NAME

editline - line editing user interface

DESCRIPTION

When a program using the editline(3) library prompts for an input string using the function el_wgets(3), it reads characters from the terminal. Invalid input bytes that do not form characters are silently discarded. For each character read, one editor command is executed. The mapping of input characters to editor commands depends on the editing mode. There are three editing modes: vi insert mode, vi command mode, and emacs mode. The default is vi insert mode. The program can switch the default to emacs mode by using the el_set(3) or el_parse(3) functions, and the user can switch to emacs mode either in the editrc(5) configuration file or interactively with the **ed-command** editor command, in all three cases executing the **bind -e** builtin command.

If trying to read from the terminal results in end of file or an error, the library signals end of file to the program and does not return a string.

Input character bindings

All default bindings described below can be overridden by individual programs and can be changed with the editrc(5) **bind** builtin command.

In the following tables, ‘Ctrl-’ indicates a character with the bit 0x40 flipped, and ‘Meta-’ indicates a character with the bit 0x80 set. In vi insert mode and in emacs mode, all Meta-characters considered printable by the current locale(1) are bound to **ed-insert** instead of to the editor command listed below. Consequently, in UTF-8 mode, most of the Meta-characters are not directly accessible because their code points are occupied by printable Unicode characters, and Meta-characters are usually input using the **em-meta-next** editor command. For example, to enter ‘Meta-B’ in order to call the **ed-prev-word** editor command in emacs mode, call **em-meta-next** by pressing and releasing the escape key (or equivalently, Ctrl-[), then press and release the ‘B’ key. If you have configured a Meta-key on your keyboard, for example with ‘setxkbmap -option altwin:left_meta_win’, the Ctrl-Meta-characters are directly accessible. For example, to enter ‘Ctrl-Meta-H’ in order to call the **ed-delete-prev-word** editor command in emacs mode, hold down the keys ‘Ctrl’, ‘Meta’, and ‘H’ at the same time. Alternatively, press and release the escape key, then press and release ‘Ctrl-H’.

In vi input mode, input characters are bound to the following editor commands by default:

Ctrl-D, EOF	**vi-list-or-eof**
Ctrl-H, BS	**vi-delete-prev-char**
Ctrl-J, LF	**ed-newline**
Ctrl-M, CR	**ed-newline**
Ctrl-Q	**ed-tty-start-output**

Ctrl-S	**ed-tty-stop-output**
Ctrl-U	**vi-kill-line-prev**
Ctrl-V	**ed-quoted-insert**
Ctrl-W	**ed-delete-prev-word**
Ctrl-[, ESC	**vi-command-mode**
Ctrl-\, QUIT	**ed-tty-sigquit**
Ctrl-?, DEL	**vi-delete-prev-char**

All other input characters except the NUL character (Ctrl-@) are bound to **ed-insert**.

In vi command mode, input characters are bound to the following editor commands by default:

Ctrl-A	**ed-move-to-beg**
Ctrl-C, INT	**ed-tty-sigint**
Ctrl-E	**ed-move-to-end**
Ctrl-H, BS	**ed-delete-prev-char**
Ctrl-J, LF	**ed-newline**
Ctrl-K	**ed-kill-line**
Ctrl-L, FF	**ed-clear-screen**
Ctrl-M, CR	**ed-newline**
Ctrl-N	**ed-next-history**
Ctrl-O	**ed-tty-flush-output**
Ctrl-P	**ed-prev-history**
Ctrl-Q	**ed-tty-start-output**
Ctrl-R	**ed-redisplay**
Ctrl-S	**ed-tty-stop-output**
Ctrl-U	**vi-kill-line-prev**
Ctrl-W	**ed-delete-prev-word**
Ctrl-[, ESC	**em-meta-next**
Ctrl-\, QUIT	**ed-tty-sigquit**
Space	**ed-next-char**
#	**vi-comment-out**
$	**ed-move-to-end**
%	**vi-match**
+	**ed-next-history**
,	**vi-repeat-prev-char**
-	**ed-prev-history**
.	**vi-redo**
/	**vi-search-prev**
0	**vi-zero**

