

NAME

readline – command line recall and editing

DESCRIPTION

The GNU *readline* library or the BSD *libedit* library may be optionally linked with `spec` during the `spec` installation. When `spec` is linked with either of these libraries, the following command completion, history substitution and command line editing features become available.

COMMAND COMPLETION

The `<tab>` key can be used to complete the current word being typed. The completion is context sensitive. If completion is asked for on the first word of a command, possible completions are all of `spec`'s built-in keywords and currently defined macros.

Possible completion on words following the commands `lsdef`, `prdef` and `undef` are the currently defined macros. Possible completion on words following the commands `syms` and `unglobal` include all the current symbol names. Completion on words following `lscmd` are the built-in keywords. Possible completions to `spec_par()` are the available `spec_par()` options. In all other cases, possible completions on words following the first word of a command are all the file and directory names in the current directory.

If only one completion is possible, typing the `<tab>` key will generate the completion. If a small number of completions are available, typing the `<tab>` key twice will list them, followed by a redisplay of the current input line. If a large number of completions are possible, typing the `<tab>` will generate a prompt asking if all should be displayed.

TILDE EXPANSION

The sequences `~/` and `~any_user` will be replaced by the full path names for the current user's and any user's home directory when those characters are used in the arguments to the `open()`, `close()`, `on()`, `off()`, `dofile()`, `qdofile()`, `file_info()`, `chdir()` and `unix()` functions. The same applies when the tilde-containing arguments are passed to the functions through macros, such as through the standard `newfile`, `do`, `cd`, `u`, etc. macros.

Note, as of `spec` release 5.04.04, tilde expansion is available whether or not `spec` is linked with the command-line editing libraries.

HISTORY SUBSTITUTION - EVENT SPECIFIERS

All or part of a previous command can be recalled for the current command using a string starting with the `!` event specifier character. The following event specifiers may appear anywhere in the input line.

`!!` – The previous command.

`!N` – Command number *N*.

`!-N` – The *N*th previous command.

`!string` – The most recent command starting with *string*.

`!?string[?]` – The most recent command containing *string*. The trailing `?` is optional at the end of a line.

HISTORY SUBSTITUTION - WORD DESIGNATORS

Individual words from previous commands can be recalled by adding a word designator to the event specifier. A colon (`:`) separates the two components and can be omitted if the word designator begins with the characters `^`, `$`, `*` or `%`. Words are numbered from the beginning of the line, starting with a 0 (zero).

`0` – The 0th word.

- N* – The *N*th word.
- \wedge – Word 1. \wedge refers to word 1 of the previous command.
- $\$$ – The last word. $\!\$$ refers to the last word of the previous command.
- $\%$ – The word most recently matched by a *?string?*.
- N-M* – A range of words; $-N$ Abbreviates $0-N$.
- $*$ – All of the words, except the 0th. This is the same as $1-\$$. If there is just one word (the 0th) in the event, an empty string is substituted.
- N** – Same as $N-\$$.
- $N-$ – Same as $N-\$$, but doesn't include the last word.

HISTORY SUBSTITUTION - MODIFIERS

One or more modifiers may follow the optional word designator, each preceded by a $:$, as follows:

- $\#$ – The entire command line typed so far.
 - h* – Remove a trailing path name component of the form */pathname*.
 - r* – Remove a trailing suffix of the form *.suffix*.
 - e* – Remove all but the suffix.
 - t* – Remove all leading path name components.
 - s/left/right[/]* – Substitute the first occurrence of the string *left* with the string *right*. The pattern delimiter can be any character. An $\&$ character in *right* is replaced by *left*, unless the $\&$ is preceded by a single backslash. The trailing delimiter is optional if *right* is at the end of the line.
 - g* – When used with *s*, apply the substitution to each occurrence of the string *left*.
 - p* – Print the new command but do not execute it.
- The usage $\wedge left \wedge right \wedge$ is shorthand for $!!:s/left/right/$.

COMMAND LINE EDITING

The following special keys can be used to recall previous command lines and to modify the current command line. In addition to the control keys described below, the up and down arrow keys move through the history, while the left and right arrow keys move the cursor within the currently displayed line.

The command names listed for readline and libedit are the names you would use in an optional key-bindings file. See the documentation for readline or libedit for information on setting up their respective key bindings files. *spec* will arrange for a key-bindings file named *.spec_keys* in the *spec* user's home directory to be loaded, if it exists.

- $\wedge A$ – Move to the beginning of the current line.
 readline: *beginning-of-line*
 libedit: *ed-move-to-beg*
- $\wedge E$ – Move to the end of the current line.
 readline: *end-of-line*
 libedit: *ed-move-to-end*
- ESC *f* – Move forward to the end of the next word.
 readline: *forward-word*
 libedit: *em-next-word*

