NAME

less − opposite of more

SYNOPSIS

less −?
less −−help
less −V
less −−version
less [−[+]aBcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]
[−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.

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.

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.

r or ^R or ^L
Repaint the screen.

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

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.)

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

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

p or % Go to a position N percent into the file. N should be between 0 and 100.

{ 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 letter, marks the current position with that letter.

’ (Single quote.) Followed by any lowercase letter, returns to the position which was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. 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.

/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 second 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.

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

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.

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.)

: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).

ˆXˆV or E
Same as :e. Warning: some systems use ˆV as a special literalization character. On such systems, you may not be able to use ˆV.

: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 ˆG 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 ˆP (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 RETURN after typing the option name. A ˆP 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 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 first line on the current screen and the position marked by the letter. <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 "−+".

For options like −P or −D which take a following string, a dollar sign ($) must be used to signal the end of the string. For example, to set two −D options on MS-DOS, you must have a dollar sign between them, like this:

LESS="-Dn9.1$-Ds4.1"

−? 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
Causes searches to start after the last line displayed on the screen, thus skipping all lines displayed on the screen. By default, searches start at the second line on the screen (or after the last found line; see the −j option).

−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 64K 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 is 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 64K (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 file 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
The −C option is like −c, but the screen is cleared before it is repainted.

−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*
[MS-DOS only] Sets the color of the text displayed. **x** is a single character which selects the type of text whose color is being set: n=normal, s=standout, d=bold, u=underlined, k=blink. *color* is a pair of numbers separated by a period. The first number selects the foreground color and the second selects the background color of the text. A single number *N* is the same as *N.0*.

−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.

−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. A target line is the object of a text search, tag search, jump to a line number, jump to a file percentage, or jump to a marked position. The screen line is 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. If the −j option is used, searches begin at the line immediately after the target line. For example, if "−j4" is used, the target line is the fourth line on the screen, so searches begin at the fifth line on the screen.

−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. The status column is also used if the −w or −W option is in effect.

−k*filename* 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 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.

−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.

−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.

−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.

−R or −−RAW-CONTROL-CHARS
Like −r, but only ANSI "color" escape sequences are output in "raw" form. Unlike −r, the screen appearance is maintained correctly in most cases. ANSI "color" escape sequences are sequences of the form:

ESC [... m

where the "..." is zero or more color specification characters For the purpose of keeping track of screen appearance, ANSI color escape sequences are assumed to not move the cursor. 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 rather than folded. That is, the portion of a long line that does not fit in the screen width is not shown. The default is to fold long lines; that is, display the remainder on the next line.

−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 and carriage returns 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. 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.

−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.

−−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.

−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*
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 *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.

−− 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 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. (The bracketed forms do not work in the MS-DOS version.) 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.

DOWNARROW [ESC-j]
Retrieve the next command line.

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.

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)    uncompress -
        if [ -s /tmp/less.$$ ]; then
                echo /tmp/less.$$
        else
                rm -f /tmp/less.$$
        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.

