Editing Text With Vim

Hassan Shabbir

February 13, 2019

Contents

1 The main text editors

There are two main (extensible) text editors for Unix and GNU/Linux[1] operating systems. These are the VI and Emacs lines of text editors.

Both of these text editors are highly configurable and therefore are highly recommended by many programmers, so much so that since their inception many programmers have been in a "war", arguing over which text editor is better.[2]

1.1 VI

VI (pronounced "ve-eye"), and other text editors based off of it are great at creating and editing text. It is (usually) a basic text editor, however newer versions of it allow a user to customize it and get IDE-like features within it.[3] The main feature of VI is the way in which it allows a user to edit text.

1.2 GNU Emacs

GNU[4] Emacs (pronounced "g'noo ee-maks") is a text editor that is extremly configurable, and allows for a lot more IDE-like features. This is due to the fact that Emacs is a text editor written and extensible in the general purpose functional programming language Lisp. Emacs can also emulate VI keybindings using EVIL mode (EVIL stands for Extensible VI Layer). It is also interesting to note that Richard Stallman, the creator of GNU Emacs, also created the first alternative to Unix, which was GNU, the Free Software Foundation, as well as the GNU General Public Licence.[5]

From here on out we will be talking about the VI line of editors. The curious reader is encouraged to read more about Emacs, if interested.

[1] I'd just like to interject for a moment. What you usually refer to as Linux, is in fact, GNU/Linux, or as I've recently taken to calling it, GNU plus Linux. Linux is not an operating system unto itself, but rather another free component of a fully functioning GNU system made useful by the GNU corelibs, shell utilities and vital system components comprising a full OS as defined by POSIX. (See GNU Linux copy pasta.)

[2] See The Editor Wars.

[3] This is usually not encouraged, expecially at the beginning, since having plugins hinder your ability to understand Vim, and are nice to have and not necessarily mandatory for the functioning of Vim.

[4] GNU stands for "GNU's Not Unix", a recursive acronym.

[5] Richard Stallman.

2 Which Vim version should I use?

NOTE: There is almost no difference between these different text editors (for our purposes) except for the information written below.

NOTE: I will assume that Vim/NeoVim is used for the rest of this file, but will refer to it as Vim, since I am lazy.

There are three different VI versions you should be familiar with.

2.1 VI

VI[6] is the oldest text editor of the three, which was created in 1976. VI is short for Visual, differentiating it from line editors.[7] If you plan to work with many servers, you should expect literally expect every server to have it. It is very minimalistic so it won't tell you when you are in insert mode in any way (see below), for example. This makes it harder to understand for beginners, and doesn't have all the features of Vim.

2.2 Vim

Vim[8] is the newer version of VI, first released in 1991. Due to it being very popular for a long time, and still is (last stable release: 17 May 2018), it has many resources available on the internet for it. It is also recently modern, so it shouldn't be too difficult to use. Therefore, whenever searching for general vim resources use vim in the search terms.

2.3 NeoVim

NeoVim[9] is the newest version of vim, first released in November 1, 2015. This has the most interactive features and therefore is the one that I would recommend to new users. For example, when you switch between different modes, the cursor changes, helping you to remember which mode you are in.

---

[6] VI pronounced "ve-eye", also pronounced "vy" but that is an unofficial pronounciation

[7] Technically, the "ed" and "ex" editors are even older, but they literally show you nothing (you have to manually print lines), so it is like editing text in the dark. This is where the command mode in VI comes from, see below. Also see Actually Using Ed for some extreme masochism.

[8] Vim pronounced "vim"

[9] NeoVim pronounced "neo-vim".

3 Installing Vim/NeoVim

Packages (programs) can easily be installed in any Linux distrobution using the command line. For example, to manage packages on Ubuntu Linux, use the package manager `apt`. The full command to install Vim would then be `sudo apt install vim`, and the command to install NeoVim would be `sudo apt install nvim`.[10]

4 Modal editing with Vim

The first thing you need to know about Vim is that there are four main modes in which you operate. Each of these modes change what the keys on your keyboard will do.

In general, when editing text you will mostly be making small changes, and very rarely do you create whole documents without mistakes from start to finish (`cat > foo.txt` anyone?). For this reason, Vim is optimized for modifying text. Understanding modal editing (along with composability, repeatability, and text objects) is the key to understanding Vim[11].

4.1 Normal Mode (Movement; Modification)

Normal mode allows you to issue commands to Vim, which will then do something, other than putting the key that you pressed into the file. When you first open a file you will be in Normal Mode. Pressing keys such as `j` or `l` will move the cursor rather than adding those letters to the file. You should almost always be in Normal Mode, since editing text requires the ability of moving around the document, and deleting, replacing, copying text, etc. which are all possible in Normal mode. Thus this mode is called "normal" since it is the default mode when using Vim. To get back into Normal Mode either use the escape key, or the key combination control and left-squrebracket keys.

