

Bash Binds

The completion keybinds use mnemonic symbols for each category. `@` for hostname, `/` pathsep for files, `!` bang for commands. Known hosts won't complete if they are hashed, which they probably should be. Most of Bash syntax has similar mnemonics in the syntax for keys & string expansions, but it sounds a bit weird explaining it.

Emacs Bindings

abort <code>C-g, C-x C-g, M-C-g</code>	<u>digit</u> └── argument <code>M--</code> └── argument <code>M-0</code> └── argument <code>M-1</code> └── argument <code>M-2</code> └── argument <code>M-3</code> └── argument <code>M-4</code> └── argument <code>M-5</code> └── argument <code>M-6</code> └── argument <code>M-7</code> └── argument <code>M-8</code> └── argument <code>M-9</code>	<u>insert</u> └── comment <code>M-#</code> └── completions <code>M-*</code> └── last-argument <code>M-., M-_</code>	spell-correct-word <code>C-x s</code> start-kbd-macro <code>C-x (</code> tilde-expand <code>M-&</code>
accept-line <code>C-j, C-m</code>			
<u>backward</u> └── char <code>C-b, <left></code> └── delete <code>C-h</code> └── kill └── line <code>C-x C-?</code> └── word <code>M-C-h</code> └── word <code>M-b, M-<left></code>		<u>kill</u> └── line <code>C-k</code> └── word <code>C-<delete>, M-d</code>	<u>transpose</u> └── chars <code>C-t</code> └── words <code>M-t</code>
<u>beginning-of</u> └── history <code>M-<, <pgup></code> └── line <code>C-a, <home></code>		<u>next</u> └── history <code>C-n, M-<down></code>	undo <code>C-, C-x C-u</code>
call-last-kbd-macro <code>C-x e</code> capitalize-word <code>M-c</code>	display-shell-version <code>C-x C-v</code> downcase-word <code>M-l</code> dynamic-complete-history <code>M-C-i</code> edit-and-execute-command <code>C-x C-e</code>	<u>non-incremental</u> └── forward-search-history <code>M-n</code> └── reverse-search-history <code>M-p</code>	<u>unix</u> └── line-discard <code>C-u</code> └── word-rubout <code>C-w</code>
<u>character-search</u> <code>C-]</code> └── backward <code>M-C-] \$char</code>	<u>end</u> └── kbd-macro <code>C-x)</code> └── of-history <code>M->, history M-<pgdown></code> └── line <code>C-e, line <end></code>	<u>operate-and-get-next</u> <code>C-o</code>	uppercase-word <code>M-u</code>
<u>clear</u> └── display <code>M-C-l</code> └── screen <code>C-l</code>	exchange-point-and-mark <code>C-x C-x</code> execute-named-command <code>M-x</code>	<u>possible</u> └── command-completions <code>C-x !</code> └── completions <code>M-, M-?</code> └── filename-completions <code>C-x *</code> └── hostname-completions <code>C-x @</code> └── username-completions <code>C-x ~</code> └── variable-completions <code>C-x \$</code>	<u>yank</u> <code>C-y</code> └── last-arg <code>M-., M-_</code> └── nth-arg <code>M-C-y</code> └── pop <code>M-y</code>
<u>complete</u> <code>C-i, <esc> <esc></code> └── command <code>M-!</code> └── filename <code>M-</code> └── hostname <code>M-@</code> └── into-braces <code>M-{</code> └── username <code>M-~</code> └── variable <code>M-\$</code>	<u>forward</u> └── char <code>C-f, M-<right></code> └── search-history <code>C-s</code> └── word <code>M-f</code> └── word <code>M-<right></code>	<u>previous</u> └── history <code>C-p, M-<up></code>	
<u>delete</u> └── char <code>C-d, <delete></code> └── horizontal-space <code>M-\</code>	<u>glob</u> └── complete-word <code>M-g</code> └── expand-word <code>C-x*</code> └── list-expansions <code>C-xg</code>	quoted-insert <code>C-q, C-v, <insert></code> re-read-init-file <code>C-x C-r</code> reverse-search-history <code>C-r</code> revert-line <code>M-C-r, M-r</code> set-mark <code>C-@, M-"</code>	
	history-expand-line <code>M-^</code>	<u>shell</u> └── backward-word <code>M-C-b</code> └── expand-line <code>M-C-e</code> └── forward-word <code>M-C-f</code> └── kill-word <code>M-C-d</code> └── transpose-words <code>M-C-t</code>	

Self-Insert

Hitting `C v` or `<insert>` runs `quoted-insert $key` which inserts the escape sequence for `$key` into the terminal. Running `quoted-insert` on these keys in sequence enters the following into the terminal. Successive `C v` keybinds seem to be handled recursively.

	000	001	010	011	100	101	110	111
	S	M	M S	C	C S	C M	C M S	
<code><insert></code>	<code>^[[2-</code>		<code>^[[2;3~</code>	<code>^[[2;4~</code>	<code>^[[2;5~</code>	<code>^[[2;6~</code>	<code>^[[2;7~</code>	<code>^[[2;8~</code>
<code>C v</code>	<code>^v</code>		<code>^[v</code>					

The terminal application will intercept `C v` before it's given to the window manager (something like this), so you need to use `C S v` to paste.

