#### Vi

While `nano` is great for simple edits, `vi` and `emacs` have more advanced and powerful features. There is a learning curve to using these editors as they are not exactly intuitive. It will require a bit of a time investment to become proficient. Let's start by looking at `vi`.

`vi [file]` - Edit file.

`vim [file]` - Same as `vi`, but with more features.

`view [file]` - Starts `vim` in read-only mode. Use `view` when you want to examine a file but not make any changes.

`Vim` stands for "Vi IMproved." It is compatible with the commands found in `vi`. Some of the additional features of `vim` include syntax highlighting, the ability to edit files over the network, multi-level undo/redo, and screen splitting. On many Linux distributions when you invoke `vi`, you are actually running `vim`.

One advantage of knowing `vi` is that `vi` or a `vi` variant like `vim` is always available on the system. Another advantage is that once you learn the key mappings for `vi` you can apply them to other commands like `man`, `more`, `less`, `view`, and even your shell.

#### `vi` Modes

##### Command Mode

`Vi` has the concept of modes. You are always working in one of three modes: command mode, insert mode, or line mode. When `vi` starts you are placed into command mode. To get back to command mode at any time hit the escape key (`Esc`). Letters typed while in command mode are not sent to the file, but are rather interpreted as commands. Command mode allows you to navigate about the file, perform searches, delete text, copy text, and paste text.

Here are some commonly used key bindings for navigation.

`k` - Up one line.

`j` - Down one line.

`h` - Left one character.

`l` - Right one character.

`w` - Right one word.

`b` - Left one word.

`^` - Go to the beginning of the line.

`$` - Go to the end of the line.

Note that commands are case sensitive. For example, if you want to move down one line type the lowercase `j`. The uppercase `J` joins lines together. The original `vi` editor did not employ the use of arrow keys, However `vim` does, so you may find that you can use arrow keys on your system. The advantages of learning the original key bindings are 1) they always work and 2) it's faster since your hand does not have to leave the home row.


**Insert mode**

To enter insert mode, press one of the following keys.

`i` - Insert at the cursor position.

`I` - Insert at the beginning of the line.

`a` - Append after the cursor position.

`A` - Append at the end of the line.

After entering into insert mode, type the desired text. When you are finished, type `Esc` to return to command mode.

**Line mode**

To enter line mode you must start from command mode and then type a colon (`:`) character. If you are in insert mode, type `Esc` to get back to command mode and then type a colon for line mode. Here are some of the most common line mode commands you will want to know.

`:w` - Writes (saves) the file.

`:w!` - Forces the file to be saved even if the write permission is not set. This only works on files you own.

`:q` - Quit. This will only works if there have not been any modifications to the file.

`:q!` - Quit without saving changes made to the file.

`:wq!` - Write and quit. After modifying a file this command ensures it gets saved and closes `vi`.

`:x` - Same as :wq.

`:n` - Positions the cursor at line `n`. For example, :5 will place the cursor on the fifth line in the file.

`:$` - Positions the cursor on the last line of the file.

`:set nu` - Turn on line numbering.