1 to 9	**ed-argument-digit**
:	**ed-command**
;	**vi-repeat-next-char**
?	**vi-search-next**
@	**vi-alias**
A	**vi-add-at-eol**
B	**vi-prev-big-word**
C	**vi-change-to-eol**
D	**ed-kill-line**
E	**vi-end-big-word**
F	**vi-prev-char**
G	**vi-to-history-line**
I	**vi-insert-at-bol**
J	**ed-search-next-history**
K	**ed-search-prev-history**
N	**vi-repeat-search-prev**
O	**ed-sequence-lead-in**
P	**vi-paste-prev**
R	**vi-replace-mode**
S	**vi-substitute-line**
T	**vi-to-prev-char**
U	**vi-undo-line**
W	**vi-next-big-word**
X	**ed-delete-prev-char**
Y	**vi-yank-end**
[	**ed-sequence-lead-in**
^	**ed-move-to-beg**
_	**vi-history-word**
a	**vi-add**
b	**vi-prev-word**
c	**vi-change-meta**
d	**vi-delete-meta**
e	**vi-end-word**
f	**vi-next-char**
h	**ed-prev-char**
i	**vi-insert**
j	**ed-next-history**
k	**ed-prev-history**
l	**ed-next-char**
n	**vi-repeat-search-next**

p	**vi-paste-next**
r	**vi-replace-char**
s	**vi-substitute-char**
t	**vi-to-next-char**
u	**vi-undo**
v	**vi-histedit**
w	**vi-next-word**
x	**ed-delete-next-char**
y	**vi-yank**
|	**vi-to-column**
~	**vi-change-case**
Ctrl-?, DEL	**ed-delete-prev-char**
Meta-O	**ed-sequence-lead-in**
Meta-[	**ed-sequence-lead-in**

In emacs mode, input characters are bound to the following editor commands by default:

0 to 9	**ed-digit**
Ctrl-@, NUL	**em-set-mark**
Ctrl-A	**ed-move-to-beg**
Ctrl-B	**ed-prev-char**
Ctrl-C, INT	**ed-tty-sigint**
Ctrl-D, EOF	**em-delete-or-list**
Ctrl-E	**ed-move-to-end**
Ctrl-F	**ed-next-char**
Ctrl-H, BS	**em-delete-prev-char**
Ctrl-J, LF	**ed-newline**
Ctrl-K	**ed-kill-line**
Ctrl-L, FF	**ed-clear-screen**
Ctrl-M, CR	**ed-newline**
Ctrl-N	**ed-next-history**
Ctrl-O	**ed-tty-flush-output**
Ctrl-P	**ed-prev-history**
Ctrl-Q	**ed-tty-start-output**
Ctrl-R	**ed-redisplay**
Ctrl-S	**ed-tty-stop-output**
Ctrl-T	**ed-transpose-chars**
Ctrl-U	**ed-kill-line**
Ctrl-V	**ed-quoted-insert**
Ctrl-W	**em-kill-region**

Ctrl-X	**ed-sequence-lead-in**
Ctrl-Y	**em-yank**
Ctrl-Z, TSTP	**ed-tty-sigtstp**
Ctrl-[, ESC	**em-meta-next**
Ctrl-\, QUIT	**ed-tty-sigquit**
Ctrl-]	**ed-tty-dsusp**
Ctrl-?, DEL	**em-delete-prev-char**
Ctrl-Meta-H	**ed-delete-prev-word**
Ctrl-Meta-L	**ed-clear-screen**
Ctrl-Meta-_	**em-copy-prev-word**
Meta-0 to 9	**ed-argument-digit**
Meta-B	**ed-prev-word**
Meta-C	**em-capitol-case**
Meta-D	**em-delete-next-word**
Meta-F	**em-next-word**
Meta-L	**em-lower-case**
Meta-N	**ed-search-next-history**
Meta-O	**ed-sequence-lead-in**
Meta-P	**ed-search-prev-history**
Meta-U	**em-upper-case**
Meta-W	**em-copy-region**
Meta-X	**ed-command**
Meta-[	**ed-sequence-lead-in**
Meta-b	**ed-prev-word**
Meta-c	**em-capitol-case**
Meta-d	**em-delete-next-word**
Meta-f	**em-next-word**
Meta-l	**em-lower-case**
Meta-n	**ed-search-next-history**
Meta-p	**ed-search-prev-history**
Meta-u	**em-upper-case**
Meta-w	**em-copy-region**
Meta-x	**ed-command**
Ctrl-Meta-?	**ed-delete-prev-word**

