file remain visible after **less** has quit.

--rscroll=*c*
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**.

--search-options=...
Sets default search modifiers. The value is a string of one or more of the characters E, F, K, N, R or W. Setting any of these has the same effect as typing that control character at the beginning of every search pattern. For example, setting --search-options=W is the same as typing ^W at the beginning of every pattern. The value may also contain a digit between 1 and 5, which has the same effect as typing ^S followed by that digit at the beginning of every search pattern. The value "-" disables all default search modifiers.

--show-preproc-errors
If a preprocessor produces data, then exits with a non-zero exit code, **less** will display a warning.

--status-col-width=*n*
Sets the width of the status column when the -J option is in effect. The default is 2 characters.

--status-line
If a line is marked, the entire line (rather than just the status column) is highlighted. Also lines highlighted due to the -w option will have the entire line highlighted. If --use-color is set, the line is colored rather than highlighted.

--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 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.

--wordwrap

When the -S option is not in use, wrap each line at a space or tab if possible, so that a word is not split between two lines. The default is to wrap at any character.

-- 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 creating a lesskey source file. This file specifies a set of command keys and an action associated with each key. You may also change the line-editing keys (see LINE EDITING), and set environment variables used by **less**. See the **lesskey**(1) manual page for details about the file format.

If the environment variable LESSKEYIN is set, **less** uses that as the name of the lesskey source file. Otherwise, **less** looks in a standard place for the lesskey source file: On Unix systems, **less** looks for a lesskey file called "$XDG_CONFIG_HOME/lesskey" or "$HOME/.config/lesskey" or "$HOME/.lesskey". On MS-DOS and Windows systems, **less** looks for a lesskey file called "$HOME/_lesskey", and if it is not found there, then looks for a lesskey file called "_lesskey" in any directory specified in the PATH environment variable. On OS/2 systems, **less** looks for a lesskey file called "$HOME/lesskey.ini", and if it is not found, then looks for a lesskey file called "lesskey.ini" in any directory specified in the INIT environment variable, and if it not found there, then looks for a lesskey file called "lesskey.ini" in any directory specified in the PATH environment variable.

A system-wide lesskey source 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 LESSKEYIN_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/syslesskey. (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:_syslesskey. On OS/2 systems, the system-wide lesskey file is c:\syslesskey.ini.

Previous versions of **less** (before v582) used lesskey files with a binary format, produced by the **lesskey** program. It is no longer necessary to use the **lesskey** program.

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 determines the behavior when the output is empty. If the output is empty and the exit status is zero, the empty output is considered to be replacement text. If the output is empty and the exit status is nonzero, 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.

When the character set is utf-8, in rare cases it may be desirable to override the Unicode definition of the type of certain characters. For example, characters in a Private Use Area are normally treated as control characters, but if you are using a custom font with printable characters in that range, it may be desirable to tell **less** to treat such characters as printable. This can be done by setting the LESSUTFCHARDEF environment variable to a comma-separated list of *character type* definitions. Each character type definition consists of either one hexadecimal codepoint or a pair of codepoints separated by a dash, followed by a colon and a type character. Each hexadecimal codepoint may optionally be preceded by a "U" or "U+". If a pair of codepoints is given, the type is set for all characters inclusively between the two values. If there are multiple comma-separated codepoint values, they must be in ascending numerical order. The type character may be one of:

p A normal printable character.

w A wide (2-space) printable character.

b A binary (non-printable) character.

c A composing (zero width) character.

For example, setting LESSUTFCHARDEF to

E000-F8FF:p,F0000-FFFFD:p,100000-10FFFD:p

would make all Private Use Area characters be treated as printable.

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. (References to the input file size below refer to the preprocessed size, if an input preprocessor is being used.)

%b*X*

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.

%d*X*

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.

%l*X*
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.

%p*X*
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.

%P*X*
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.

?b*X*
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).

?d*X*
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).

?l*X* 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.

?p*X*
True if the percent into the current input file, based on byte offsets, of the specified line is known.

?P*X*
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 pshell 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)

history file

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**(1) 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 depends on the operating system, but is usually:

Linux and Unix
"$XDG_STATE_HOME/lesshst" or "$HOME/.local/state/lesshst" or "$XDG_DATA_HOME/lesshst" or "$HOME/.lesshst".

Windows and MS-DOS
"$HOME/_lesshst".

OS/2
"$HOME/lesshst.ini" or "$INIT/lesshst.ini".

LESSHISTSIZE
The maximum number of commands to save in the history file. The default is 100.

LESSKEYIN
Name of the default *lesskey source* file.

LESSKEY
Name of the default *lesskey binary* file. (Not used if "$LESSKEYIN" exists.)

LESSKEYIN_SYSTEM
Name of the default system-wide *lesskey source* file.

LESSKEY_SYSTEM
Name of the default system-wide *lesskey binary* file. (Not used if "$LESSKEYIN_SYSTEM" exists.)

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.

LESSUTFCHARDEF
Overrides the type of specified Unicode characters.

LESS_COLUMNS
Sets the number of columns on the screen. Unlike COLUMNS, takes precedence over the system’s idea of the screen size, so it can be used to make **less** use less than the full screen width. If set to a negative number, sets the number of columns used to this much less than the actual screen width.

LESS_LINES
Sets the number of lines on the screen. Unlike LINES, takes precedence over the system’s idea of the screen size, so it can be used to make **less** use less than the full screen height. If set to a negative number, sets the number of lines used to this much less than the actual screen height. When set, **less** repaints the entire screen on every movement command, so scrolling may be slower.

LESS_DATA_DELAY
Duration (in milliseconds) after starting to read data from the input, after which the "Waiting for data" message will be displayed. The default is 4000 (4 seconds).

LESS_IS_MORE
Emulate the **more**(1) command.

LESS_TERMCAP_xx
Where "xx" is any two characters, overrides the definition of the termcap "xx" capability for the terminal.

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).

XDG_CONFIG_HOME
Possible location of the **lesskey** file; see the KEY BINDINGS section.

XDG_DATA_HOME
Possible location of the history file; see the description of the LESSHISTFILE environment variable.

XDG_STATE_HOME
Possible location of the history file; see the description of the LESSHISTFILE environment variable.

SEE ALSO

lesskey(1), **lessecho**(1)

COPYRIGHT

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.

AUTHOR

Mark Nudelman
Report bugs at https://github.com/gwsw/less/issues.
For more information, see the less homepage at
https://greenwoodsoftware.com/less