NAME

editline − command-line editing library with history

SYNOPSIS

char *readline(char ***_prompt_)**

DESCRIPTION

Editline is a library that provides an line-editing interface with text recall. It is intended to be compatible with the *readline* library provided by the Free Software Foundation, but much smaller. The bulk of this manual page describes the user interface.

The *readline* routine returns a line of text with the trailing newline removed. The data is returned in a buffer allocated with *malloc*(3), so the space should be released with *free*(3) when the calling program is done with it. Before accepting input from the user, the specified *prompt* is displayed on the terminal.

Each line returned is copied to the internal history list, unless it happens to be equal to the previous line.

User Interface

A program that uses this library provides a simple emacs-like editing interface to its users. A line may be edited before it is sent to the calling program by typing either control characters or escape sequences. A control character, shown as a caret followed by a letter, is typed by holding down the ‘‘control’’ key while the letter is typed. For example, ‘‘^A’’ is a control-A. An escape sequence is entered by typing the ‘‘escape’’ key followed by one or more characters. The escape key is abbreviated as ‘‘ESC.’’ Note that unlike control keys, case matters in escape sequences; ‘‘ESC F’’ is not the same as ‘‘ESC f’’.

An editing command may be typed anywhere on the line, not just at the beginning. In addition, a return may also be typed anywhere on the line, not just at the end.

Most editing commands may be given a repeat count, *n*, where *n* is a number. To enter a repeat count, type the escape key, the number, and then the command to execute. For example, ‘‘ESC 4 ^f’’ moves forward four characters. If a command may be given a repeat count then the text ‘‘[n]’’ is given at the end of its description.

The following control characters are accepted:

^A	Move to the beginning of the line
^B	Move left (backwards) [n]
^D	Delete character [n]
^E	Move to end of line
^F	Move right (forwards) [n]
^G	Ring the bell
^H	Delete character before cursor (backspace key) [n]
^I	Complete filename (tab key); see below
^J	Done with line (return key)
^K	Kill to end of line (or column [n])
^L	Redisplay line
^M	Done with line (alternate return key)
^N	Get next line from history [n]
^P	Get previous line from history [n]
^R	Search backward (forward if [n]) through history for text; must start line if text begins with an uparrow
^T	Transpose characters
^V	Insert next character, even if it is an edit command
^W	Wipe to the mark
^X^X	Exchange current location and mark
^Y	Yank back last killed text
^[	Start an escape sequence (escape key)
^]c	Move forward to next character ‘‘c’’
^?	Delete character before cursor (delete key) [n]

The following escape sequences are provided.

ESC ^H	Delete previous word (backspace key) [n]
ESC DEL	Delete previous word (delete key) [n]
ESC SP	Set the mark (space key); see ^X^X and ^Y above
ESC .	Get the last (or [n]'th) word from previous line
ESC ?	Show possible completions; see below
ESC <	Move to start of history
ESC >	Move to end of history
ESC b	Move backward a word [n]
ESC d	Delete word under cursor [n]
ESC f	Move forward a word [n]
ESC l	Make word lowercase [n]
ESC m	Toggle if 8bit chars display normally or with ''M−'' prefix
ESC u	Make word uppercase [n]
ESC y	Yank back last killed text
ESC v	Show library version
ESC w	Make area up to mark yankable
ESC nn	Set repeat count to the number nn
ESC C	Read from environment variable ''_C_'', where C is an uppercase letter

The *editline* library has a small macro facility. If you type the escape key followed by an uppercase letter, *C*, then the contents of the environment variable *_C_* are read in as if you had typed them at the keyboard. For example, if the variable *_L_* contains the following:

    ^A^Kecho '^V^[[H^V^[[2J'^M

Then typing ''ESC L'' will move to the beginning of the line, kill the entire line, enter the echo command needed to clear the terminal (if your terminal is like a VT-100), and send the line back to the shell.

The *editline* library also does filename completion. Suppose the root directory has the following files in it:

bin	vmunix
core	vmunix.old

If you type ''rm /v'' and then the tab key. *Editline* will then finish off as much of the name as possible by adding ''munix''. Because the name is not unique, it will then beep. If you type the escape key and a question mark, it will display the two choices. If you then type a period and a tab, the library will finish off the filename for you:

rm /v[TAB]*munix*.TAB*old*

The tab key is shown by ''[TAB]'' and the automatically-entered text is shown in italics.

BUGS AND LIMITATIONS

Doesn't know how to handle multiple lines.

AUTHORS

Simmule R. Turner <uunet.uu.net!capitol!sysgo!simmy> and Rich $alz <rsalz@osf.org>. Original manual page by DaviD W. Sanderson <dws@ssec.wisc.edu>.