The remaining ascii(7) characters in the range 0x20 to 0x7e are bound to **ed-insert**.

If standard output is not connected to a terminal device or el_set(3) was used to set EL_EDITMODE to 0, all input character bindings are disabled and all characters typed are appended to the edit buffer. In that case, the edit buffer is returned to the program after a newline or carriage return character is typed,

or after the first character typed if el_set(3) was used to set EL_UNBUFFERED to non-zero.

Editor commands

Most editor commands accept an optional argument. The argument is entered by prefixing the editor command with one or more of the editor commands **ed-argument-digit**, **ed-digit**, **em-universal-argument**, or **vi-zero**. When an argument is not provided, it defaults to 1. For most editor commands, the effect of an argument is to repeatedly execute the command that number of times.

When talking about a character string from a left character to a right character, the left character is included in the string, while the right character is not included.

If an editor command causes an error, the input character is discarded, no action occurs, and the terminal bell is rung. In case of a non-fatal error, the terminal bell is also rung, but the editor command takes effect anyway.

In the following list, the default key bindings are listed after each editor command.

ed-argument-digit (vi command: 1 to 9; emacs: Meta-0 to Meta-9)
: If in argument input mode, append the input digit to the argument being read. Otherwise, switch to argument input mode and use the input digit as the most significant digit of the argument. It is an error if the input character is not a digit or if the existing argument is already greater than a million.

ed-clear-screen (vi command: Ctrl-L; emacs: Ctrl-L, Ctrl-Meta-L)
: Clear the screen and display the edit buffer at the top. Ignore any argument.

ed-command (vi command: ‘:’; emacs: Meta-X, Meta-x)
: Read a line from the terminal bypassing the normal line editing functionality and execute that line as an editrc(5) builtin command. If in vi command mode, also switch back to vi insert mode. Ignore any argument.

ed-delete-next-char (vi command: x)
: Delete the character at the cursor position. With an argument, delete that number of characters. In emacs mode, it is an error if the cursor is at the end of the edit buffer. In vi mode, the last character in the edit buffer is deleted in that case, and it is an error if the buffer is empty.

ed-delete-prev-char (vi command: X, Ctrl-H, BS, Ctrl-?, DEL)
: Delete the character to the left of the cursor position. With an argument, delete that number of characters. It is an error if the cursor is at the beginning of the edit buffer.

ed-delete-prev-word (vi: Ctrl-W; emacs: Ctrl-Meta-H, Ctrl-Meta-?)
Move to the left to the closest beginning of a word, delete the string from that position to the cursor, and save it to the cut buffer. With an argument, delete that number of words. It is an error if the cursor is at the beginning of the edit buffer.

ed-digit (emacs: 0 to 9)
If in argument input mode, append the input digit to the argument being read. Otherwise, call **ed-insert**. It is an error if the input character is not a digit or if the existing argument is already greater than a million.

ed-end-of-file (not bound by default)
Discard the edit buffer and indicate end of file to the program. Ignore any argument.

ed-ignore (various)
Discard the input character and do nothing.

ed-insert (vi input: almost all; emacs: printable characters)
In insert mode, insert the input character left of the cursor position. In replace mode, overwrite the character at the cursor and move the cursor to the right by one character position. Accept an argument to do this repeatedly. It is an error if the input character is the NUL character (Ctrl-@). Failure to enlarge the edit buffer also results in an error.