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
        ;;
esac
```

To use this script, put it where it can be executed and set LESSOPEN="|lesspipe.sh %s". 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 "-".

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.

windows
: Selects a character set appropriate for Microsoft Windows (cp 1251).

In special 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_TYPE 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<%X>". 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, illegal 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:

%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.

%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.

%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(file %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(file %i of %m) ..?e(END) ?x- Next\: %x.:
?pB%pB\%:byte %bB?s/%s...%t

?f%f .?n?m(file %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(file %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. %f

Note that this expands to the editor name, followed by a + and the line number, followed by the 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.

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 "−", 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.

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.)

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)

WARNINGS

The = command and prompts (unless changed by −P) report the line numbers of the lines at the top and bottom of the screen, but the byte and percent of the line after the one at the bottom of the screen.

If the :e command is used to name more than one file, and one of the named files has been viewed previously, the new files may be entered into the list in an unexpected order.

On certain older terminals (the so-called "magic cookie" terminals), search highlighting will cause an erroneous display. On such terminals, search highlighting is disabled by default to avoid possible problems.

In certain cases, when search highlighting is enabled and a search pattern begins with a ^, more text than the matching string may be highlighted. (This problem does not occur when less is compiled to use the POSIX regular expression package.)

When viewing text containing ANSI color escape sequences using the −R option, searching will not find text containing an embedded escape sequence. Also, search highlighting may change the color of some of the text which follows the highlighted text.

On some systems, *setlocale* claims that ASCII characters 0 thru 31 are control characters rather than binary characters. This causes *less* to treat some binary files as ordinary, non-binary files. To workaround this problem, set the environment variable LESSCHARSET to "ascii" (or whatever character set is appropriate).

This manual is too long.

See http://www.greenwoodsoftware.com/less for the list of known bugs in all versions of less.

COPYRIGHT

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 <markn@greenwoodsoftware.com>
Send bug reports or comments to the above address or to bug-less@gnu.org.
For more information, see the less homepage at http://www.greenwoodsoftware.com/less.

NAME

lessecho − expand metacharacters

SYNOPSIS

lessecho *[-ox] [-cx] [-pn] [-dn] [-mx] [-nn] [-ex] [-a] file ...*

DESCRIPTION

lessecho is a program that simply echos its arguments on standard output. But any argument containing spaces is enclosed in quotes.

OPTIONS

A summary of options is included below.

−ox Specifies "x" to be the open quote character.

−cx Specifies "x" to be the close quote character.

−pn Specifies "n" to be the open quote character, as an integer.

−dn Specifies "n" to be the close quote character, as an integer.

−mx Specifies "x" to be a metachar.

−nn Specifies "n" to be a metachar, as an integer.

−ex Specifies "x" to be the escape char for metachars.

−fn Specifies "n" to be the escape char for metachars, as an integer.

−a Specifies that all arguments are to be quoted. The default is that only arguments containing spaces are quoted.

SEE ALSO

less(1)

AUTHOR

This manual page was written by Thomas Schoepf <schoepf@debian.org>, for the Debian GNU/Linux system (but may be used by others).

Send bug reports or comments to bug-less@gnu.org.

NAME

lesskey − specify key bindings for less

SYNOPSIS

lesskey [-o output] [--] [input]
lesskey [--output=output] [--] [input]
lesskey -V
lesskey --version

DESCRIPTION

Lesskey is used to specify a set of key bindings to be used by *less.* The input file is a text file which describes the key bindings, If the input file is "-", standard input is read. If no input file is specified, a standard filename is used as the name of the input file, which depends on the system being used: On Unix systems, $HOME/.lesskey is used; on MS-DOS systems, $HOME/_lesskey is used; and on OS/2 systems $HOME/lesskey.ini is used, or $INIT/lesskey.ini if $HOME is undefined. The output file is a binary file which is used by *less.* If no output file is specified, and the environment variable LESSKEY is set, the value of LESSKEY is used as the name of the output file. Otherwise, a standard filename is used as the name of the output file, which depends on the system being used: On Unix and OS-9 systems, $HOME/.less is used; on MS-DOS systems, $HOME/_less is used; and on OS/2 systems, $HOME/less.ini is used, or $INIT/less.ini if $HOME is undefined. If the output file already exists, *lesskey* will overwrite it.

The -V or --version option causes *lesskey* to print its version number and immediately exit. If -V or --version is present, other options and arguments are ignored.

The input file consists of one or more *sections.* Each section starts with a line that identifies the type of section. Possible sections are:

#command
: Defines new command keys.

#line-edit
: Defines new line-editing keys.

#env Defines environment variables.

Blank lines and lines which start with a pound sign (#) are ignored, except for the special section header lines.

COMMAND SECTION

The command section begins with the line

#command

If the command section is the first section in the file, this line may be omitted. The command section consists of lines of the form:

string <whitespace> *action* [extra-string] <newline>

Whitespace is any sequence of one or more spaces and/or tabs. The *string* is the command key(s) which invoke the action. The *string* may be a single command key, or a sequence of up to 15 keys. The *action* is the name of the less action, from the list below. The characters in the *string* may appear literally, or be prefixed by a caret to indicate a control key. A backslash followed by one to three octal digits may be used to specify a character by its octal value. A backslash followed by certain characters specifies input characters as follows:

\b BACKSPACE

\e ESCAPE

\n NEWLINE

\r RETURN

\t	TAB
\ku	UP ARROW
\kd	DOWN ARROW
\kr	RIGHT ARROW
\kl	LEFT ARROW
\kU	PAGE UP
\kD	PAGE DOWN
\kh	HOME
\ke	END
\kx	DELETE

A backslash followed by any other character indicates that character is to be taken literally. Characters which must be preceded by backslash include caret, space, tab and the backslash itself.

An action may be followed by an "extra" string. When such a command is entered while running *less*, the action is performed, and then the extra string is parsed, just as if it were typed in to *less*. This feature can be used in certain cases to extend the functionality of a command. For example, see the "{" and ":t" commands in the example below. The extra string has a special meaning for the "quit" action: when *less* quits, first character of the extra string is used as its exit status.

EXAMPLE

The following input file describes the set of default command keys used by less:

```
#command
\r              forw-line
\n              forw-line
e               forw-line
j               forw-line
\kd     forw-line
^E              forw-line
^N              forw-line
k               back-line
y               back-line
^Y              back-line
^K              back-line
^P              back-line
J               forw-line-force
K               back-line-force
Y               back-line-force
d               forw-scroll
^D              forw-scroll
u               back-scroll
^U              back-scroll
\40     forw-screen
f               forw-screen
^F              forw-screen
^V              forw-screen
\kD     forw-screen
b               back-screen
^B              back-screen
\ev             back-screen
\kU     back-screen
z               forw-window
w               back-window
\e\40           forw-screen-force
F               forw-forever
```

