tcsh Command Editor Commands

Paul DuBois
dubois@primate.wisc.edu
19 May 1995

This document started out as part of one of the appendices to the O'Reilly & Associates handbook *Using csh & tcsh*, but it quickly got too long, so it exists separately. Background for the command editor may be found in that handbook (Chapter 7, *The tcsh Command Editor*), and in the *Editing* section of the *tcsh* manual page.

This document provides information about the following:

- The names of the *tcsh* command line editing commands and what they do. These are the commands that can be bound to keys with *bindkey*.
- The default key bindings.

The *tcsh* command editor provides two sets of command bindings. One set is patterned after the *emacs* editor, the other is patterned after the *vi* editor. At most sites, the command editor uses the *emacs* bindings by default, but you can select one set or the other explicitly using one of the following commands in your *~/.cshrc* (or *~/.tcshrc*) file:

`bindkey -e`	Select *emacs* bindings
`bindkey -v`	Select *vi* bindings

Some commands are available only in one mode or the other. If the description of a command is marked "*emacs* only" or "*vi* only", it means the command is available only in the given editing mode. Otherwise it's likely available in both (although I haven't verified this for every unmarked command).

The default key bindings are listed at the end of this document. You can also use *bindkey* to give you online help:

- To find out what bindings are actually in effect:

  ```
  % bindkey
  ```

- To get a concise online listing of command names:

  ```
  % bindkey -l
  ```

Comments on or corrections to this document are welcome. Send them to the address listed above.

Notes

The following terms are used in the command descriptions in the next section:

- The **current word** is the word in which the cursor is located. Many commands affect only the part of the current word immediately to the left of the cursor. (That is, if the cursor is in the middle of a word, such commands affect only that part of the current word.)

  Note that if the character to the left of the cursor is a space, the current word is empty.

- Filename **pattern** characters are `*`, `?`, `[ ]`, and `{ }`.
- The **cut buffer** is used by several commands that delete text (such as *delete-word*). The text can be put back from the cut buffer into the command at the cursor position with the *yank* command.

- The **mark** is a position in the command line that can be set (with *set-mark-command*). Then when you move the cursor, the area between the mark and the cursor is referred to as a **region**, which can be copied or deleted with *copy-region-as-kill* or *kill-region*.
- The **end of file** indicator causes the current shell to terminate if the *ignoreeof* shell variable is not set. If the current shell is the login shell, this logs you out.

Command Editor Command Names

This section lists the names of the command editor commands and describes what they do. When you see a command that you think you'd like to use, you can find out what keys it's bound to by looking in the section "Command Editor Default Bindings" later in this document. If the command name doesn't appear there, you can use *bindkey* to set up your own binding for the command.

backward-char
Move the cursor left one character.

backward-delete-char
Delete the character to the left of the cursor.

backward-delete-word
Delete from the beginning of the current word to the cursor. Deleted text is saved in the cut buffer.

backward-kill-line
Delete from the beginning of the line to the cursor. Deleted text is saved in the cut buffer. See also *kill-line*.

backward-word
Move the cursor to the beginning of the current word, or to the beginning of the previous word if the cursor is already at beginning of the current word. See also *forward-word*. In *vi* mode, words are delimited by space or punctuation; see also *vi-word-back*.

beginning-of-line
In *emacs* mode, move the cursor to the beginning of the line. In *vi* mode, move the cursor to the first non-whitespace character; see also *vi-zero*.

capitalize-word
Convert the character under the cursor to uppercase and move the cursor to the end of the word.

change-case (*vi* only)
Change the case of the character under the cursor and move the cursor right one character.

change-till-end-of-line (*vi* only)
Change text from the cursor position to the end of the line, replacing it with new characters until `ESC` is typed.

clear-screen
Clear the screen, redrawing the current prompt and command line at the top of the screen. This is much quicker than using the *clear* shell command. See also *redisplay*.

complete-word
Perform completion on the current word. Uses programmed completions, if any are applicable. See also *complete-word-raw*.