ed-kill-line (vi command: D, Ctrl-K; emacs: Ctrl-K, Ctrl-U)
Delete the string from the cursor position to the end of the line and save it to the cut buffer. Ignore any argument.

ed-move-to-beg (vi command: ^, Ctrl-A; emacs: Ctrl-A)
In vi mode, move the cursor to the first non-space character in the edit buffer. In emacs mode, move the cursor to the beginning of the edit buffer. Ignore any argument. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**.

ed-move-to-end (vi command: $, Ctrl-E; emacs: Ctrl-E)
Move the cursor to the end of the edit buffer. Ignore any argument. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**.

ed-newline (all modes: Ctrl-J, LF, Ctrl-M, CR)
Append a newline character to the edit buffer and return the edit buffer to the program. Ignore any argument.

ed-next-char (vi command: Space, l; emacs: Ctrl-F)

Move the cursor one character position to the right. With an argument, move by that number of characters. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the end of the edit buffer.

ed-next-history (vi command: j, +, Ctrl-N; emacs: Ctrl-N)
Replace the edit buffer with the next history line. That line is older than the current line. With an argument, go forward by that number of history lines. It is a non-fatal error to advance by more lines than are available.

ed-next-line (not bound by default)
Move the cursor down one line. With an argument, move down by that number of lines. It is an error if the edit buffer does not contain enough newline characters to the right of the cursor position.

ed-prev-char (vi command: h; emacs: Ctrl-B)
Move the cursor one character position to the left. With an argument, move by that number of characters. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the beginning of the edit buffer.

ed-prev-history (vi command: k, -, Ctrl-P; emacs: Ctrl-P)
Replace the edit buffer with the previous history line. That line is newer than the current line. With an argument, go back by that number of lines. It is a non-fatal error to back up by more lines than are available.

ed-prev-line (not bound by default)
Move the cursor up one line. With an argument, move up by that number of lines. It is an error if the edit buffer does not contain enough newline characters to the left of the cursor position.

ed-prev-word (emacs: Meta-B, Meta-b)
Move the cursor to the left to the closest beginning of a word. With an argument, repeat that number of times. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the beginning of the edit buffer.

ed-quoted-insert (vi insert, emacs: Ctrl-V)
Read one character from the terminal bypassing the normal line editing functionality and call **ed-insert** on it. If trying to read the character returns end of file or an error, call **ed-end-of-file** instead.

ed-redisplay (vi command, emacs: Ctrl-R)
Redisplay everything. Ignore any argument.

ed-search-next-history (vi command: J; emacs: Meta-N, Meta-n)
Replace the edit buffer with the next matching history entry.

ed-search-prev-history (vi command: K; emacs: Meta-P, Meta-p)
Replace the edit buffer with the previous matching history entry.

ed-sequence-lead-in (vi cmd: O, [; emacs: Ctrl-X; both: Meta-O, Meta-[)
Call a macro. See the section about *Macros* below for details.

ed-start-over (not bound by default)
Discard the contents of the edit buffer and start from scratch. Ignore any argument.

ed-transpose-chars (emacs: Ctrl-T)
Exchange the character at the cursor position with the one to the left of it and move the cursor to the character to the right of the two exchanged characters. Ignore any argument. It is an error if the cursor is at the beginning of the edit buffer or if the edit buffer contains less than two characters.

ed-unassigned (all characters not listed)
This editor command always results in an error.

em-capitol-case (emacs: Meta-C, Meta-c)
Capitalize the string from the cursor to the end of the current word. That is, if it contains at least one alphabetic character, convert the first alphabetic character to upper case, and convert all characters to the right of it to lower case. In any case, move the cursor to the next character after the end of the current word.

em-copy-prev-word (emacs: Ctrl-Meta-_)
Copy the string from the beginning of the current word to the cursor and insert it to the left of the cursor. Move the cursor to the character after the inserted string. It is an error if the cursor is at the beginning of the edit buffer.

