vim Tutorial

vim(1) is a text editor that can be used from a terminal. *vim* is a very powerful editor and has many useful functions. In this tutorial I will go over two of *vim*'s modes, namely insert mode and command mode. *vim* also has other modes but for now we will focus on those two.

Getting Started with *vim*

In order to open a file using *vim* all you need to do is type the following in a terminal window:

`vim filename`

where *filename* is the name of the file you want to edit. You can also just type `vim`. Note that if you open *vim* this way, it will open a default *vim* file. In order to save your work to a specific file, you will have to specify the filename when you save it. This tutorial will go over saving files later.

As mentioned above, *vim* has many modes, but we will focus on two of them: insert mode and command mode.

- **Command mode**: In command mode, you can type commands to navigate around the file, delete lines, move lines around, and do other useful editing functions.
- **Insert mode**: In insert mode you can modify the file. This mode allows you to type into the file.

Useful *vim* Commands

I put together a short list of useful *vim* commands. To use these commands, you have to be in command mode.

- Editing commands
  - `h` – move cursor left
  - `j` – move cursor down
  - `k` – move cursor up
  - `l` – move cursor right
  - `a` – enter insert mode and add what you type after the character under the cursor; hit the ESCAPE key to leave insert mode.
  - `i` – enter insert mode and add what you type before the character under the cursor; hit the ESCAPE key to leave insert mode.
  - `x` – delete character under the cursor
  - `:set number` – shows line numbers on the side of the screen
  - `:set nonumber` – hide line numbers
  - `yy` – yank (copy) current line
  - `p` – paste a line one line below where cursor is
  - `P` – paste a line one line above where cursor is
  - `dd` – delete line where the cursor is
  - `dw` – delete the word under the cursor
  - `:u` – undo the last change made
- Range-based editing commands
  vim has some commands that can be applied to multiple lines (a *range*). A range is specified by `start_line,end_line`. "`$`" means the last line in the file; "`0`" means the beginning of the file.
  - `:0,4y` – copy all lines between lines 0 and 4 inclusive
  - `:0,$d` – delete all lines in the file
- Saving and exiting commands
  - `:w` – saves a file (won't work if you didn't open *vim* as `vim filename`)
  - `:w filename` – saves the file to *filename*
  - `:q` – exit *vim*
  - `:q!` – exit *vim*, discarding unsaved changes
  - `:wq` – save changes and then exit *vim*
  - `ZZ` – save changes and then exit *vim* (same as `:wq`)
- Navigation commands
  - *n*`G` – jump to line *n* of the file. For example, `27G` takes you to line 27. Note that `0G` takes you to the end of the file.
  - `L` – moves the cursor to the end of the window.
  - `M` – moves the cursor to the middle of the window.

More Information

The CSIF machine have a command that will bring up a longer tutorial; just run

```
vimtutor
```

Another quick and simple tutorial can be found at `http://heather.cs.ucdavis.edu/~matloff/vim.html`

Credit

This was written for ECS 30, Programming and Problem Solving, in Fall 2015 by Jonathan Vronsky, and modified slightly by Matt Bishop.