complete-word-back

Like *complete-word-fwd*, but displays the possible completions in reverse order.

complete-word-fwd
The current word is treated as a completion prefix. Successive *complete-word-fwd* commands successively replace the current word with each of the possible completions. When the list of completions is exhausted, the shell beeps and replaces the current word with the original prefix. You can use *complete-word-fwd* and *complete-word-back* to move back and forth in the list of completions. Uses programmed completions, if any are applicable.

complete-word-raw
Like *complete-word*, but ignores programmed completions.

copy-prev-word
Copy the previous word to the cursor position, including any whitespace between the word and the cursor. See also *insert-last-word*.

copy-region-as-kill
Copy the area between the mark and the cursor to the cut buffer. See also *kill-region* and *set-mark-command*.

delete-char
Delete the character under the cursor.

delete-char-or-eof
Delete the character under the cursor or indicate end of file if the current line is empty.

delete-char-or-list
Delete the character under the cursor or list completions if the cursor is at the end of the line.

delete-char-or-list-or-eof
Delete the character under the cursor, list completions if the cursor is at the end of the line, or indicate end of file if the current line is empty.

delete-word
Delete the characters from the cursor to the end of the current word. Deleted text is saved in the cut buffer.

digit
If a repeat count is currently being collected, add the digit to the end of it. Otherwise, enter the digit into the command line.

digit-argument
Begin collecting repeat count for the following command. Subsequent *digit* comands are added to the repeat count.

down-history
Recall the next history line into the edit buffer. When repeated, continues down through the history list, stopping at the end of the list. See also *up-history*.

downcase-word
Convert characters from the cursor to the end of the current word to lowercase. See also *upcase-word*.

end-of-file
Indicate end of file to the shell.

end-of-line
Move the cursor to the end of the line.

exchange-point-and-mark

Exchange the cursor (point) and mark positions. This is useful if you've forgotten where the mark is. A second *exchange-point-and-mark* command returns the cursor to its original position. See also *set-mark-command*.

expand-glob
Expand the filename pattern to the left of the cursor, replacing it with the matching filenames. See also *list-glob*.

expand-history
Expand history references in the current word. History references beginning with `!#` are not expanded.

expand-line
Expand all history references in a command line, including references beginning with `!#`. See also *magic-space* and *toggle-literal-history*.

expand-variables
Expand variable references in the current word.

forward-char
Move the cursor right one character

forward-word
Move the cursor forward to the end of the current word, or to the end of the next word if the cursor is already at the end of the current word. See also *backward-word*.

gosmacs-transpose-chars
Exchange the two characters the the left of the cursor (like Gosling *emacs*). See also *transpose-chars*.

history-search-backward
Search backward through the history list using the current contents of the edit buffer up to the cursor as a search string (which may be a filename pattern). The command retrieves the previous command beginning with that string. (If the command buffer is empty, all commands match and *history-search-backward* simply retrieves the previous command like *up-history*.) If the first command retrieved is not the one you wanted, repeat *history-search-backward* until you find the right one. If you go too far, *history-search-fwd* searches in the other direction. Identical matches are skipped. *history-search-backward* does not wrap around when the beginning of the history list is reached.

history-search-forward
Like *history-search-backward*, but searches forward through the history list. *history-search-backward* does not wrap around when the end of the history list is reached.

insert-last-word
Insert the final word of the previous command at the cursor position. See also *copy-prev-word*.

i-search-back
Perform an *emacs*-style incremental search. Presents `bck:` as a prompt and waits for you to type a search string. As you type successive characters, the command editor searches back through your history list for commands matching the current search string and successively copies them into the edit buffer. To back up through the commands retrieved, or if you make a typing mistake, delete the last character from the search string and the shell returns the previously retrieved command into the edit buffer. Type `ESC` to terminate the search and leave the current line in the edit buffer. Hit `RETURN` to execute the command.

For each command retrieved, the cursor is positioned at the end of the matching string within the command. Type `CTRL−W` to copy the rest of the word under the cursor to the end of the search string.