em-copy-region (emacs: Meta-W, Meta-w)
Copy the string from the cursor to the mark to the cut buffer. It is an error if the mark is not set.

em-delete-next-word (emacs: Meta-D, Meta-d)
Delete the string from the cursor to the end of the current word and save it to the cut buffer. It is an error if the cursor is at the end of the edit buffer.

em-delete-or-list (emacs: Ctrl-D, EOF)

If the cursor is not at the end of the line, delete the character at the cursor. If the edit buffer is empty, indicate end of file to the program. It is an error if the cursor is at the end of the edit buffer and the edit buffer is not empty.

em-delete-prev-char (emacs: Ctrl-H, BS, Ctrl-?, DEL)
Delete the character to the left of the cursor. It is an error if the cursor is at the beginning of the edit buffer.

em-exchange-mark (not bound by default)
Exchange the cursor and the mark.

em-gosmacs-transpose (not bound by default)
Exchange the two characters to the left of the cursor. It is an error if the cursor is on the first or second character of the edit buffer.

em-inc-search-next (not bound by default)
Emacs incremental next search.

em-inc-search-prev (not bound by default)
Emacs incremental reverse search.

em-kill-line (not bound by default)
Delete the entire contents of the edit buffer and save it to the cut buffer.

em-kill-region (emacs: Ctrl-W)
Delete the string from the cursor to the mark and save it to the cut buffer. It is an error if the mark is not set.

em-lower-case (emacs: Meta-L, Meta-l)
Convert the characters from the cursor to the end of the current word to lower case.

em-meta-next (vi command, emacs: Ctrl-[, ESC)
Set the bit 0x80 on the next character typed. Unless the resulting code point is printable, holding down the ‘Meta-’ key while typing that character is a simpler way to achieve the same effect.

em-next-word (Meta-F, Meta-f)
Move the cursor to the end of the current word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the end of the edit buffer.

em-set-mark (emacs: Ctrl-Q, NUL)
Set the mark at the current cursor position.

em-toggle-overwrite (not bound by default)
Switch from insert to overwrite mode or vice versa.

em-universal-argument (not bound by default)
If in argument input mode, multiply the argument by 4. Otherwise, switch to argument input mode and set the argument to 4. It is an error if the existing argument is already greater than a million.

em-upper-case (emacs: Meta-U, Meta-u)
Convert the characters from the cursor to the end of the current word to upper case.

em-yank (emacs: Ctrl-Y)
Paste the cut buffer to the left of the cursor.

vi-add (vi command: a)
Switch to vi insert mode. Unless the cursor is already at the end of the edit buffer, move it one character position to the right.

vi-add-at-eol (vi command: A)
Switch to vi insert mode and move the cursor to the end of the edit buffer.

vi-alias (vi command: @)
If an alias function was defined by calling the el_set(3) or el_wset(3) function with the argument EL_ALIAS_TEXT, read one character from the terminal bypassing the normal line editing functionality, call the alias function passing the argument that was specified with EL_ALIAS_TEXT as the first argument and the character read, with an underscore prepended, as the second argument, and pass the string returned from the alias function to el_wpush(3). It is an error if no alias function is defined or if trying to read the character results in end of file or an error.

vi-change-case (vi command: ~)
Change the case of the character at the cursor and move the cursor one character position to the right. It is an error if the cursor is already at the end of the edit buffer.

vi-change-meta (vi command: c)
Delete the string from the cursor to the position specified by the following movement command and save a copy of it to the cut buffer. When given twice in a row, instead delete the whole

contents of the edit buffer and save a copy of it to the cut buffer. In either case, switch to vi insert mode after that.

vi-change-to-eol (vi command: C)
Delete the string from the cursor position to the end of the line and save it to the cut buffer, then switch to vi insert mode.

vi-command-mode (vi insert: Ctrl-[, ESC)
Discard pending actions and arguments and switch to vi command mode. Unless the cursor is already at the beginning of the edit buffer, move it to the left by one character position.

vi-comment-out (vi command: #)
Insert a '#' character at the beginning of the edit buffer and return the edit buffer to the program.

