Vim Tutorial

Note that all commands presented here must be used from command mode, which can be easily accessed be pressing the ⟨ESC⟩ key.

Navigation Motions

Navigation forms the basis of all Vim commands. Not only do you use these commands to navigate a text file, you can also chain them with other commands to perform them over different ranges of words.

`h l k j` - Move the cursor one position left, right, up, and down, respectively.

`w b` - Move the cursor forward and backward respectively to the *beginning* of a word.

`e ge` - Move the cursor forward and backward respectively to the *end* of a word.

`( )` - Move the cursor forward and backward respectively to the *beginning* of a sentence.

`{ }` - Move the cursor forward and backward respectively to the *beginning* of a paragraph.

`^ $` - Move the cursor to the first or last character of a line.

`:`*n* - Jump to beginning of line *n*

`gg G` - Jump to beginning or end of file

`%` - Jump to corresponding parenthesis/curly bracket/square bracket of one under cursor

Editing Text - Enter Insert/Replace Modes

There are many ways to modify text in Vim, and many commands for changing text incorporate the functionality of different commands. These commands allow you to enter Insert and Replace modes after executing them. Many of these can be combined with a number before them to repeat it that many times (i.e. `10~` will change the case of 10 letters).

`i a` - Insert before, append after cursor

`I A` - Insert at beginning, append at end of line

`o O` - Open new line below, above current line, and insert

`r`*c* - Replace current character with *c*

`R` - Start replacing characters from cursor (Enter Replace Mode)

`c`*m* - Change text in motion *m* (See Navigation Motions)

`cc S` - Change entire current line (Both do the same thing)

`C` - Change from current cursor until the end of current line

`~` - Change case under cursor and move forward

`<< >>` - Indent current line to left or right

Editing Text - Outside of Insert/Replace Modes

In addition to changing text from inside of Insert and Replace modes, you can also edit text from outside. These commands are useful for restructuring text and moving large chunks of text around in a file, if need be. Similarly to the previous commands, these can be combined with numbers before them, to apply the command to several characters or lines.

`y`*`m`* - Copy text from current cursor toward motion *m*

`yy Y` - Copy current line (Both do the same thing)

`d`*`m`* - Cut text from current cursor toward motion *m*

`dd D` - Cut current line (Both do the same thing)

`p P` - Paste contents after or before, respectively, cursor position

`J` - Join the current line with the next, leaving no space between

`.` - Repeat last change in text

Searching

Searching in Vim is useful for finding instances of a word you may want to change, or to check the function signature when you can't remember exactly what parameters it takes.

`/`*`string`* - Search for *string*, moving cursor to next instance and highlighting it in the window

`n N` - Move to the next or previous instance of the last searched string

Folds

Folds are an advanced but excellent utility in Vim, allowing you to condense your code so that it is immensely more readable while you are working on it.

`zf`*`m`* - Create a fold toward motion *m*

`zo zc` - Open or close a fold, respectively (Can by anywhere in range of fold)

`zd zE` - Delete current fold or delete all folds

Other Useful Commands

Here are some miscellaneous commands that are useful for every developer.

`gg=G` - Note that `gg` and `G` are both motions. The command `=`*m* performs auto-indentation toward motion *m*. Running the command `gg=G` will jump to the beginning of the file and auto-indent until the end of the file.

`K` - Note that `K` is capitalized. This command will search the Unix manual pages for the word under the cursor. This is excellent for looking up functions and function signatures. You can exit this by typing `q`.

`gf` - This opens the file under the cursor. It saves and quits the current file and will replace it with the newly opened file. This is useful if, for example, you finish editing an implementation file and need to go back to the header file.