If no command matches the search string when you type a new character, the shell beeps. Typing `CTRL−G` returns to the previous successful search. `CTRL−G` aborts the search if the previous character resulted in a successful search.

i-search-fwd
Like *i-search-back*, but searches forward.

keyboard-quit
Clear the entire command line.

kill-line
Delete from the cursor to the end of the line. Deleted text is saved in the cut buffer. See also *backward-kill-line*.

kill-region
Delete the characters between the mark and the cursor. If the region isn't what you thought it was, issue a *yank* command to put the deleted text back. Deleted text is saved in the cut buffer. See also *copy-region-as-kill*, *set-mark-command*, and *exchange-point-and-mark*.

kill-whole-line
Delete the entire line. Deleted text is saved in the cut buffer.

list-choices
List possible completions for the current word. Uses programmed completions, if any are applicable. See also *list-choices-raw*.

list-choices-raw
List possible completions for the current word, ignoring programmed completions. See also *list-choices*.

list-glob
List filename wildcard matches for the current word. See also *expand-glob*.

list-or-eof
List possible completions for the current word or indicate end of file if the line is empty.

load-average
Display the system load average and current process status.

magic-space
Expand history references in the current word, then add a space. (Acts similar to *expand-history*, so references beginning with `!#` are not expanded.) If you bind *magic-space* to `SPACE`, then all history references are expanded as you type successive words of command lines. See also *expand-history*.

newline
Execute the current command (the command in the edit buffer).

normalize-command
Looks for the current word as a command in your search path and replaces it with the command's full pathname. If the word is an alias, the word is replaced by the alias definition. If the word is a builtin command, the word remains unchanged. If the word isn't a command, an alias, or a builtin, the shell beeps.

normalize-path
Expand the current word as a pathname, eliminating leading `.` and `..` components as if the *symlinks* shell variable were set to `expand`.

overwrite-mode (*emacs* only?)
Switch from insert to overwrite mode or vice versa. In insert mode, new characters are inserted into the command line at the cursor position. In overwrite mode, new characters overwrite characters under the cursor. See also *self-insert-command*.

prefix-meta
Turn on the high bit of the next character you type.

quoted-insert
Add the next character you type to the command line literally without interpretation, even if the character is otherwise special. For instance, to enter a literal `TAB` into the command line instead of having it trigger filename completion, precede it with *quoted-insert*.

redisplay
Redisplay the command line. This is useful if another program blats output on the screen and messes it up while you're typing a command. See also *clear-screen*.

run-fg-editor
Look for a stopped editor job and restart it. Editor jobs are identified by looking at the values of the `EDITOR` and `VISUAL` environment variables. If neither is set, jobs beginning with `ed` or `vi` are considered editor jobs.

Restarting an editor is a job control operation you're likely to do often. The default binding is `ESC CTRL−Z`, but you can bind it to `CTRL−Z` to make it easier to use. This way you can restart an editor using the same key you use to stop it.

run-help
Look for documentation on the current command and display it. This is intended for display of short files since the output is not run through a pager. The shell looks for help files in directories named in the `HPATH` environment variable. For a command *xyz*, files named *xyz.help*, *xyz.1*, *xyz.6*, or *xyz.8* are considered help files.

self-insert-command
Add a character as itself to the current command, or replace the character under the cursor if in overwrite mode. See also *overwrite-mode*.

sequence-lead-in
You don't actually bind this command to a key sequence. When you use *bindkey* to display key bindings and a character is indicated as bound to *sequence-lead-in*, it means that one or more multiple-character sequences beginning with the character have been bound to a command.

set-mark-command
Set a mark at the cursor position. Some keyboards may not generate the proper character for the default binding (`CTRL−@`, a null character); in that case you'll need to rebind the command to something else. See also *exchange-point-and-mark*.

spell-line
For each word of the current line, attempts to correct the word as for *spell-word*, except for words beginning with `!`, `.`, `\`, `^`, `−`, or `%`, or containing filename pattern characters.

spell-word
Correct the spelling of the current word. The first word of a line is corrected as a command name and subsequent words are corrected as filenames. If the word appears to be a pathname, correction is attempted for each component of the pathname.

