READLINE REFERENCE

CONTENTS

DEFINITIONS

This online reference card describes the **readline** library that comes with version 2.02.0 of **bash**. It is a companion to SSC's BASH REFERENCE, which simply didn't have room for the full **readline** description.

Several typefaces are used to clarify the meaning:

- **Serifa Bold** is used for computer input.
- *Serifa Italic* is used to indicate user input and for syntactic placeholders, such as *variable* or *cmd*.
- Serifa Roman is used for explanatory text.

blank – separator between words. Blanks consist of one or more spaces and/or tab characters. In addition, words are terminated by any of the following characters:

; & ( ) | < > space tab newline

n – an integer.

name – a variable, alias, function or command name.

word – a generic argument; a word. Quoting may be necessary if it contains special characters.

This reference card was written by Arnold Robbins. We thank Chet Ramey (**bash**'s maintainer) for his help.

READLINE

The **readline** library implements command-line editing. By default, it provides an *emacs* editing interface, although a *vi* interface is available. **readline** is initialized either from the file named by **$INPUTRC** (if set), or from **~/.inputrc**. In that file, you can use conditionals, define key bindings for macros and functions, and set variables.

From the **bash** level, the **bind** command allows you to add, remove and change macro and key bindings. There are five input mode map names that control the action taken for each input character. The map names are **emacs**, **emacs-standard**, **emacs-meta**, **emacs-ctlx**, **vi**, **vi-command**, and **vi-insert**. **emacs** is the same as **emacs-standard**, and **vi** is the same as **vi-command**.

You choose which editor you prefer with **set –o emacs** or **set –o vi** in your **~/.bashrc** file, or at runtime.

readline understands the character names *DEL*, *ESC*, *LFD*, *NEWLINE*, *RET*, *RETURN*, *RUBOUT*, *SPACE*, *SPC* and *TAB*.

READLINE DIRECTIVES

Directives in the **.inputrc** file provide conditionals and include facilities similar to the C preprocessor.

$include
include a file, e.g., a system-wide **/etc/inputrc** file

$if
start a conditional, for terminal or application-specific settings. You can test the following:

application=	test the application, e.g., **bash** or **gdb**
mode=	test the editing mode, *emacs* or *vi*
term=	test the terminal type

The use of **application=** is optional; e.g., **$if Bash**

$else
start the “else” part of a conditional

$endif
finish a conditional

READLINE KEY BINDINGS

Keys bound to a macro place the macro text into the input; keys bound to a function run the function.

You can use these escape sequences in bindings:

\a	alert (bell)	**\r**	carriage return
\b	backspace	**\t**	horizontal tab (TAB)
\C–	control prefix	**\v**	vertical tab
\d	delete (DEL)	\\	backslash
\e	escape (ESC)	\"	literal "
\f	form feed	\´	literal ´
\M–	meta prefix	*ddd*	octal value *ddd*
\n	newline	**\x***hhh*	hex value *hhh*

Macros and function bindings look like:

macro:	*key-seq*:"*text*"
function:	*key-seq*:*function-name*

Macros have quoted text on the right of the colon; functions have function names. A *key-seq* is either a single character or character name (such as **Control–o**), or a quoted string of characters (single or double quotes).

READLINE VARIABLES

Variables control different aspects of **readline**'s behavior. You set a variable with

set *variable value*

Unless otherwise noted, *value* should be either **On** or **Off**. The descriptions below describe the effect when the variable is **On**. Default values are shown in parentheses.

bell-style (**audible**)
defines how **readline** should ring the bell:

audible	ring the bell
none	never ring the bell
visible	flash the screen

comment-begin (**#**)
insert this string for **readline-insert-comment** (bound to **M-#** in *emacs* mode and to **#** in *vi* mode)
completion-ignore-case (**Off**)
ignore case when doing completions
completion-query-items (**100**)
if the number of completion items is less than this value, place them in the command line. Otherwise, ask the user if they should be shown
convert-meta (**On**)
treat characters with the eighth bit set as the meta version of the equivalent seven-bit character
disable-completion (**Off**)
do not do completion
editing-mode (**emacs**)
set the initial editing mode. Possible values are **emacs** or **vi**
enable-keypad (**Off**)
attempt to enable the application keypad. This may be needed to make the arrow keys work
expand-tilde (**Off**)
attempt tilde expansion as part of word completion
input-meta (**Off**)
meta-flag (**Off**)
enable eight bit input. The two variable names are synonyms
keymap (**emacs**)
set the current keymap. See Readline for a list of allowed values. The **editing-mode** variable also affects the keymap
mark-directories (**On**)
append a **/** to completed directory names
mark-modified-lines (**Off**)
place a ***** at the front of modified history lines
output-meta (**Off**)
print characters with the eighth bit set directly, not as **M-***x*
print-completions-horizontally (**Off**)
display completions horizontally, with the matches sorted alphabetically, instead of vertically down the screen
show-all-if-ambiguous (**Off**)
immediately list words with multiple possible completions, instead of ringing the bell
visible-stats (**Off**)
when listing possible completions, append a character that denotes the file's type