4.2 Insert Mode (Add Text)

When opening a document with Vim, you will be in Normal Mode. To get into Insert Mode, for example, you can press keys such as `i` or `a` and then you will be in Insert Mode. If you are using NeoVim, you will see the cursor

[10] The command to both run and install it is `nvim` NOT `neovim`.

[11] For more on how vim works see this awesome answer on Stackoverflow Your problem with Vim is that you don't grok vi.

become thin, and in both Vim and NeoVim you will see `--INSERT--` at the bottom of the terminal.[12] You can then use the arrow keys to get to the location, and press the keys to add them to the document. To get back to Normal Mode press escape. (This is not recommended, but can help you get used to Vim. Movement should be done using Normal Mode, not the arrow keys, allowing your hand to stay on home row) You will notice that the cursor will become a block again in NeoVim.

4.3 Command Mode (System Commands; Ed commands)

For now, the most important command mode you need to know will be the commands to exit Vim (which is accessible from Command Mode). This is such a problem for Vim beginners that this stackoverflow answer has 4,000 upvotes, and over 1 Million views. The first step is to press `:` from Normal Mode. If you are not in Normal Mode, get to it using the escape key. You will now see a colon on the last line of the terminal. If you wish to save your changes type `wq`, and then press enter. This command stands for write (save) the file then quit Vim. If you wish to throw away your changes type `q!` instead and then press enter.

4.4 Visual Mode (Select Text)

Visual Mode is used for performing an operation over all of the characters in the text. This can be useful when you don't know how to run operations using text objects. Text objects allow you to refer to regions of text, such as "in braces", "in tag", "all paragraph", etc. Text objects will replace most simple uses of Visual Mode.

5 Vim editing commands

NOTE: The 'Beginner' subheading will let you know which parts to focus on as a beginner. Only learn the Beginner commands that you want. Then when you get annoyed by inefficiency you can come back to learn more.

NOTE: Pressing the Escape key will return you back to Normal Mode from any mode.

NOTE: Vim uses mnemonic devices (ie. `d` stands for delete) to help you remember what command does what. Use this to remember what each command does. Also, commands that are related, but do something different

[12] In VI you will neither see the cursor change nor the `--INSERT--` at the bottom

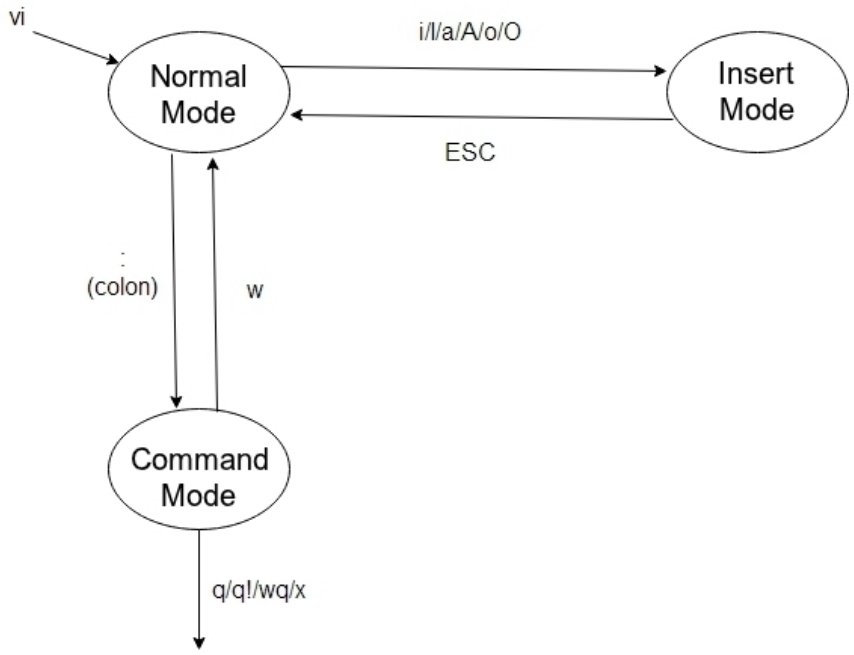

Figure 1: General overview of Vim Modes. Will be covered in depth later.

are capitalized (`D` deletes to the end of the line), and the default action is defined by the repeated letter (such as `dd` for delete with default action, delete line).

Sections will be in the form: CommandName (from StartingMode)
Commands will be in the form:

- `COMMAND`: (mnemonic device) Description of command

5.1 Entering NeoVim (from bash prompt)

You can enter NeoVim from the commandline (not to be confused with Vim's Command Mode) by typing `nvim file.txt`, replacing `file.txt` for the file you want to edit. If the file doesn't exist, it will be created. You will now be in NeoVim.

If you wish to use Vim, replace `nvim` in the command above with `vim`.