stuff-char
Send the character to the terminal in cooked mode.

toggle-literal-history
Toggle between the literal and lexical representations of the current history line. This affects all commands that retrieve lines from the history list into the edit buffer, such as *up-history*. The literal representation of a line is the line just as you typed it, with history references unexpanded. The lexical

representation of a line has history references expanded and a space between successive words. See also *expand-history*.

transpose-chars
Exchange the character to the left of the cursor with the character under the cursor. See also *gosmacs-transpose-chars*.

transpose-gosling
Same as *gosmacs-transpose-chars*.

tty-dsusp
Terminal delayed-suspend character. Generally the `dsusp` character. See the section "Terminal Control Characters" below.

tty-flush-output
Terminal flush-output character. Generally the `oflush` character. See the section "Terminal Control Characters" below.

tty-sigintr
Terminal interrupt character. Generally the `intr` character. See the section "Terminal Control Characters" below.

tty-sigquit
Terminal quit character. Generally the `quit` character. See the section "Terminal Control Characters" below.

tty-sigtsusp
Terminal suspend character. Generally the `susp` character. See the section "Terminal Control Characters" below.

tty-start-output
Terminal allow-output character. Generally the `start` character. See the section "Terminal Control Characters" below.

tty-stop-output
Terminal disallow-output character. Generally the `stop` character. See the section "Terminal Control Characters" below.

undefined-key
This command indicates that a key is ignored, i.e., the key is unbound. When you type an unbound key, the shell beeps. Normally, you don't actually bind this command to a key, you use *bindkey −r* instead.

universal-argument
emacs universal argument. Repeats the following command four times. If specified twice, repeats the following command 16 times.

up-history
Recall the previous history line into the edit buffer. When repeated, continues up through the history list, stopping at the beginning of the list. See also *down-history*.

upcase-word
Convert characters from the cursor to the end of the current word to uppercase. See also *downcase-word* and *capitalize-word*.

vi-add (*vi* only)
Enter *vi* insert mode, allowing text entry to the right of the cursor.

vi-add-at-eol (*vi* only)
Enter *vi* insert mode, allowing text entry at the end of the line.

vi-beginning-of-next-word (*vi* only)
Move the cursor to the beginning of the next space- or punctuation-delimited word. See also *vi-word-fwd*.

vi-char-back (*vi* only)
Move the cursor backward to the previous instance of the next character you type.

vi-char-fwd (*vi* only)
Move the cursor forward to the next instance of the next character you type.

vi-charto-back (*vi* only)
Move the cursor backward to the right of the previous instance of the next character you type.

vi-charto-fwd (*vi* only)
Move the cursor forward to the left of the next instance of the next character you type.

vi-chg-case (*vi* only)
Change the case of the character under the cursor and move the cursor right one character.

vi-chg-meta (*vi* only)
This command is the prefix in *vi* command mode for change-text commands. When followed by a cursor motion command, enters insert mode so you can change the text from the cursor position to the text where the cursor motion command would place the cursor. For instance, use `cw` to change a word, `c$` to change the rest of the line, and `c0` to change everything from the beginning of the line to the cursor.

vi-chg-to-eol (*vi* only)
Same as *change-till-end-of-line*.

vi-cmd-mode (*vi* only)
Enter *vi* command mode. The bindings from the alternative key map are used in command mode.

vi-cmd-mode-complete (*vi* only)
Like *complete-word* but works in *vi* command mode.

vi-delprev (*vi* only)
Backspace over the previous character in *vi* insert mode. Backspaces only when the cursor is at the end of the characters added since insert mode was entered, and backspaces only to the beginning of those characters.

vi-delmeta (*vi* only)
This command is the prefix in *vi* command mode for delete-text commands. When followed by a cursor motion command, deletes text from the cursor position to where the cursor motion command would place the cursor. For instance, use `dw` to delete a word, `d$` to delete the rest of the line, and `d0` to delete backward to the beginning of the line.