```
R               repaint-flush
r               repaint
^R              repaint
^L              repaint
\eu             undo-hilite
g               goto-line
\kh         goto-line
<               goto-line
\e<             goto-line
p               percent
%               percent
\e[             left-scroll
\e]             right-scroll
\e(             left-scroll
\e)             right-scroll
{               forw-bracket {}
}               back-bracket {}
(               forw-bracket ()
)               back-bracket ()
[               forw-bracket []
]               back-bracket []
\e^F            forw-bracket
\e^B            back-bracket
G               goto-end
\e>             goto-end
>               goto-end
\ke         goto-end
=               status
^G              status
:f              status
/               forw-search
?               back-search
\e/             forw-search *
\e?             back-search *
n               repeat-search
\en             repeat-search-all
N               reverse-search
\eN             reverse-search-all
m               set-mark
'               goto-mark
^X^X            goto-mark
E               examine
:e              examine
^X^V            examine
:n              next-file
:p              prev-file
t               next-tag
T               prev-tag
:x              index-file
:d              remove-file
-               toggle-option
:t              toggle-option t
s               toggle-option o
_               display-option
|               pipe
v               visual
!               shell
+               firstcmd
H               help
```

```
h           help
V           version
0           digit
1           digit
2           digit
3           digit
4           digit
5           digit
6           digit
7           digit
8           digit
9           digit
q           quit
Q           quit
:q          quit
:Q          quit
ZZ          quit
```

PRECEDENCE

Commands specified by *lesskey* take precedence over the default commands. A default command key may be disabled by including it in the input file with the action "invalid". Alternatively, a key may be defined to do nothing by using the action "noaction". "noaction" is similar to "invalid", but *less* will give an error beep for an "invalid" command, but not for a "noaction" command. In addition, ALL default commands may be disabled by adding this control line to the input file:

```
#stop
```

This will cause all default commands to be ignored. The #stop line should be the last line in that section of the file.

Be aware that #stop can be dangerous. Since all default commands are disabled, you must provide sufficient commands before the #stop line to enable all necessary actions. For example, failure to provide a "quit" command can lead to frustration.

LINE EDITING SECTION

The line-editing section begins with the line:

```
#line-edit
```

This section specifies new key bindings for the line editing commands, in a manner similar to the way key bindings for ordinary commands are specified in the #command section. The line-editing section consists of a list of keys and actions, one per line as in the example below.

EXAMPLE

The following input file describes the set of default line-editing keys used by less:

```
#line-edit
\t          forw-complete
\17         back-complete
\e\t        back-complete
^L          expand
^V          literal
^A          literal
\el         right
\kr         right
\eh         left
\kl         left
\eb         word-left
```

\e\kl	word-left
\ew	word-right
\e\kr	word-right
\ei	insert
\ex	delete
\kx	delete
\eX	word-delete
\ekx	word-delete
\e\b	word-backspace
\e0	home
\kh	home
\e$	end
\ke	end
\ek	up
\ku	up
\ej	down

LESS ENVIRONMENT VARIABLES

The environment variable section begins with the line

#env

Following this line is a list of environment variable assignments. Each line consists of an environment variable name, an equals sign (=) and the value to be assigned to the environment variable. White space before and after the equals sign is ignored. Variables assigned in this way are visible only to *less*. If a variable is specified in the system environment and also in a lesskey file, the value in the lesskey file takes precedence. Although the lesskey file can be used to override variables set in the environment, the main purpose of assigning variables in the lesskey file is simply to have all *less* configuration information stored in one file.

EXAMPLE

The following input file sets the -i option whenever *less* is run, and specifies the character set to be "latin1":

```
#env
LESS = -i
LESSCHARSET = latin1
```

SEE ALSO

less(1)

WARNINGS

It is not possible to specify special keys, such as uparrow, in a keyboard-independent manner. The only way to specify such keys is to specify the escape sequence which a particular keyboard sends when such a key is pressed.

On MS-DOS and OS/2 systems, certain keys send a sequence of characters which start with a NUL character (0). This NUL character should be represented as \340 in a lesskey file.

COPYRIGHT

AUTHOR

Mark Nudelman <markn@greenwoodsoftware.com>
Send bug reports or comments to the above address or to bug-less@gnu.org.

NAME

lessfile, lesspipe − "input preprocessor" for less.

SYNOPSIS

lessfile, lesspipe

DESCRIPTION

This manual page documents briefly the *lessfile*, and *lesspipe* commands. This manual page was written for the Debian GNU/Linux distribution because the input preprocessor scripts are provided by Debian GNU/Linux and are not part of the original program.

lessfile and *lesspipe* are programs that can be used to modify the way the contents of a file are displayed in *less*. What this means is that *less* can automatically open up tar files, uncompress gzipped files, and even display something reasonable for graphics files.

lesspipe will toss the contents/info on STDOUT and *less* will read them as they come across. This means that you do not have to wait for the decoding to finish before less shows you the file. This also means that you will get a 'byte N' instead of an N% as your file position. You can seek to the end and back to get the N% but that means you have to wait for the pipe to finish.

lessfile will toss the contents/info on a file which *less* will then read. After you are done, *lessfile* will then delete the file. This means that the process has to finish before you see it, but you get nice percentages (N%) up front.

USAGE

Just put one of the following two commands in your login script (e.g. ~/.bash_profile):

```
eval "$(lessfile)"
```

or

```
eval "$(lesspipe)"
```

FILE TYPE RECOGNITION

File types are recognized by their extensions. This is a list of currently supported extensions (grouped by the programs that handle them):