vi-delete-meta (vi command: d)
Delete the string from the cursor to the position specified by the following movement command and save a copy of it to the cut buffer. When given twice in a row, instead delete the whole contents of the edit buffer and save a copy of it to the cut buffer.

vi-delete-prev-char (vi insert: Ctrl-H, BS, Ctrl-?, DEL)
Delete the character to the left of the cursor. It is an error if the cursor is already at the beginning of the edit buffer.

vi-end-big-word (vi command: E)
Move the cursor to the end of the current space delimited word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the end of the edit buffer.

vi-end-word (vi command: e)
Move the cursor to the end of the current word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the end of the edit buffer.

vi-history-word (vi command: _)
Insert the first word from the most recent history entry after the cursor, move the cursor after to the character after the inserted word, and switch to vi insert mode. It is an error if there is no history entry or the most recent history entry is empty.

vi-insert (vi command: i)
Enter insert mode.

vi-insert-at-bol (vi command: I)
Move the cursor to the beginning of the edit buffer and switch to vi insert mode.

vi-kill-line-prev (vi: Ctrl-U)
Delete the string from the beginning of the edit buffer to the cursor and save it to the cut buffer.

vi-list-or-eof (vi insert: Ctrl-D, EOF)
If the edit buffer is empty, indicate end of file to the program. It is an error if the edit buffer is not empty.

vi-match (vi command: %)
Consider opening and closing parentheses, braces, and brackets as delimiters. If the cursor is not at a delimiter, move it to the right until it gets to one, then move it to the matching delimiter. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if there is no delimiter at the cursor or in the string to the right of the cursor, or if the first such delimiter has no matching delimiter.

vi-next-big-word (vi command: W)
Move the cursor to the right to the beginning of the next space delimited word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the end of the edit buffer or on its last character.

vi-next-char (vi command: f)
Read one character from the terminal bypassing the normal line editing functionality and move the cursor to the right to the next instance of that character in the edit buffer. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. If trying to read the character results in end of file or an error, call **ed-end-of-file** instead. It is an error if the character is not found searching to the right in the edit buffer.

vi-next-word (vi command: w)
Move the cursor to the right to the beginning of the next word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the end of the edit buffer or on its last character.

vi-paste-next (vi command: p)
Insert a copy of the cut buffer to the right of the cursor. It is an error if the cut buffer is empty.

vi-paste-prev (vi command: P)
Insert a copy of the cut buffer to the left of the cursor. It is an error if the cut buffer is empty.

vi-prev-big-word (vi command: B)
Move the cursor to the left to the next beginning of a space delimited word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the beginning of the edit buffer.

vi-prev-char (vi command: F)
Read one character from the terminal bypassing the normal line editing functionality and move the cursor to the left to the next instance of that character in the edit buffer. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. If trying to read the character results in end of file or an error, call **ed-end-of-file** instead. It is an error if the character is not found searching to the left in the edit buffer.

vi-prev-word (vi command: b)
Move the cursor to the left to the next beginning of a word. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. It is an error if the cursor is already at the beginning of the edit buffer.

vi-redo (vi command: '.')
Redo the last non-motion command.

vi-repeat-next-char (vi command: ';')
Repeat the most recent character search in the same search direction. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**.

vi-repeat-prev-char (vi command: ',')
Repeat the most recent character search in the opposite search direction. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**.

vi-repeat-search-next (vi command: n)
Repeat the most recent history search in the same search direction.

vi-repeat-search-prev (vi command: N)
Repeat the most recent history search in the opposite search direction.

vi-replace-char (vi command: r)
Switch to vi replace mode, and automatically switch back to vi command mode after the next character typed. See **ed-insert** for a description of replace mode. It is an error if the cursor is at the end of the edit buffer.

vi-replace-mode (vi command: R)

Switch to vi replace mode. This is a variant of vi insert mode; see **ed-insert** for the difference.

vi-search-next (vi command: ?)
Replace the edit buffer with the next matching history entry.