READLINE EMACS MODE

Every regular character you type goes into the input line. Control characters and meta-characters move the cursor or perform editing operations. **C-** precedes control keys. **M-** precedes meta-characters. Case matters only for meta-characters. You can have meta-control characters.

The *mark* is a saved position on the line. Many operations work relative to the current position (*point*) and the mark. Text between them is called the *region*.

Numeric parameters give a repeat count for the command. To enter a numeric parameter, press ESC, the number, and then the command character.

The descriptions below show the default key binding with the function name and description. □ indicates an unbound function.

History Search Commands

CR	**accept-line**. Run the command (carriage return or linefeed)
C-p	**previous-history**. Get previous history line
C-n	**next-history**. Get next history line
M-<	**beginning-of-history**. Get oldest history line
M->	**end-of-history**. Get youngest history line
C-r	**reverse-search-history**. Incrementally search backward (up) through history
C-s	**forward-search-history**. Incrementally search forward (down) through history
M-p	**non-incremental-reverse-search-history**. Non-incrementally search backward (up) through history
M-n	**non-incremental-forward-search-history**. Non-incrementally search forward (down) through history
□	**history-search-backward**. Non-incremental search backward (up) through history for the text between the start of the line and point
□	**history-search-forward**. Non-incremental search forward (down) through history for the text between the start of the line and point
M-C-y	**yank-nth-arg**. With argument, retrieve *n*'th argument from previous command. Count starts at 0, default is 1. Negative count goes from left
M-.	**yank-last-arg**. Insert last argument from previous command. With argument, just like **yank-nth-arg**. Successive commands retrieve the last argument from successively older commands. **insert-last-arg** is another name
M-_	**yank-last-arg**
M-C-e	**shell-expand-line**. Expand the line the way the shell would
M-^	**history-expand-line**. Do history substitution on the current line
□	**magic-space**. Do history substitution on the current line and insert a space
□	**alias-expand-line**. Do alias expansion on the current line
□	**history-and-alias-expand-line**. Do history and alias expansion on the current line
C-o	**operate-and-get-next**. Execute current line and fetch next history line. Any arguments are ignored

READLINE EMACS MODE (continued)

Line Change Commands

C-d	**delete-char**. Delete the character under the cursor. At the beginning of the line with no characters, generate EOF
DEL	**backward-delete-char**. Delete the character left of the cursor. With argument, save the text on the kill-ring
C-q	**quoted-insert**. Treat the next character literally
C-v	**quoted-insert**
C-v TAB	**tab-insert**. Insert a tab character
Any key	**self-insert**. Insert the typed character. All regular characters are bound to this function
C-t	**transpose-chars**. Transpose the current and previous characters and advance the cursor
M-t	**transpose-words**. Transpose the current and previous words and advance the cursor
M-u	**upcase-word**. Uppercase the current or next word. With negative argument, uppercase the previous word, but don't move point
M-l	**downcase-word**. Lowercase the current or next word. With negative argument, lowercase the previous word, but don't move point
M-c	**capitalize-word**. Capitalize the current or next word. With negative argument, capitalize the previous word, but don't move point

Killing and Yanking

C-k	**kill-line**. Kill the text from point to the end of the line
C-x DEL	**backward-kill-line**. Kill backwards to the beginning of the line
C-u	**unix-line-discard**. Kill backward from point to the beginning of the line, save the text on the kill-ring
□	**kill-whole-line**. Kill the whole line, no matter where cursor is
M-d	**kill-word**. Kill from cursor to end of current or next word
M-DEL	**backward-kill-word**. Kill the word in front of the cursor
C-w	**unix-word-rubout**. Kill the word in front of the cursor, using whitepsace as the boundary
M-	**delete-horizontal-space**. Delete all spaces and tabs around point
□	**kill-region**. Kill the text between point and mark
□	**copy-region-as-kill**. Copy the region to the kill buffer
□	**copy-backward-word**. Copy the word before point to the kill buffer
□	**copy-forward-word**. Copy the word after point to the kill buffer
C-y	**yank**. Yank the top of the kill-ring into the **readline** buffer at the current cursor position
M-y	**yank-pop**. Rotate the kill-ring, and yank the new top into the **readline** buffer at the current cursor position