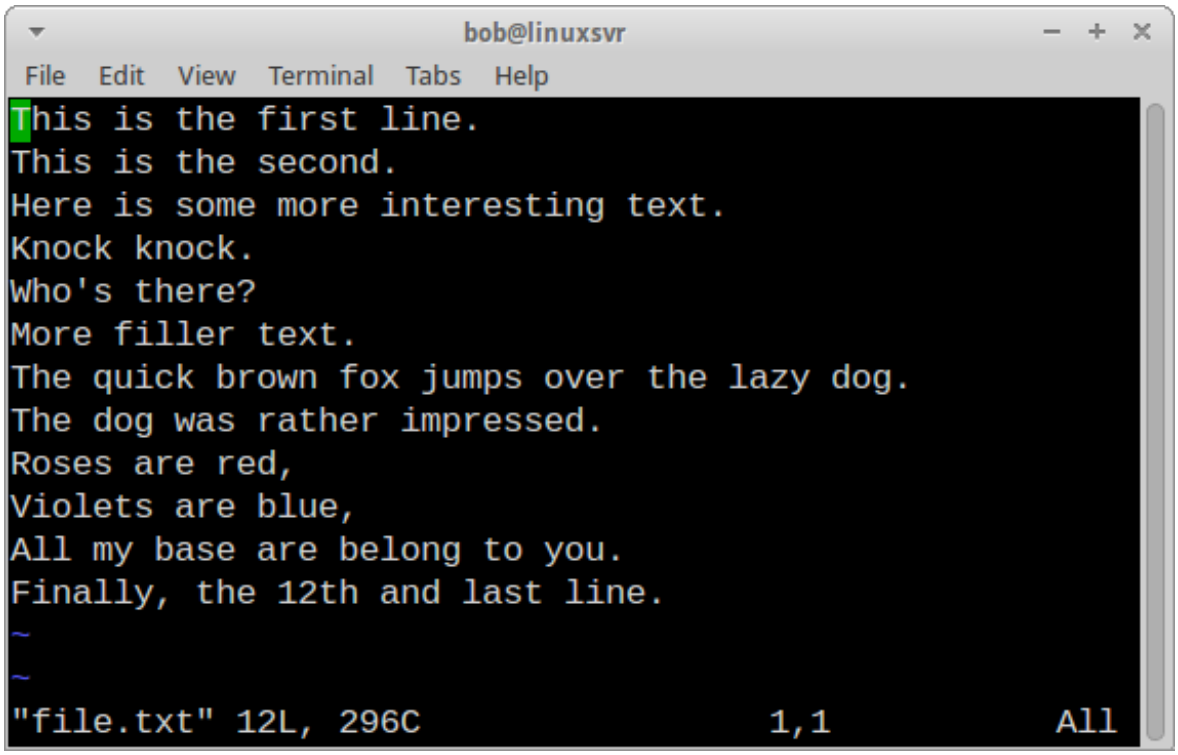
`:set nonu` - Turn off line numbering.

`:help [subcommand]` - Get help. If you want more information on the `:w` command type `:help :w`.

| Mode | Key | Description |
|---|---|---|
| Command | `Esc` | Used to navigate, search, and copy/paste text. |
| Insert | `i I a A` | Also called text mode. Allows text to be inserted in the file. |
| Line | `:` | Also called command-line mode. Save the file, quit `vi`, replace text, and |

perform some navigation.

Here is a screenshot of `vim`. Tildes (`~`) represent lines beyond the end of the file.




#### Advanced Editing with `vi`

You can repeat commands in `vi` by preceding them with a number. For instance, if you would like to move the cursor up 5 lines type `5k`. If you would like to insert a piece of text 80 times, type `80i` and start entering the text. Once you hit `Esc` to return to command mode the text you typed will be repeated 80 times. If you would like to make a line of asterisks, you could type `80i*Esc`. Can you start to see how `vi` is more powerful than an editor like `nano`?

#### Deleting Text

`x` - Delete a character.

`dw` - Delete a word. To delete five words, type `d5w`. The repeating concept in `vi` shows up in many places.

`dd` - Delete a line. To delete 3 lines, type `3dd`.

`D` - Delete from the current position to the end of the line.

#### Changing Text

`r` - Replace the current character.

`cw` - Change the current word.

`cc` - Change the current line.

`c$` - Change the text from the current position to the end of the line.

`C` - Same as `c$`.

`~` - Reverses the case of a character.

#### Copying and Pasting

`yy` - Yank (copy) the current line.

`y<position>` - Yank the <position>. For example, to yank a word type `yw`. To yank three words type `y3w`.

`p` - Paste the most recent deleted or yanked text.

#### Undo / Redo

`u` - Undo.

`Ctrl-r` - Redo.

#### Searching

`/<pattern>` - Start a forward search for <pattern>.

`?<pattern>` - Start a reverse search for <pattern>.

http://www.LinuxTrainingAcademy.com