vi-search-prev (vi command: /)
Replace the edit buffer with the previous matching history entry.

vi-substitute-char (vi command: s)
Delete the character at the cursor and switch to vi insert mode.

vi-substitute-line (vi command: S)
Delete the entire contents of the edit buffer, save a copy of it in the cut buffer, and enter vi insert mode.

vi-to-column (vi command: |)
Move the cursor to the column specified as the argument. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**.

vi-to-history-line (vi command: G)
Replace the edit buffer with the specified history entry.

vi-to-next-char (vi command: t)
Read one character from the terminal bypassing the normal line editing functionality and move the cursor to the right to the character before the next instance of that character in the edit buffer. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. If trying to read the character results in end of file or an error, call **ed-end-of-file** instead. It is an error if the character is not found searching to the right in the edit buffer.

vi-to-prev-char (vi command: T)
Read one character from the terminal bypassing the normal line editing functionality and move the cursor to the left to the character after the next instance of that character in the edit buffer. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**. If trying to read the character results in end of file or an error, call **ed-end-of-file** instead. It is an error if the character is not found searching to the left in the edit buffer.

vi-undo (vi command: u)
Undo the last change.

vi-undo-line (vi command: U)

Undo all changes to the edit buffer.

vi-yank (vi command: y)
Copy the string from the cursor to the position specified by the following movement command to the cut buffer. When given twice in a row, instead copy the whole contents of the edit buffer to the cut buffer.

vi-yank-end (vi command: Y)
Copy the string from the cursor to the end of the edit buffer to the cut buffer.

vi-zero (vi command: 0)
If in argument input mode, multiply the argument by ten. Otherwise, move the cursor to the beginning of the edit buffer. Can be used as a movement command after **vi_change_meta**, **vi_delete_meta**, or **vi_yank**.

Macros

If an input character is bound to the editor command **ed-sequence-lead-in**, **editline** attempts to call a macro. If the input character by itself forms the name of a macro, that macro is executed. Otherwise, additional input characters are read until the string read forms the name of a macro, in which case that macro is executed, or until the string read matches the beginning of none of the existing macro names, in which case the string including the final, mismatching character is discarded and the terminal bell is rung.

There are two kinds of macros. Command macros execute a single editor command. Keyboard macros return a string of characters that is appended as a new line to the *Input Queue*.

The following command macros are defined by default in vi command mode and in emacs mode:

Esc [A, Esc O A	**ed-prev-history**
Esc [B, Esc O B	**ed-next-history**
Esc [C, Esc O C	**ed-next-char**
Esc [D, Esc O D	**ed-prev-char**
Esc [F, Esc O F	**ed-move-to-end**
Esc [H, Esc O H	**ed-move-to-beg**

In vi command mode, they are also defined by default without the initial escape character.

In addition, the **editline** library tries to bind the strings generated by the arrow keys as reported by the terminfo(5) database to these editor commands, unless that would clobber user settings.

In emacs mode, the two-character string "Ctrl-X Ctrl-X" is bound to the **em-exchange-mark** editor command.

Input Queue

The **editline** library maintains an input queue operated in FIFO mode. Whenever it needs an input character, it takes the first character from the first line of the input queue. When the queue is empty, it reads from the terminal.

A line can be appended to the end of the input queue in several ways:

- By calling one of the keyboard *Macros*.
- By calling the editor command **vi-redo**.
- By calling the editor command **vi-alias**.
- By pressing a key in emacs incremental search mode that doesn't have a special meaning in that mode but returns to normal emacs mode.
- If an application program directly calls the functions el_push(3) or el_wpush(3), it can provide additional, program-specific ways of appending to the input queue.

SEE ALSO

mg(1), vi(1), editline(3), el_wgets(3), el_wpush(3), el_wset(3), editrc(5)

HISTORY

This manual page first appeared in OpenBSD 6.0 and NetBSD 8.

AUTHORS

This manual page was written by Ingo Schwarze <*schwarze@openbsd.org*>.