vi-endword (*vi* only)
Move the cursor to the end of the current space-delimited word. See also *vi-eword*.

vi-eword (*vi* only)
Move the cursor to the end of the current space- or punctuation-delimited word. See also *vi-endword*.

vi-insert (*vi* only)
Enter *vi* insert mode, allowing text entry to the left of cursor.

vi-insert-at-bol (*vi* only)
Enter *vi* insert mode, allowing text entry at the beginning of the line.

vi-repeat-char-back (*vi* only)
Repeat the current character search in the opposite search direction. That is, it repeats *vi-char-back* as

vi-char-fwd and *vi-char-fwd* as *vi-char-back*.

vi-repeat-char-fwd (*vi* only)
Repeat the current character search in the same search direction. That is, it repeats *vi-char-back* as *vi-char-back* and *vi-char-fwd* as *vi-char-fwd*.

vi-repeat-search-back (*vi* only)
Like *vi-repeat-search-fwd*, but in the opposite direction.

vi-repeat-search-fwd (*vi* only)
Repeat the current search in the same search direction. If you begin a backward search with *vi-search-back*, then *vi-repeat-search-back* does another backward search. If you begin a forward search with *vi-search-fwd*, then *vi-repeat-search-back* does another forward search. *vi-repeat-search-fwd* and *vi-repeat-search-back* might better be named as *vi-repeat-search* and *vi-repeat-search-reverse* or something like that.

vi-replace-char (*vi* only)
Replace the character under the cursor with the next character you type. Advances the cursor, unlike *vi* itself.

vi-replace-mode (*vi* only)
Enter *vi* character replacement mode, where characters you type replace successive characters in the command line.

vi-search-back (*vi* only)
Search the history list backward. When you use *vi-search-back*, the shell prompts with a `?` character. Type a search string (which may be a filename pattern) and hit `RETURN`. The command retrieves the previous command containing with that string, or beeps if there is no match in the history list. If the command retrieved is not the one you wanted, repeat the search until you find the one you want using *vi-repeat-search-fwd* (no, that's not a typo, see the description of *vi-repeat-search-fwd*). Hit `RETURN` to terminate the search and leave the most recently retrieved command in the edit buffer. Hit `ESC` to execute the currently retrieved command. *vi-search-back* and *vi-repeat-search-back* wrap around when the beginning of the history list is reached.

vi-search-fwd (*vi* only)
Like *vi-search-back*, but prompts with a `/` character and searches forward.

vi-substitute-char (*vi* only)
Enter insert mode to replace the character under the cursor.

vi-substitute-line (*vi* only)
Enter insert mode to replace the entire line.

vi-word-back (*vi* only)
Move the cursor to the previous word. See also *backward-word*.

vi-word-fwd (*vi* only)
Move the cursor to the next word. See also *vi-beginning-of-next-word*.

vi-undo (*vi* only)
Undo the last change. *vi-undo* is unreliable, unfortunately.

vi-zero (*vi* only)
Move the cursor to the beginning of the line. See also *beginning-of-line*, which in *vi* mode moves the cursor to the first non-whitespace character.

which-command
Runs *which* for the first word of the current line.

yank

Yank the contents of the cut buffer into the command line at the cursor position. The cut buffer contents remain unchanged.

Terminal Control Commands

The terminal control commands (*tty-dsusp*, *tty-flush-output*, *tty-sigintr*, *tty-sigquit*, *tty-sigtsusp*, *tty-start-output*, and *tty-stop-output*) are by default bound to characters that are commonly used for the corresponding terminal driver control functions. For instance, `CTRL−C` is often used for the terminal driver `intr` function, so `CTRL−C` is bound to *tty-sigintr*. If you want to use different characters for terminal control functions, you should make the change using both *stty* and *bindkey* so that the terminal driver and the command editor both know about the change. For instance, if you want to use `CTRL−T` as the `intr` character, issue both these commands:

`stty intr ^t`	(in *~/.login*)
`bindkey ^t tty-sigintr`	(in *~/.cshrc*)

Command Editor Default Bindings

This section lists the default bindings for both *emacs* and *vi* editing modes. For *vi* mode, bindings are listed separately for insert mode and command mode. In *vi* command mode, bindings from the alternate key map are used.