- ESC b – Move to the start of the current or previous word.
readline: *backward-word*
libedit: *ed-prev-word*
- ^L – Clear the screen and reprint the current line at the top.
readline: *clear-screen*
libedit: *ed-clear-screen*
- ^K – Kill from the cursor to the end of the line.
readline: *kill-line*
libedit: *ed-kill-line*
- ESC d – Kill from the cursor to the end of the current word. If between words, kill to the end of the next word.
readline: *kill-word*
libedit: *em-delete-next-word*
- ESC DEL – Kill from the cursor the start of the previous word. If between words, kill to the start of the previous word.
readline: *backward-kill-word*
libedit: *ed-delete-prev-word*
- ^W – Kill from the cursor to the previous white space.
readline: *unix-word-rubout*
libedit: *em_kill_region*
- ^Y – Insert the most recently killed text at the cursor.
readline: *yank*
libedit: *em-yank*
- ESC y – Rotate the kill-ring, and yank the new top. You can only do this if the previous command was ^Y or ESC y.
readline: *yank-pop*
libedit: not available
- ^F – Move forward a character.
readline: *forward-char*
libedit: *ed-next-char*
- ^B – Move back a character.
readline: *backward-char*
libedit: *ed-prev-char*
- <return> – Accept the line regardless of where the cursor is. If non-empty, add the line to the history list.
readline: *accept-line*
libedit: *ed-newline*
- ^P – Move up through the history list.
readline: *previous-history*
libedit: *ed-prev-history*
- ^N – Move down through the history list.
readline: *next-history*
libedit: *ed-next-history*
- ESC < – Move to the first line of the history.
readline: *beginning-of-history*
libedit: not available

- ESC > – Move to last line of the history.
readline: *end-of-history*
libedit: not available
- ^R – Search backward, incrementally, starting at the current line and moving up through the history as necessary.
readline: *reverse-search-history*
libedit: *em-inc-search-prev*
- ^S – Search forward starting at the current line and moving down through the the history as necessary.
readline: *forward-search-history*
libedit: *em-inc-search-next*
- ^D – Delete the character under the cursor.
readline: *delete-char*
libedit: *em-delete-or-list*
- ^H, DEL – Delete the character behind the cursor. A numeric argument means kill the characters instead of deleting them.
readline: *backward-delete-char*
libedit: *ed-delete-prev-char*
- ^Q, ^V – Add the literal next character to the line.
readline: *quoted-insert*
libedit: *ed-quoted-insert*
- a, b, etc. – Characters to be inserted as themselves.
readline: *self-insert*
libedit: *ed-insert*
- ^T – Transpose the previous and current characters and move forward one character.
readline: *transpose-chars*
libedit: *ed-transpose-chars*
- ESC t – Transpose the previous and current words and move forward one word.
readline: *transpose-words*
libedit: not available
- ESC ^_ – Insert a copy of the previous word
readline: not available
libedit: *em_copy_prev-word*
- ESC u – Capitalize the current word.
readline: *upcase-word*
libedit: *em-upper-case*
- ESC l – Make the current word lower case.
readline: *downcase-word*
libedit: *em-lower-case*
- ESC c – Capitalize the current word.
readline: *capitalize-word*
libedit: *em-capitol-case*
- unbound* – Kill backward to the beginning of the line.
readline: *backward-kill-line*
libedit: not available
- ^U – Kill the current line.
readline: *unix-line-discard*

libedit: *em-kill-line*

ESC 0, ESC 1, ... – Enter number arguments for commands that can operate on more than one object.
 readline: *digit-argument*
 libedit: *ed-argument-digit*

^G, ESC ^G, ^X ^G – Stop.
 readline: *abort*
 libedit: no binding, but ^G aborts incremental seaches

ESC – Use this character as a *meta* prefix for keyboards that lack a meta key.
 readline: *prefix-meta*
 libedit: *em-meta-next*

^_, ^X ^U – Undo last change.
 readline: *undo*, is incremental and remembered for each line
 libedit: *vi-undo*, only one level of undo

ESC r, ESC ^R – Undo all changes made to this line.
 readline: *revert-line*
 libedit: *vi-undo-line*

^X (– Save the key strokes that follow.
 readline: *start-kbd-macro*
 libedit: not available

^X) – Cease saving key strokes.
 readline: *end-kbd-macro*
 libedit: not available

^X e – Recall the previously saved key strokes.
 readline: *call-last-kbd-macro*
 libedit: not available

unbound – Redraw the current line.
 readline: *redraw-current-line*
 libedit: *ed-redisplay*

ESC ^Y –
 readline: *yank-nth-arg*
 libedit: not available

unbound – List the current key bindings.
 readline: *dump-functions*
 libedit: not available

^X ^R – Reread the *\$HOME/.spec_keys* key-bindings file, if it exists.
 readline: *re-read-init-file*
 libedit: not available

NOTES

With the optional history libraries, the ! character must be preceded by a \ when it is typed from the keyboard and is not intended to be part of a history substitution.

DISTRIBUTION

The readline library is included with most modern *Linux* distributions. In addition, sources can be retrieved from ftp.gnu.org/gnu/readline and installed on any platform supported by spec. spec should link with all current and previous releases of readline. spec is known to link with readline versions 2.0, 2.1, 2.2, 2.2.1, 4.0, 4.1, 4.2, 4.2a and 4.3.

A prebuilt version of the NetBSD libedit library is included with the spec distribution in the *libedit* subdirectory. The official source repository for the NetBSD version is at <http://cvsweb.netbsd.org/bsdweb.cgi/src/lib/libedit>. Portable versions configurable for other platforms are available from www.thrysoee.dk/editline, although that site doesn't always have the most recent version. The source for the prebuilt version included with the spec distribution is available at <ftp.certif.com/pub/misc>.