READLINE EMACS MODE (continued)

Completing

TAB	**complete**. Attempt variable, username, hostname or command (including alias and function) completion. If no match, attempt filename completion
M-?	**possible-completions**. List the possible completions for the text before point
M-*	**insert-completions**. Insert all the completions that **possible-completions** would generate
□	**menu-complete**. Like **complete**, but cycles through the list of possible completions
M-/	**complete-filename**. Attempt filename completion on the text before point
C-x /	**possible-filename-completions**. List possible filename completions for the text before point
M-~	**complete-username**. Attempt username completion on the text before point
C-x ~	**possible-username-completions**. List possible username completions for the text before point
M-$	**complete-variable**. Attempt variable completion on the text before point
C-x $	**possible-variable-completions**. List possible shell variable completions for the text before point
M-@	**complete-hostname**. Attempt hostname completion on the text before point
C-x @	**possible-hostname-completions**. List possible hostname completions for the text before point
M-!	**complete-command**. Attempt command completion on the text before point. Try aliases, reserved words, functions, built-ins and external commands
C-x !	**possible-command-completions**. List possible command completions for the text before point
M-TAB	**dynamic-complete-history**. Attempt to complete text before point with history lines
M-{	**complete-into-braces**. Perform filename completion, returning the list enclosed in braces for use in brace expansion

Keyboard Macros

C-x (	**start-kbd-macro**. Begin saving characters typed into the current keyboard macro
C-x)	**end-kbd-macro**. Stop saving characters typed into the current keyboard macro and store the definition
C-x e	**call-last-kbd-macro**. Execute the last keyboard macro defined, as if the saved characters had been typed at the keyboard

Cursor Motion Commands

C-a	**beginning-of-line**. Move to start of line
C-e	**end-of-line**. Move to end of line
C-f	**forward-char**. Move forward one character
C-b	**backward-char**. Move backward one character
M-f	**forward-word**. Move forward one word
M-b	**backward-word**. Move backward one word
C-l	**clear-screen**. Clear the screen. With argument, just redraw the current line
□	**redraw-current-line**. Refresh the current line

READLINE EMACS MODE (continued)

Numeric Arguments

M-0, ...	**digit-argument**. Bound to **M-0**, **M-1**, etc. Add the digit to the accumulating argument. **M--** (meta-minus) starts a negative argument
□	**universal-argument**. Start accumulating a numeric argument, with optional leading sign. Executing **universal-argument** again ends the argument. With no digits, the default argument is four

Miscellaneous

C-x C-r	**re-read-init-file**. Read the **inputrc** file, adding new bindings or variable settings
C-g	**abort**. Abort the current editing command and ring the bell
M-*x*, ...	**do-uppercase-version**. If the metafied character *x* is lowercase, run the command bound to the corresponding uppercase character
ESC	**prefix-meta**. Metafy the next character typed
C-_	**undo**. Incremental undo, remembered separately for each line
C-x C-u	**undo**.
M-r	**revert-line** Undo all changes made to this line
M-&	**tilde-expand**. Attempt tilde expansion on the current word
C-@	**set-mark**. Set the mark to the current point. With argument, set it to that position
M-SPC	**set-mark**.
C-x C-x	**exchange-point-and-mark**. Swap point and mark
C-]	**character-search**. Read a character and move to the next occurrence of that character. With negative argument, search backwards
M-C-]	**character-search-backward**. Read a character and move to the previous occurrence of that character. With negative argument, search forwards
M-#	**insert-comment**. Insert the value of the **comment-begin** variable at the beginning of the current line, which is then accepted
C-x *	**glob-expand-word**. Filename expand the word before the cursor and insert the resulting list
C-x g	**glob-list-expansion**. Display the list that **glob-expand-word** would produce, then redraw the line
□	**dump-functions**. Print all functions and their key bindings. With numeric argument, print in **inputrc** format
□	**dump-variables**. Print all settable variables and their values. With numeric argument, print in **inputrc** format
□	**dump-macros**. Print all macros and their key bindings. With numeric argument, print in **inputrc** format
C-x C-v	**display-shell-version**. Display **bash** version information

READLINE VI MODE

Insert mode is the default. Press ESC to enter command mode. Press CR to run the command and return to insert mode. If the **shopt** option **cmdhist** is set, you may edit multi-line commands. Preceding a **vi** command with a number provides a repeat count.