```
*.arj
*.tar.bz2
*.bz
*.bz2
*.deb, *.udeb
*.doc
*.gif, *.jpeg, *.jpg, *.pcd, *.png, *.tga, *.tiff, *.tif
*.iso, *.raw, *.bin
*.lha, *.lzh
*.pdf
*.rar, *.r[0-9][0-9]
*.rpm
*.tar.gz, *.tgz, *.tar.z, *.tar.dz
*.gz, *.z, *.dz
*.tar
*.jar, *.war, *.xpi, *.zip
*.zoo
```

USER DEFINED FILTERS

It is possible to extend and overwrite the default *lesspipe* and *lessfile* input processor if you have specialized requirements. Create an executable program with the name *.lessfilter* and put it into your home directory. This can be a shell script or a binary program.

It is important that this program returns the correct exit code: return 0 if your filter handles the input, return 1 if the standard *lesspipe/lessfile* filter should handle the input.

Here is an example script:

```
#!/bin/sh

case "$1" in
   *.extension)
      extension-handler "$1"
      ;;
   *)
      # We don't handle this format.
      exit 1
esac

# No further processing by lesspipe necessary
exit 0
```

FILES

~/.lessfilter

Executable file that can do user defined processing. See section USER DEFINED FILTERS for more information.

BUGS

When trying to open compressed 0 byte files, *less* displays the actual binary file contents. This is not a bug. *less* is designed to do that (see manual page less(1), section INPUT PREPROCESSOR). This is the answer of Mark Nudelman <markn@greenwoodsoftware.com>:

> "I recognized when I designed it that a lesspipe filter cannot output an empty file and have less display nothing in that case; it's a side effect of using the "no output" case to mean "the filter has nothing to do". It could have been designed to have some other mechanism to indicate "nothing to do", but "no output" seemed the simplest and most intuitive for lesspipe writers."

Sometimes, less does not display the contents file you want to view but output that is produced by your login scripts (~/.bashrc or ~/.bash_profile). This happens because less uses your current shell to run the lesspipe filter. Bash first looks for the variable $BASH_ENV in the environment expands its value and uses the expanded value as the name of a file to read and execute. If this file produces any output less will display this. A way to solve this problem is to put the following lines on the top of your login script that produces output:

```
if [ -z "$PS1" ]; then
   exit
fi
```

This tests whether the prompt variable $PS1 is set and if it isn't (which is the case for non-interactive shells) it will exit the script.

SEE ALSO

less(1)

AUTHOR

This manual page was written by Thomas Schoepf <schoepf@debian.org>, for the Debian GNU/Linux system (but may be used by others). Most of the text was copied from a description written by Darren Stalder <torin@daft.com>.

NAME

more − display files on a page-by-page basis

SYNOPSIS

more [-ceisu][-n *number***][-p** *command***][-t** *tagstring***][**_file_ **...]**

DESCRIPTION

The *more* utility shall read files and either write them to the terminal on a page-by-page basis or filter them to standard output. If standard output is not a terminal device, all input files shall be copied to standard output in their entirety, without modification, except as specified for the **-s** option. If standard output is a terminal device, the files shall be written a number of lines (one screenful) at a time under the control of user commands. See the EXTENDED DESCRIPTION section.

Certain block-mode terminals do not have all the capabilities necessary to support the complete *more* definition; they are incapable of accepting commands that are not terminated with a <newline>. Implementations that support such terminals shall provide an operating mode to *more* in which all commands can be terminated with a <newline> on those terminals. This mode:

* Shall be documented in the system documentation
* Shall, at invocation, inform the user of the terminal deficiency that requires the <newline> usage and provide instructions on how this warning can be suppressed in future invocations
* Shall not be required for implementations supporting only fully capable terminals
* Shall not affect commands already requiring <newline>s
* Shall not affect users on the capable terminals from using *more* as described in this volume of IEEE Std 1003.1-2001

OPTIONS

The *more* utility shall conform to the Base Definitions volume of IEEE Std 1003.1-2001, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

-c If a screen is to be written that has no lines in common with the current screen, or *more* is writing its first screen, *more* shall not scroll the screen, but instead shall redraw each line of the screen in turn, from the top of the screen to the bottom. In addition, if *more* is writing its first screen, the screen shall be cleared. This option may be silently ignored on devices with insufficient terminal capabilities.

-e By default, *more* shall exit immediately after writing the last line of the last file in the argument list. If the **-e** option is specified:

1. If there is only a single file in the argument list and that file was completely displayed on a single screen, *more* shall exit immediately after writing the last line of that file.
2. Otherwise, *more* shall exit only after reaching end-of-file on the last file in the argument list twice without an intervening operation. See the EXTENDED DESCRIPTION section.

-i Perform pattern matching in searches without regard to case; see the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.2, Regular Expression General Requirements.

-n *number*
Specify the number of lines per screenful. The *number* argument is a positive decimal integer. The **-n** option shall override any values obtained from any other source.

-p *command*
Each time a screen from a new file is displayed or redisplayed (including as a result of *more* commands; for example, **:p**), execute the *more* command(s) in the command arguments in the order specified, as if entered by the user after the first screen has been displayed. No intermediate results shall be displayed (that is, if the command is a movement to a screen different from the normal first screen, only the screen resulting from the command shall be displayed.)

If any of the commands fail for any reason, an informational message to this effect shall be written, and no further commands specified using the **-p** option shall be executed for this file.

-s Behave as if consecutive empty lines were a single empty line.

-t *tagstring*
Write the screenful of the file containing the tag named by the *tagstring* argument. See the *ctags* utility. The tags feature represented by **-t** *tagstring* and the **:t** command is optional. It shall be provided on any system that also provides a conforming implementation of *ctags*; otherwise, the use of **-t** produces undefined results.

The filename resulting from the **-t** option shall be logically added as a prefix to the list of command line files, as if specified by the user. If the tag named by the *tagstring* argument is not found, it shall be an error, and *more* shall take no further action.

If the tag specifies a line number, the first line of the display shall contain the beginning of that line. If the tag specifies a pattern, the first line of the display shall contain the beginning of the matching text from the first line of the file that contains that pattern. If the line does not exist in the file or matching text is not found, an informational message to this effect shall be displayed, and *more* shall display the default screen as if **-t** had not been specified.

If both the **-t** *tagstring* and **-p** *command* options are given, the **-t** *tagstring* shall be processed first; that is, the file and starting line for the display shall be as specified by **-t**, and then the **-p** *more* command shall be executed. If the line (matching text) specified by the **-t** command does not exist (is not found), no **-p** *more* command shall be executed for this file at any time.

-u Treat a <backspace> as a printable control character, displayed as an implementation-defined character sequence (see the EXTENDED DESCRIPTION section), suppressing backspacing and the special handling that produces underlined or standout mode text on some terminal types. Also, do not ignore a <carriage-return> at the end of a line.

OPERANDS

The following operand shall be supported:

file A pathname of an input file. If no *file* operands are specified, the standard input shall be used. If a *file* is **'-'** , the standard input shall be read at that point in the sequence.

STDIN

The standard input shall be used only if no *file* operands are specified, or if a *file* operand is **'-'** .

INPUT FILES

The input files being examined shall be text files. If standard output is a terminal, standard error shall be used to read commands from the user. If standard output is a terminal, standard error is not readable, and command input is needed, *more* may attempt to obtain user commands from the controlling terminal (for example, **/dev/tty**); otherwise, *more* shall terminate with an error indicating that it was unable to read user commands. If standard output is not a terminal, no error shall result if standard error cannot be opened for reading.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *more*:

COLUMNS
Override the system-selected horizontal display line size. See the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 8, Environment Variables for valid values and results when it is unset or null.

EDITOR
Used by the **v** command to select an editor. See the EXTENDED DESCRIPTION section.

LANG Provide a default value for the internationalization variables that are unset or null. (See the Base Definitions volume of IEEE Std 1003.1-2001, Section 8.2, Internationalization Variables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL
If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE
Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions.

LC_CTYPE
Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classes within regular expressions.

LC_MESSAGES
Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH
Determine the location of message catalogs for the processing of *LC_MESSAGES* .

LINES Override the system-selected vertical screen size, used as the number of lines in a screenful. See the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 8, Environment Variables for valid values and results when it is unset or null. The **-n** option shall take precedence over the *LINES* variable for determining the number of lines in a screenful.

MORE Determine a string containing options described in the OPTIONS section preceded with hyphens and <blank>-separated as on the command line. Any command line options shall be processed after those in the *MORE* variable, as if the command line were:

more $MORE *options operands*

The *MORE* variable shall take precedence over the *TERM* and *LINES* variables for determining the number of lines in a screenful.

TERM Determine the name of the terminal type. If this variable is unset or null, an unspecified default terminal type is used.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The standard output shall be used to write the contents of the input files.

STDERR

The standard error shall be used for diagnostic messages and user commands (see the INPUT FILES section), and, if standard output is a terminal device, to write a prompting string. The prompting string shall appear on the screen line below the last line of the file displayed in the current screenful. The prompt shall contain the name of the file currently being examined and shall contain an end-of-file indication and the name of the next file, if any, when prompting at the end-of-file. If an error or informational message is displayed, it is unspecified whether it is contained in the prompt. If it is not contained in the prompt, it shall be displayed and then the user shall be prompted for a continuation character, at which point another message or the user prompt may be displayed. The prompt is otherwise unspecified. It is unspecified whether informational messages are written for other user commands.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

The following section describes the behavior of *more* when the standard output is a terminal device. If the standard output is not a terminal device, no options other than **-s** shall have any effect, and all input files shall be copied to standard output otherwise unmodified, at which time *more* shall exit without further action.

The number of lines available per screen shall be determined by the **-n** option, if present, or by

examining values in the environment (see the ENVIRONMENT VARIABLES section). If neither method yields a number, an unspecified number of lines shall be used.

The maximum number of lines written shall be one less than this number, because the screen line after the last line written shall be used to write a user prompt and user input. If the number of lines in the screen is less than two, the results are undefined. It is unspecified whether user input is permitted to be longer than the remainder of the single line where the prompt has been written.

The number of columns available per line shall be determined by examining values in the environment (see the ENVIRONMENT VARIABLES section), with a default value as described in the Base Definitions volume of IEEE Std 1003.1-2001, Chapter 8, Environment Variables.

Lines that are longer than the display shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. Folding may occur between glyphs of single characters that take up multiple display columns.

When standard output is a terminal and **-u** is not specified, *more* shall treat <backspace>s and <carriage-return>s specially:

* A character, followed first by a sequence of *n* <backspace>s (where *n* is the same as the number of column positions that the character occupies), then by *n* underscore characters (**'_'**), shall cause that character to be written as underlined text, if the terminal type supports that. The *n* underscore characters, followed first by *n* <backspace>s, then any character with *n* column positions, shall also cause that character to be written as underlined text, if the terminal type supports that.

* A sequence of *n* <backspace>s (where *n* is the same as the number of column positions that the previous character occupies) that appears between two identical printable characters shall cause the first of those two characters to be written as emboldened text (that is, visually brighter, standout mode, or inverse-video mode), if the terminal type supports that, and the second to be discarded. Immediately subsequent occurrences of <backspace>/ character pairs for that same character shall also be discarded. (For example, the sequence **"a\ba\ba\ba"** is interpreted as a single emboldened **'a'** .)

* The *more* utility shall logically discard all other <backspace>s from the line as well as the character which precedes them, if any.

* A <carriage-return> at the end of a line shall be ignored, rather than being written as a non-printable character, as described in the next paragraph.

It is implementation-defined how other non-printable characters are written. Implementations should use the same format that they use for the *ex* **print** command; see the OPTIONS section within the *ed* utility. It is unspecified whether a multi-column character shall be separated if it crosses a display line boundary; it shall not be discarded. The behavior is unspecified if the number of columns on the display is less than the number of columns any single character in the line being displayed would occupy.

When each new file is displayed (or redisplayed), *more* shall write the first screen of the file. Once the initial screen has been written, *more* shall prompt for a user command. If the execution of the user command results in a screen that has lines in common with the current screen, and the device has sufficient terminal capabilities, *more* shall scroll the screen; otherwise, it is unspecified whether the screen is scrolled or redrawn.

For all files but the last (including standard input if no file was specified, and for the last file as well, if the **-e** option was not specified), when *more* has written the last line in the file, *more* shall prompt for a user command. This prompt shall contain the name of the next file as well as an indication that *more* has reached end-of-file. If the user command is **f**, <control>-F, <space>, **j**, <newline>, **d**, <control>-D, or **s**, *more* shall display the next file. Otherwise, if displaying the last file, *more* shall exit. Otherwise, *more* shall execute the user command specified.

Several of the commands described in this section display a previous screen from the input stream. In the case that text is being taken from a non-rewindable stream, such as a pipe, it is implementation-defined how much backwards motion is supported. If a command cannot be executed because of a limitation on backwards motion, an error message to this effect shall be displayed, the current screen shall not change, and the user shall be prompted for another command.

If a command cannot be performed because there are insufficient lines to display, *more* shall alert the

terminal. If a command cannot be performed because there are insufficient lines to display or a / command fails: if the input is the standard input, the last screen in the file may be displayed; otherwise, the current file and screen shall not change, and the user shall be prompted for another command.