`CTRL−`*X* means to hold down the `CTRL` (control) key as you type *X*. `META−`*X* means to hold down the `META` key as you type *X*. `ESC` *X* means to type `ESC`, then *X* (two characters).

In general, when `META−`*X* is bound to a given command, `ESC` *X* is bound to the same command, for keyboards that have no `META` key.

The character names in the first column below are used in the binding lists that follow. Each name is equivalent to the corresponding control character in the right column.

TAB	CTRL−I
RETURN	CTRL−M
LINEFEED	CTRL−J
BACKSPACE	CTRL−H

Most single character sequences not shown in the lists below are bound to *self-insert-char*. However, if you type a character and it does nothing, then most likely it's unbound (i.e., bound to *undefined-key*).

Some commands are bound to multiple key sequences. For such commands, the sequences are shown separated by commas.

Default Bindings—emacs mode

To repeat a command *n* times in *emacs* mode, precede it with `ESC` *n*, e.g., `ESC 3 ESC d` to delete three words.

Command	**Default Key Sequence(s)**
backward-char	CTRL-B, LEFT-ARROW
backward-delete-char	BACKSPACE, DEL
backward-delete-word	ESC BACKSPACE, ESC DEL, META-BACKSPACE, META-DEL
backward-word	ESC B, ESC b, META-B, META-b
beginning-of-line	CTRL-A
capitalize-word	ESC C, ESC c, META-C, META-c
clear-screen	CTRL-L, ESC CTRL-L, META-CTRL-L
complete-word	TAB, ESC TAB, ESC ESC, META-TAB, META-ESC
complete-word-raw	CTRL-X TAB
copy-prev-word	ESC CTRL-_, META-_
copy-region-as-kill	ESC W, ESC w, META-W, META-w
dabbrev-expand	ESC /, META-/
delete-char-or-list-or-eof	CTRL-D
delete-word	ESC D, ESC d, META-D, META-d
digit	0 through 9
digit-argument	ESC 0 through ESC 9, META-0 through META-9
down-history	CTRL-N, DOWN-ARROW
downcase-word	ESC L, ESC l, META-L, META-l
end-of-line	CTRL-E
exchange-point-and-mark	CTRL-X CTRL-X
expand-glob	CTRL-X *
expand-history	ESC SPACE, META-SPACE, ESC-!, META-!
expand-variables	CTRL-X $
forward-char	CTRL-F, RIGHT-ARROW
forward-word	ESC F, ESC f, META-F, META-f
history-search-backward	ESC P, ESC p, META-P, META-p
history-search-forward	ESC N, ESC n, META-N, META-n
insert-last-word	ESC _, META-_
kill-line	CTRL-K
kill-region	CTRL-W
kill-whole-line	CTRL-U
list-choices	ESC CTRL-D, META-CTRL-D
list-choices-raw	CTRL-X CTRL-D
list-glob	CTRL-X G, CTRL-X g
newline	LINEFEED, RETURN
normalize-command	CTRL-X ?
normalize-path	CTRL-X N, CTRL-X n
quoted-insert	CTRL-V
redisplay	CTRL-R
run-fg-editor	ESC CTRL-Z, META-CTRL-Z
run-help	ESC H, ESC h, META-H, META-h
set-mark-command	CTRL-@
spell-line	ESC $, META-$
spell-word	ESC S, ESC s, META-S, META-s
toggle-literal-history	ESC R, ESC r, META-R, META-r
transpose-chars	CTRL-T
tty-dsusp	CTRL-]
tty-flush-output	CTRL-O
tty-sigintr	CTRL-C

Command	Default Key Sequence(s)
tty-sigquit	`CTRL-\`
tty-sigtsusp	`CTRL-Z`
tty-start-output	`CTRL-Q`
tty-stop-output	`CTRL-S`
up-history	`CTRL-P, UP-ARROW`
upcase-word	`ESC U, ESC u, META-U, META-u`
which-command	`ESC ?, META-?`
yank	`CTRL-Y`

Default Bindings—vi mode

The arrow keys allow movement up or down in the history list, or back and forth in the current line, but be aware that if your arrow keys send out key sequences beginning with `ESC` (as is typical), you will be in command mode after using them.