In `xkb`, the `shift` modkey causes alphanumeric keybinds to be interpreted as their raw character codes. `C S v` literally inserts `C` `v` and `AltGr` ; into `¶`. Thus, I removed `self-insert` & `do-lowercase-version` binds from this reference.

Escape Sequences

F1	<code>^[[0P</code>	F5	<code>^[[15~</code>	F9	<code>^[[20~</code>	home	<code>^[[H</code>	del	<code>^[[3~</code>
F2	<code>^[[0Q</code>	F6	<code>^[[17~</code>	F10	<code>^[[21~</code>	end	<code>^[[F</code>	bksp	<code>^?</code>
F3	<code>^[[0R</code>	F7	<code>^[[18~</code>	F11	<code>^[[23~</code>	pgup	<code>^[[5~</code>	bktab	<code>^[[Z</code>
F4	<code>^[[0S</code>	F8	<code>^[[19~</code>	F12	<code>^[[24~</code>	pgdn	<code>^[[6~</code>	esc	<code>^[[</code>

Quality of Life Keybinds

```
abort C-g
capitalize-word M-c
character-search C-] $char
└── backward M-C-] $char

complete
├── command M-!
├── filename M-\*
├── hostname M-@
├── into-braces M-{
├── username M-~
└── variable M-$

digit-argument M-[0-9-]
downcase-word M-l
dynamic-complete-history M-C-i
edit-and-execute-command C-x C-e
exchange-point-and-mark C-x C-x

glob
├── complete-word M-g
├── expand-word C-x
└── list-expansions C-x g

insert
├── completions <esc>-*
└── last-argument M-., M_-

print-last-kbd-macro
quoted-insert C-v, <insert>
undo C _, C-x C-u
upcase-word M-u

yank
└── nth-arg M-C-y
└── pop M-y
```

Avoid RSI: ensure your keyboard has symmetric modkeys for **ambidextrous usage** and *to disrupt muscle memory*. Do not use the control in the lower corners. $\tau = (\text{rxF})$

Keybind Combos

To avoid the strange unix-word-rubout

Keybinds involving **C-w** require setting:

```
bind '"\C-@":set-mark'
bind '"\e ":"set-mark'
bind '"\C-w":kill-region'
bind '"\ew":copy-region-as-kill'
```

From middle of file or url path C-@ M-C] <space> C-x C-x C-w

Jump to last space, cut region, leaving the remainder of the path.

Then **C-a varname=value; C-x C-x <right>**

From midline C-@ C-] | C-] | C-x C-x C-w

Set mark with **C-@**, search forward the next two **|** chars, then **exchange-point-and-mark** with **C-x C-x**.

This highlights the region. Then **C-w** to cut the text with **kill-region**.

From midline =C-k C-a C-y

Swap order of commands

From midline C-k C-a C-] | C-] | ... C-y

Cut to end of line, search through line for **|** char and insert the tail of a bash pipeline. Simply repeating **C-]** doesn't work.

From midline C-a export VAR=value;

Prefix command with environment var

From midline C-k # C-y

Kill to end of line, insert **#** and paste. **C-y** to **yank**. Commented command content is still in history.

Running print-last-keyboard-macro ...

Shows you what's recorded. You can make composite functions.

Unmapped by default

```
alias-expand-line      dump
arrow-key-prefix      functions
backward              macros
└── byte               variables

bash-vi-complete      emacs-editing-mode
copy                  export-completions
└── backward-word     fetch-history
└── forward-word      forward
└── region-as-kill    backward-delete-char
                     └── byte

dabbrev-expand        history
delete                and-alternate-expand-line
└── char-or-list      search-backward
                     search-forward
                     substring
                     └── search-backward
                     └── search-forward

kill                  region
└── whole-line
magic-space           menu
                     └── complete
                     └── complete-backward

non-incremental       next
                     └── screen-line

old-menu-complete     non-incremental
                     └── forward-search-history-again
                     └── reverse-search-history-again
overwrite-mode        previous
                     └── screen-line

print-last-kbd-macro shell
redraw-current-line  └── backward-kill-word

skip-csi-sequence    skip-csi-sequence
tab-insert            tab-insert
tty-status             tty-status
universal-argument   unix
                     └── filename-rubout
```

Vi Bindings

For vi bindings ... you're on your own.

vi	vi	vi	vi
append	char-search	forward	redo
eol	column	bigword	replace
mode	complete	word	rubout
arg-digit	delete-to	fword	search-again
back-to-indent	edit-and-execute-command	fWord	set-mark
backward	editing-mode	goto-mark	subst
bigword	end	insert-beg	tilde-expand
word	bigword	insertion-mode	undo
bword	word	match	unix-word-rubout
bWord	eof-maybe	movement-mode	yank
change	eword	next-word	arg
case	eWord	overstrike-delete	pop
char	fetch-history	prev-word	to
to	first-print	put	