The interactive commands in the following sections shall be supported. Some commands can be preceded by a decimal integer, called *count* in the following descriptions. If not specified with the command, *count* shall default to 1. In the following descriptions, *pattern* is a basic regular expression, as described in the Base Definitions volume of IEEE Std 1003.1-2001, Section 9.3, Basic Regular Expressions. The term "examine" is historical usage meaning "open the file for viewing''; for example, *more* **foo** would be expressed as examining file **foo**.

In the following descriptions, unless otherwise specified, *line* is a line in the *more* display, not a line from the file being examined.

In the following descriptions, the *current position* refers to two things:

1. The position of the current line on the screen
2. The line number (in the file) of the current line on the screen

Usually, the line on the screen corresponding to the current position is the third line on the screen. If this is not possible (there are fewer than three lines to display or this is the first page of the file, or it is the last page of the file), then the current position is either the first or last line on the screen as described later.

Help

Synopsis:

h

Write a summary of these commands and other implementation-defined commands. The behavior shall be as if the *more* utility were executed with the **-e** option on a file that contained the summary information. The user shall be prompted as described earlier in this section when end-of-file is reached. If the user command is one of those specified to continue to the next file, *more* shall return to the file and screen state from which the **h** command was executed.

Scroll Forward One Screenful

Synopsis:

[*count***]f**
[*count***]<control>-F**

Scroll forward *count* lines, with a default of one screenful. If *count* is more than the screen size, only the final screenful shall be written.

Scroll Backward One Screenful

Synopsis:

[*count***]b**
[*count***]<control>-B**

Scroll backward *count* lines, with a default of one screenful (see the **-n** option). If *count* is more than the screen size, only the final screenful shall be written.

Scroll Forward One Line

Synopsis:

[*count***]<space>**
[*count***]j**

[*count***]<newline>**

Scroll forward *count* lines. The default *count* for the <space> shall be one screenful; for **j** and <newline>, one line. The entire *count* lines shall be written, even if *count* is more than the screen size.

Scroll Backward One Line

Synopsis:

[*count***]k**

Scroll backward *count* lines. The entire *count* lines shall be written, even if *count* is more than the screen size.

Scroll Forward One Half Screenful

Synopsis:

[*count***]d**
[*count***]<control>-D**

Scroll forward *count* lines, with a default of one half of the screen size. If *count* is specified, it shall become the new default for subsequent **d**, <control>-D, and **u** commands.

Skip Forward One Line

Synopsis:

[*count***]s**

Display the screenful beginning with the line *count* lines after the last line on the current screen. If *count* would cause the current position to be such that less than one screenful would be written, the last screenful in the file shall be written.

Scroll Backward One Half Screenful

Synopsis:

[*count***]u**
[*count***]<control>-U**

Scroll backward *count* lines, with a default of one half of the screen size. If *count* is specified, it shall become the new default for subsequent **d**, <control>-D, **u**, and <control>-U commands. The entire *count* lines shall be written, even if *count* is more than the screen size.

Go to Beginning of File

Synopsis:

[*count***]g**

Display the screenful beginning with line *count*.

Go to End-of-File

Synopsis:

[*count***]G**

If *count* is specified, display the screenful beginning with the line *count*. Otherwise, display the last screenful of the file.

Refresh the Screen

Synopsis:

r
<control>-L

Refresh the screen.

Discard and Refresh

Synopsis:

R

Refresh the screen, discarding any buffered input. If the current file is non-seekable, buffered input shall not be discarded and the **R** command shall be equivalent to the **r** command.

Mark Position

Synopsis:

m*letter*

Mark the current position with the letter named by *letter*, where *letter* represents the name of one of the lowercase letters of the portable character set. When a new file is examined, all marks may be lost.

Return to Mark

Synopsis:

'*letter*

Return to the position that was previously marked with the letter named by *letter*, making that line the current position.

Return to Previous Position

Synopsis:

''

Return to the position from which the last large movement command was executed (where a "large movement" is defined as any movement of more than a screenful of lines). If no such movements have been made, return to the beginning of the file.

Search Forward for Pattern

Synopsis:

[*count***]/[!]***pattern***<newline>**

Display the screenful beginning with the *count*th line containing the pattern. The search shall start after the first line currently displayed. The null regular expression (**'/'** followed by a <newline>) shall repeat the search using the previous regular expression, with a default *count*. If the character **'!'** is included, the matching lines shall be those that do not contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be displayed.

Search Backward for Pattern

Synopsis:

[*count***]?[!]***pattern***<newline>**

Display the screenful beginning with the *count*th previous line containing the pattern. The search shall start on the last line before the first line currently displayed. The null regular expression (**'?'** followed by a <newline>) shall repeat the search using the previous regular expression, with a default *count*. If the character **'!'** is included, matching lines shall be those that do not contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be displayed.

Repeat Search

Synopsis:

[*count***]n**

Repeat the previous search for *count*th line containing the last *pattern* (or not containing the last *pattern*, if the previous search was **"/!"** or **"?!"**).

Repeat Search in Reverse

Synopsis:

[*count***]N**

Repeat the search in the opposite direction of the previous search for the *count*th line containing the last *pattern* (or not containing the last *pattern*, if the previous search was **"/!"** or **"?!"**).

Examine New File

Synopsis:

:e [*filename***]<newline>**

Examine a new file. If the *filename* argument is not specified, the current file (see the **:n** and **:p** commands below) shall be re-examined. The *filename* shall be subjected to the process of shell word expansions (see *Word Expansions*); if more than a single pathname results, the effects are unspecified. If *filename* is a number sign (**'#'**), the previously examined file shall be re-examined. If *filename* is not accessible for any reason (including that it is a non-seekable file), an error message to this effect shall be displayed and the current file and screen shall not change.

Examine Next File

Synopsis:

[*count***]:n**

Examine the next file. If a number *count* is specified, the *count*th next file shall be examined. If *filename* refers to a non-seekable file, the results are unspecified.

Examine Previous File

Synopsis:

[*count***]:p**

Examine the previous file. If a number *count* is specified, the *count*th previous file shall be examined. If

filename refers to a non-seekable file, the results are unspecified.

