15-131 : Great Practical Ideas for Computer Scientists

VIM cheat sheet

Allison McKnight (aemcknig@andrew.cmu.edu)

Navigation

`h`	Move left	`H`	Top line on screen
`j`	Move down	`M`	Middle line on screen
`k`	Move up	`L`	Bottom line on screen
`l`	Move right		
`10j`	Move down 10 lines		

`gg`	First line of the file	`e`	The end of the current word
`G`	Last line of the file	`b`	Beginning of current word
`:20`	Line 20 of the file	`w`	Beginning of next word

`0`	Beginning of current line
`^`	First non-whitespace character of current line
`$`	End of current line
`%`	Move to matching parenthesis, bracket or brace

The left, right, up and down arrow keys can also be used to navigate.

Editing

`i`	Insert before current character
`a`	Insert after current character
`I`	Insert at the first non-whitespace character of the line
`o`	Insert a line below the current line, then enter insert mode
`O`	Insert a line above the current line, then enter insert mode
`r`	Overwrite one character and return to command mode
`U`	Undo
`Ctrl+R`	Redo

Opening, closing and saving files

`:w`	Save the current file
`:wq`	Save the current file and close it; exits vim if no open files remain
`:w newname`	Save a copy of the current file as 'newname,' but continue editing the original file
`:sav newname`	Save a copy of the current file as 'newname' and continue editing the file 'newname'
`:q!`	Close a file without saving
`:e somefile`	Opens `file` in the current buffer
`:x`	Write any changes to the file and close the file

Mode switching

`i`	Enter insert mode
`:`	Enter command mode
`R`	Enter replace mode
`v`	Enter visual mode (highlighting)
`V`	Enter visual line mode (highlighting lines)
`esc`	Return to normal mode from insert or replace mode
`esc+esc`	Return to normal mode from command or visual mode

Copy/pasting

Within vim

`y`	Yank
`c`	'Change'; cut and enter insert mode
`C`	Change the rest of the current line
`d`	Delete; cut but remain in normal mode
`D`	Delete the rest of the current line
`p`	Paste after the cursor
`P`	Paste before the cursor
`x`	Delete characters after the cursor
`X`	Delete characters before the cursor

Copy/paste commands operate on the specified range. If in visual mode, that range is the highlighted text. If in normal mode, that range is specified by a series of modifiers to the commands:

`cw`	Change one word
`c4w`	Change four words
`c4l`	Change four letters
`cc`	Change current line
`4x`	Change four characters after the cursor
`4p`	Paste five times after the cursor.

Modifiers work similarly for cut, delete, yank and paste.

From system clipboard

`:set paste`	Enter paste mode
`:set nopaste`	Exit paste mode
`Ctrl+Shift+V`	Paste into file, if in paste mode; `Command+Shift+V` for Mac

Replace

:s/foo/bar/	Replace the first occurrence of foo on the current line with bar
:[range]s/foo/bar/[flags]	Replace foo with bar in range according to flags

Ranges

%	The entire file
'<,'>	The current selection; the default range while in visual mode
25	Line 25
25,50	Lines 25-50
$	Last line; can be combined with other lines as in '50,$'
.	Current line; can be combined with other lines as in '.,50'
,+2	The current lines and the two lines therebelow
-2,	The current line and the two lines thereabove

Flags

g	Replace all occurrences on the specified line(s)
i	Ignore case
c	Confirm each substitution

Regular expressions

Both vim's find and replace functions accept regular expressions. By default, vim assumes that some characters are part of a regular expression and these characters must be escaped with '\' to be searched for literally; these characters include (,), *, ., ^, $. Other regular expression patterns are interpreted literally by default and must be escaped to be used as a part of a regular expression; these include +.

Search

*	Find the next instance of the current word
#	Find the previous instance of the current word
n	Find the next instance of the word being searched for, in the direction specified by the last use of {*,#}
N	Find the previous instance of the word being searched for, in the direction specified by the last use of {*,#}
/word	Find the next occurrence of 'word'
/word\c	Find the next occurrence of 'word', ignoring case ('\c' can appear anywhere in the sequence being searched for)
/\<word\>	Find the next occurrence of the word 'word', where 'word' is bounded by word boundaries (ex. space, dash)
:noh	Un-highlight words

Handy tricks

~	Toggle case of character beneath the cursor
xp	Transpose current character with that to its right
J	Join the current line and the line beneath it
:Ni!	Receive inquiry about shrubberies
:help	View vim help file
:help foo	View vim help entry on 'foo'

Resources (other than google.com)

http://www.arl.wustl.edu/~fredk/Courses/Docs/vim/
The vim manual, online.

http://www.stackoverflow.com/
Good general-purpose programming questions site.

http://vim.wikia.com/
A wiki dedicated to vim.