

<code>\n</code>	toggle relative number
<code>\d</code>	dispatch and make current file
<code>\w</code>	toggle word wrap
<code>F2</code>	YcmCompleter GoTo
<code>F3</code>	YcmCompleter GoToDeclaration
<code>F4</code>	YcmCompleter GoToDefinition
<code>F5</code>	open nerd tree
<code>F8</code>	toggle ctag bar

tcoment	
<code>gc {motion}</code>	Toggle inline comments
<code>gc[Count]c {motion}</code>	Toggle comment with count and motion
<code>gcc</code>	Toggle comment for the current line
<code>g< {motion}</code>	Uncomment region
<code>g</code>	Uncomment the current line
<code>g</code>	Uncomment the current region as block
<code>g> {motion}</code>	Comment region
<code>g>c</code>	Comment the current line
<code>g>b</code>	Comment the current region as block
<code>gc</code>	Toggle comments (visual mode)
<code>g></code>	Comment selected text (visual mode)

C Tags	
<code>C-]</code>	jump to tag
<code>C-t</code>	ctags jump back from tag
<code>gd</code>	goto local declaration
<code>gD</code>	goto global declaration
<code>g]</code>	list matching tags
<code>gC-]</code>	jump to tag if defined once otherwise list matching tags
<code>:tags</code>	list tag stack
<code>\ut</code>	update ctags

Cscope	
<code>C-\ s</code>	show uses of symbol in quickfix

Find	
<code>\gf (Normal)</code>	(global find) start empty Ag for cpp
<code>\gf (Visual)</code>	(global find) Ag with selection for cpp
<code>\f (Normal)</code>	(find)start empty Ag in current file
<code>\f (Visual)</code>	(find) Ag with selection in current file

Replace	
<code>rx</code>	replace all chars in selection with x
<code>:{range}sort u</code>	sort and remove dups
<code>:{range}sort</code>	sort
<code>:{range}sort!</code>	sort reverse
<code>:{range}sort n</code>	numeric sort
<code>:s/foo/bar/g</code>	Change each 'foo' to 'bar' in the current line.
<code>:%s/foo/bar/gc</code>	Change each 'foo' to 'bar' in all the lines., but ask for confirmation first.
<code>:{range}s/foo/bar/gc</code>	Change each 'foo' to 'bar' in all lines within a visual selection, but ask for confirmation first.
<code>:g/pattern/d</code>	remove every line matching pattern

```
!ctags -R --c++-kinds=+p --fields=+iaSl --extra=+q --links=yes . update tag file
!find . -iname *.c -iname *.cc -iname *.hpp -iname *.h -iname *.cpp >.cscopelist.tmp; cscope -q -R -b -i .cscopelist.tmp; rm .cscopelist.tmp update csope db
```

Multiple cursors

<code>C-n</code> in Normal	highlights the current word under the cursor in Visual mode/finds next
<code>C-n</code> in Visual	virtual cursor at every line and leaves you in Normal mode.
<code>C-p</code> in Visual	remove the current virtual cursor and go back to the previous virtual cursor location.
<code>C-x</code> in Visual	remove the current virtual cursor and skip to the next virtual cursor location.
<code>:</code>	
<code>{r}</code> MultipleCursorsFind	regex

Ctrlp

<code>C-p</code>	CtrlP file
<code>\b</code>	CtrlP buffer
<code>\m</code>	CtrlP mixed
<code>\r</code>	CtrlP most recently used
<code>\t</code>	CtrlP buffer tags
<code>F5</code>	(in CtrlP) purge the cache
<code>C-f/C-b</code>	(in CtrlP) cycle between modes
<code>C-j/C-k</code>	(in CtrlP) up/down
<code>C-d</code>	(in CtrlP) switch to filename only search
<code>C-r</code>	(in CtrlP) switch to regexp mode

Format

<code>\fx</code>	format XML
<code>\fj</code>	format JSON
<code>:%!xmlint --format -</code>	format xml in whole file
<code>:{range}!xmlint --format -</code>	format xml in selection
<code>:%!python -m json.tool</code>	format json in selection

<code>v</code>	enter Visual Mode
<code>C-v</code>	enter Visual Mode
<code>C-v</code>	in insert mode adds tab char
<code>:retab</code>	fix tabs to spaces
<code>:so ~/.vimrc</code>	Reload vimrc
<code>:! wc %</code>	run external command on current file
<code>:noh</code>	turn off search highlight
<code>vim -d file1 file2</code>	diff 2 files
<code>%</code>	in {range} will use whole file
<code>:set pastetoggle</code>	
<code>:set paste</code>	turn off paste formatting
<code>:set nopaste</code>	

tmux

<code>C-a [</code>	enter text mode, hjkl to move, /? to search, C-ud to page
<code>C-a :</code>	enter command mode
<code>C-a c</code>	create window
<code>C-a .</code>	rename window
<code>C-a -</code>	split horizontal
<code>C-a </code>	split vertical
<code>C-a 1..9</code>	jump to window

Buffer

<code>C-w ghjk</code>	move to the left/bottom/top/right window
<code>C-w W</code>	move to next window
<code>:sp</code>	horizontal split
<code>:vsp</code>	vertical split
<code>:b</code>	open buffer with filename
<code>:b</code>	open buffer with number
<code>:bd</code>	close current buffer
<code>:bn</code>	next buffer
<code>:bp</code>	prev buffer
<code>C-6</code>	last buffer
<code>:enew</code>	new buffer, with filename creates
<code>:e</code>	reload buffer, with filename creates
<code>:bufdo e</code>	reload buffer all buffers, runs foreach buffer

A

<code>\a</code>	switch to the header/cpp file
<code>\as</code>	splits and switches
<code>\av</code>	vertical splits and switches
<code>\at</code>	new tab and switches
<code>\an</code>	cycles through next match

Easy Motion

<code>\\w</code>	easy motion word forward
<code>\\b</code>	easy motion word back
<code>\\k</code>	easy motion up line
<code>\\j</code>	easy motion down line

Errors

<code>[q</code>	:cprevious prev error
<code>]q</code>	:cnext next error
<code>[Q</code>	:cfirst first line
<code>]Q</code>	:clast last line
<code>:copen</code>	open Quickfix