5.2 Movement Commands (from Normal Mode)

5.2.1 Character Movement

1. Beginner

   - `h`: Move cursor left
   - `j`: Move cursor down
   - `k`: Move cursor up
   - `l`: Move cursor right

   The way to remember this is that the `h` key is on the left of the four keys, `l` is on the right, `j` is written with the hook below the line, and `k` has the vertical line above the line.

   Character movement can also be prefixed with a number such as `5l`, to go 5 characters right.

5.2.2 Line Movement

1. Beginner

   - `^`: (This is from Regexes[13]) Go to start of line
   - `$`: (This is from Regexes) Go to end of line

[13] Regexes, or regular expressions, are a way of doing things like parsing and substituting in a file. The regex `^hi` says match the line starting with `hi`, and the regex `^$` says match the empty line (ie. the line that starts and ends with nothing in between).

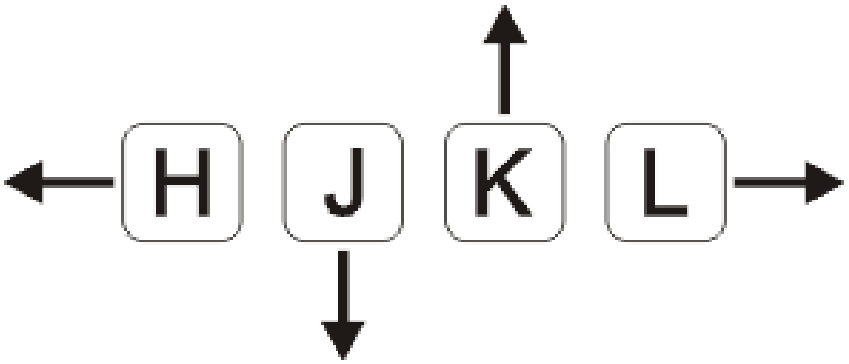

Figure 2: A graphical depiction of h, j, k, l

5.2.3 File Movement

1. Beginner
    - `gg`: Go to start of file
    - `G`: Go to end of file
2. Intermediate
    - `50gg`: Go to line 50

5.2.4 Word Movement

1. Intermediate Frankly, I used to just spam `h` and `l` for quite a while, so these commands aren't strictly necessary.
    - `w`: (Word) Go forward by one word
    - `b`: (Back) Go back by one word
    - `e`: (End) Go to the next end of word

5.2.5 Find Char Movement

1. Beginner
    - `fx`: (Find Char) Find character 'x' forwards
    - `;`: Run `f` / `F` again
2. Intermediate

- `Fx`: (Find Char) Find character 'x' backwards
- `,`: Run `f` / `F` again in opposite direction
- `tx`: ('Til/Until) Go up until character 'x', forwards
- `Tx`: ('Til/Until) Go up until character 'x', backwards

5.2.6 Search Term Movement

1. Beginner
    - `/`: Input search term, then press enter
    - `n`: (Next) Go to next location matching search term
2. Intermediate
    - `N`: (Previous/Backwards Next) Go to previous location matching search term

5.3 Insert Commands (from Normal Mode)

These commands will change you automatically from Normal Mode to Insert Mode.

1. Beginner
    - `i`: (Insert) Enter Insert Mode before current character
    - `I`: (Insert) Enter Insert Mode at the beginning of the line
    - `a`: (Append) Enter Insert Mode after current character
    - `A`: (Append) Enter Insert Mode at the end of the line
2. Intermediate
    - `o`: (Open) Enter Insert Mode at the end of the line
    - `O`: (Open) Enter Insert Mode at the end of the line

5.4 Deletion Commands (from Normal Mode)

NOTE: The composable nature of Vim should be apparent in this section.

1. Beginner
    - `x`: Delete character under cursor

- `dd`: (Delete, Default) Delete current line
- `dw`: (Delete Word) Delete until the end of the word
- `dfc`: (Delete Find 'c') Delete including the first 'c' on the right of the cursor
- `diw`: (Delete In Word) Delete the whole word
- `diW`: (Delete In Word) Delete the whole space delimited word

2. Intermediate I can't really be bothered to count how many words I want to delete. I prefer doing things like `dw..` instead, see below.

    - `d3w`: (Delete Word) Delete 3 number of words, etc.

5.5 Deletion Commands (from Visual Mode)

1. Beginner

    - `d`: (Delete) Delete current visual selection
    - `x`: (Delete) Delete current visual selection

5.6 Change Commands (from Normal Mode)

Change deletes something then puts you in Insert Mode to add text.

1. Beginner

    - `cc`: (Change, Default) Delete line, then go into Insert Mode
    - `cw`: (Change Word) Delete until the end of the word, then go into Insert Mode
    - `ciw`: (Change In Word) Delete the whole word, then go into Insert Mode
    - `ciW`: (Change In Word) Delete the whole space delimited word, then go into Insert Mode

2. Intermediate I can't really be bothered to count how many words I want to change. I prefer doing things like `cw..` instead, see below.