Function names for bindings are omitted to save space.

Input Editing Commands

ESC	terminate insert mode (begin command mode)
CR	(carriage return or line-feed) run command(s)
INTR	(**stty**(1) **intr** character) start over
^U	delete everything to the left of the cursor
^W	delete the previous *blank*-separated word
^D	end-of-file, end the session
^Q	escape the next character
^V	escape the next character

History Search Commands

[*n*]**k**	get previous command; successive **k**'s keep going backward (older commands)
[*n*]**–**	same as **k**
[*n*]**j**	get next command; successive **j**'s keep going forward (newer commands)
[*n*]**+**	same as **j**
[*n*]**G**	get command number *n*
/*string*	search backward (older commands) for *string*. Use ^*string* to match *string* at the beginning of a line
?*string*	search forward (newer commands) for *string*
n	find next match of last **/** or **?** pattern
N	like **n**, but in the opposite direction

Text Modification Commands

a	enter input mode, appending after current character
A	enter input mode at end of line, same as **$a**
[*n*]**c***motion*	
c[*n*]*motion*	delete from current character through character that *motion* moves to, and enter input mode; if *motion* is **c**, delete the whole line and enter input mode
C	change from current character through end of the line; same as **c$**
[*n*]**s**	replace characters under the cursor; enters input mode
S	same as **cc** (change the whole line)
[*n*]**d***motion*	
d[*n*]*motion*	delete from current character through character that *motion* moves to; if *motion* is **d**, delete the whole line
D	delete from current character through end of line; same as **d$**
i	enter input mode, inserting before current character
I	enter input mode at beginning of line (like **0i**)
[*n*]**p**	append previous text change after cursor
[*n*]**P**	insert previous text change before cursor
[*n*]**r***c*	replace *n* characters with *c*
R	enter overlay mode, replacing characters until pressing ESC
[*n*]**x**	delete current character
[*n*]**X**	delete previous character
[*n*]**.**	repeat last command
[*n*]**~**	invert the case of *n* characters

Text Modification Commands (continued)

[*n*]_	insert *n*'th word of previous shell command and enter insert mode; use the last word if no *n* (not in real **vi**)
*	append * to current word and do filename expansion, replacing current word with matching filenames, then enter insert mode
\	replace current word with longest unique prefix of matching filenames; if unique, append / if directory, otherwise append a space, and enter insert mode
&	do tilde expansion on current word

Motion Commands

[*n*]**l**	forward one character
[*n*]**w**	forward one alpha-numeric word
[*n*]**W**	move to next word after *blank*
[*n*]**e**	move to end of word
[*n*]**E**	move to end of word before *blank*
[*n*]**h**	backward one character
[*n*]**b**	backward one word
[*n*]**B**	backward one *blank* delimited word
[*n*]**|**	move to column *n*
[*n*]**f***c*	find next *c*
[*n*]**F***c*	find previous *c*
[*n*]**t***c*	like **f** followed by **h**
[*n*]**T***c*	like **F** followed by **l**
[*n*]**;**	do last **f**, **F**, **t**, or **T**, *n* times
[*n*]**,**	like **;** but in opposite direction
0	move to start of line
^	move to first non-*blank* character of line
$	move to end of line
%	find balancing (,), {, }, [, or]

Other Commands

[*n*]**y***motion*	
y[*n*]*motion*	yank from current character through where *motion* would go to
yy	yank the whole line
Y	yank from current character through end of line; same as **y$** (differs from real **vi**)
u	undo last command
U	undo all editing done to the line
[*n*]**v**	put **fc −e ${VISUAL:−${EDITOR:−vi}}** *n* into input and run it (net effect is to run an editor on the current line and to execute the results when editing is finished)
^L	clear the screen and re-print current line
=	list files that would match the current word if a * were to be appended; doesn't modify line
#	put a # at the front of the line and send it; used mainly to save a line in the history without executing it
@*letter*	macro expansion; look for an alias named *_letter* and, if found, read the value as command mode input
m*letter*	save the current position in the mark named by *letter*, which must be uppercase
`*letter*	move to the mark previously saved in *letter*, which must be uppercase

On the first word, *, \, and = expand aliases, functions, and commands.