Insert mode bindings

Command	Default Key Sequence(s)
backward-char	`CTRL-B`
backward-delete-char	`BACKSPACE, DEL`
backward-delete-word	`CTRL-W`
backward-kill-line	`CTRL-U`
beginning-of-line	`CTRL-A`
clear-screen	`CTRL-L`
complete-word	`TAB`
down-history	`CTRL-N, DOWN-ARROW`
end-of-line	`CTRL-E`
expand-line	`CTRL-X`
kill-line	`CTRL-K`
list-glob	`CTRL-G`
list-or-eof	`CTRL-D`
newline	`LINEFEED, RETURN`
quoted-insert	`CTRL-V`
redisplay	`CTRL-R`
run-help	`META-?`
transpose-chars	`CTRL-T`
tty-dsusp	`CTRL-Y`
tty-flush-output	`CTRL-O`
tty-sigintr	`CTRL-C`
tty-sigquit	`CTRL-\`
tty-sigtsusp	`CTRL-Z`
tty-start-output	`CTRL-Q`
tty-stop-output	`CTRL-S`
up-history	`CTRL-P, UP-ARROW`
vi-cmd-mode	`ESC`

Command mode bindings

You cannot repeat a command in *vi* insert mode. In *vi* command mode, many of the commands may be repeated by typing the repeat count before the command, e.g., `3dw` to delete three words.

Command	Default Key Sequence(s)
backward-char	`BACKSPACE, LEFT-ARROW, h`
backward-delete-char	`DEL, X`
backward-delete-word	`CTRL-W`
backward-kill-line	`CTRL-U`
backward-word	`b`
beginning-of-line	`CTRL-A, ^`
change-case	`~`
change-till-end-of-line	`C`
clear-screen	`CTRL-L`
delete-char-or-eof	`x`
digit-argument	`1` through `9`
down-history	`CTRL-N, DOWN-ARROW, +, j`
end-of-line	`CTRL-E, $`
expand-glob	`*`
expand-history	`!`
expand-line	`CTRL-X`
expand-variables	`V, v`
forward-char	`CTRL-F, RIGHT-ARROW, SPACE, l`
history-search-backward	`K`
history-search-forward	`J`
kill-line	`CTRL-K, D`
list-choices	`CTRL-D`
list-glob	`CTRL-G`
newline	`LINEFEED, RETURN`
redisplay	`CTRL-R`
run-help	`ESC ?`
tty-flush-output	`CTRL-O`
tty-sigintr	`CTRL-C`
tty-sigquit	`CTRL-\`
tty-start-output	`CTRL-Q`
tty-stop-output	`CTRL-S`
up-history	`CTRL-P, UP-ARROW, -, k`
vi-add	`a`
vi-add-at-eol	`A`
vi-beginning-of-next-word	`w`
vi-char-back	`F`
vi-char-fwd	`f`
vi-charto-back	`T`
vi-charto-fwd	`t`
vi-chg-meta	`c`
vi-cmd-mode-complete	`TAB`
vi-delmeta	`d`
vi-endword	`E`

Command	**Default Key Sequence(s)**
vi-eword	`e`
vi-insert	`i`
vi-insert-at-bol	`I`
vi-repeat-char-back	`,`
vi-repeat-char-fwd	`;`
vi-repeat-search-back	`N`
vi-repeat-search-fwd	`n`
vi-replace-char	`r`
vi-replace-mode	`R`
vi-search-back	`?`
vi-search-fwd	`/`
vi-substitute-char	`s`
vi-substitute-line	`S`
vi-undo	`u`
vi-word-back	`B`
vi-word-fwd	`W`
vi-zero	`0`