Go to Tag

Synopsis:

:t *tagstring*<**newline**>

If the file containing the tag named by the *tagstring* argument is not the current file, examine the file, as if the **:e** command was executed with that file as the argument. Otherwise, or in addition, display the screenful beginning with the tag, as described for the **-t** option (see the OPTIONS section). If the *ctags* utility is not supported by the system, the use of **:t** produces undefined results.

Invoke Editor

Synopsis:

v

Invoke an editor to edit the current file being examined. If standard input is being examined, the results are unspecified. The name of the editor shall be taken from the environment variable *EDITOR* , or shall default to *vi*. If the last pathname component in *EDITOR* is either *vi* or *ex*, the editor shall be invoked with a **-c** *linenumber* command line argument, where *linenumber* is the line number of the file line containing the display line currently displayed as the first line of the screen. It is implementation-defined whether line-setting options are passed to editors other than *vi* and *ex*.

When the editor exits, *more* shall resume with the same file and screen as when the editor was invoked.

Display Position

Synopsis:

=
<control>-G

Write a message for which the information references the first byte of the line after the last line of the file on the screen. This message shall include the name of the file currently being examined, its number relative to the total number of files there are to examine, the line number in the file, the byte number and the total bytes in the file, and what percentage of the file precedes the current position. If *more* is reading from standard input, or the file is shorter than a single screen, the line number, the byte number, the total bytes, and the percentage need not be written.

Quit

Synopsis:

q
:q
ZZ

Exit *more*.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

If an error is encountered accessing a file when using the **:n** command, *more* shall attempt to examine the next file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the **:p** command, *more* shall attempt to examine the previous file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the **:e** command, *more* shall remain in the current file and the final exit status shall not be affected.

The following sections are informative.

APPLICATION USAGE

When the standard output is not a terminal, only the **-s** filter-modification option is effective. This is based on historical practice. For example, a typical implementation of *man* pipes its output through *more* **-s** to squeeze excess white space for terminal users. When *man* is piped to *lp*, however, it is undesirable for this squeezing to happen.

EXAMPLES

The **-p** allows arbitrary commands to be executed at the start of each file. Examples are:

more **-p G** *file1 file2*

Examine each file starting with its last screenful.

more **-p** 100 *file1 file2*

Examine each file starting with line 100 in the current position (usually the third line, so line 98 would be the first line written).

more **-p** /100 *file1 file2*

Examine each file starting with the first line containing the string **"100"** in the current position

RATIONALE

The *more* utility, available in BSD and BSD-derived systems, was chosen as the prototype for the POSIX file display program since it is more widely available than either the public-domain program *less* or than *pg*, a pager provided in System V. The 4.4 BSD *more* is the model for the features selected; it is almost fully upwards-compatible from the 4.3 BSD version in wide use and has become more amenable for *vi* users. Several features originally derived from various file editors, found in both *less* and *pg*, have been added to this volume of IEEE Std 1003.1-2001 as they have proved extremely popular with users.

There are inconsistencies between *more* and *vi* that result from historical practice. For example, the single-character commands **h**, **f**, **b**, and <space> are screen movers in *more*, but cursor movers in *vi*. These inconsistencies were maintained because the cursor movements are not applicable to *more* and the powerful functionality achieved without the use of the control key justifies the differences.

The tags interface has been included in a program that is not a text editor because it promotes another degree of consistent operation with *vi*. It is conceivable that the paging environment of *more* would be superior for browsing source code files in some circumstances.

The operating mode referred to for block-mode terminals effectively adds a <newline> to each Synopsis line that currently has none. So, for example, **d** <newline> would page one screenful. The mode could be triggered by a command line option, environment variable, or some other method. The details are not imposed by this volume of IEEE Std 1003.1-2001 because there are so few systems known to support such terminals. Nevertheless, it was considered that all systems should be able to support *more* given the exception cited for this small community of terminals because, in comparison to *vi*, the cursor movements are few and the command set relatively amenable to the optional <newline>s.

Some versions of *more* provide a shell escaping mechanism similar to the *ex* **!** command. The standard developers did not consider that this was necessary in a paginator, particularly given the wide acceptance of multiple window terminals and job control features. (They chose to retain such features in the editors and *mailx* because the shell interaction also gives an opportunity to modify the editing buffer, which is not applicable to *more*.)

The **-p** (position) option replaces the **+** command because of the Utility Syntax Guidelines. In early proposals, it took a *pattern* argument, but historical *less* provided the *more* general facility of a

command. It would have been desirable to use the same **-c** as *ex* and *vi*, but the letter was already in use.

The text stating "from a non-rewindable stream ... implementations may limit the amount of backwards motion supported" would allow an implementation that permitted no backwards motion beyond text already on the screen. It was not possible to require a minimum amount of backwards motion that would be effective for all conceivable device types. The implementation should allow the user to back up as far as possible, within device and reasonable memory allocation constraints.

Historically, non-printable characters were displayed using the ARPA standard mappings, which are as follows:

1. Printable characters are left alone.
2. Control characters less than \177 are represented as followed by the character offset from the **'@'** character in the ASCII map; for example, \007 is represented as **'G'** .
3. \177 is represented as followed by **'?'** .

The display of characters having their eighth bit set was less standard. Existing implementations use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed characters with their eighth bit set as the two characters **"M-"** , followed by the seven-bit display as described previously.) The latter probably has the best claim to historical practice because it was used with the **-v** option of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

No specific display format is required by IEEE Std 1003.1-2001. Implementations are encouraged to conform to historic practice in the absence of any strong reason to diverge.

FUTURE DIRECTIONS

None.

SEE ALSO

Shell Command Language , *ctags* , *ed* , *ex* , *vi*

COPYRIGHT