    - `c3w`: (Change Word) Delete 3 number of words, etc., then go into Insert Mode

5.7 Yank (Copy) Commands (from Normal Mode)

NOTE: To copy text to use in other applications, use the `"+` prefix, which may not work in VI/Vim, also see registers below.

1. Beginner

    - `yy`: (Yank, Default) Yank (copy) the current line, for Vim use only
    - `yiw`: (Yank) Yank (copy) the current line, for Vim use only
    - `"+yy`: (Yank, Default) Yank (copy) the current line, for any application
    - `"+yiw`: (Yank) Yank (copy) the current line, for any application

5.8 Yank (Copy) Commands (from Visual Mode)

1. Beginner

    - `y`: (Yank) Yank (copy) current visual selection

5.9 Paste Commands (from Normal Mode)

1. Beginner

    - `p`: (Paste) Paste the last deletion/yank

5.10 Paste Commands (from Visual Mode)

1. Beginner

    - `p`: (Paste) Paste, replacing current visual selection

5.11 Undo Command (from Normal Mode)

1. Beginner

    - `u`: (Undo) undo last change

5.12 Visual Mode Commands (from Normal Mode)

First enter Visual Mode using any of the below, then make the selection using the movement commands as you would from Normal Mode. Then run the command on the selection, such as yank, delete, etc.

1. Beginner
    - `v`: (Visual) Enter character-wise Visual Mode
    - `V`: (Visual) Enter line-wise Visual Mode
2. Intermediate
    - `ctrl-v`: (Visual) Enter block-wise Visual Mode

    NOTE: To comment out lines, use block-wise selection with `ctrl-v`, then press `I`, and type the character comment (`//` for example), and hit escape. It can also be used as a poor man's version of a macro (see below). Another way would be to use the Vim Commentary plugin (see below), with the command `gc`.

5.13 Command Mode (from Normal Mode)

1. Beginner
    - `:w`: (Write) Write the file
    - `:q`: (Quit) Quit Vim, without having modified the file
    - `:q!`: (Quit!) Quit Vim, throwing away modifications
    - `:wq`: (Write-Quit) Write the file, then quit Vim
    - `:x`: (Exit) Shorthand for `:wq`
2. Intermediate
    - `:! date`: (`!` is similar to `|`) Run bash command `date` and show the result without adding to file
    - `:s/foo/bar/g`: (Substitute) Substitute 'foo' with 'bar', globally (ie. each occurrence)
    - `:r! date`: Run bash command `date` and read in the result into the file

5.14 Command Mode (from Visual Mode)

Visually select text then enter Command Mode using `:`. NOTE: you will see `:'<,'>` instead. This just tells Vim to run the command over the whole selection.[14]

1. Intermediate
    - `:'<,'>! wc -l`: Run bash command `wc -l` on visually selected text

6 Composability and repeatability

6.1 Text Objects

NOTE: All text objects can be used with delete, yank, copy, etc. "In" deletes the text inside, while "All" deletes quotes, braces, and a single space (so the spaces around it end up balanced).

1. Beginner
    - `iw`: (In Word)
    - `aw`: (All Word)
    - `is`: (In Sentence)
    - `as`: (All Sentence)
    - `ip`: (In Paragraph)
    - `ap`: (All Paragraph)
    - `i"`: (In Quote)
    - `a"`: (All Quote)
    - `i}`: (In Brace)
    - `a}`: (All Brace)
    - `it`: (In Tag) Used in HTML
    - `at`: (All Tag) Used in HTML

[14] So the command will run in the range `x,y`, and a `'a` refers to the mark a, with the `<` referring to the first and `>` referring to the last selection. So all together it says "run the command from the beginning of the selection, to the end of the selection."

6.2 Dot (.) command

1. Beginner The dot command repeats the last complete command that you ran. For example if you changed a word to "Hi" using `ciwHi` and then escape, you can change another word to "Hi" using dot.

   This is one way of renaming variables.[15] First search for a variable using `/`, then using `ciw` change the variable to something else. Finally, repeat this change all throughout the document using `n` to go to the next instance, and `.` to apply the change.

6.3 Number Prefixes

1. Intermediate Most commands can be prefixed, meaning you can run commands like `d5w` which will delete the next 5 words.

6.4 Macros

1. Intermediate Macros can be used for creating groups of repeatable commands. In other words, start macro, run general commands (ie. `w` rather than `llllllll`), stop macro, run the macro previously defined on all of the remaining text. The steps are:

   (a) `qa`: Record Macro in register `a`, see below

   (b) `q`: While recording, it will end the macro

   (c) `@a`: Run Macro in register `a`

   Fun fact: you can also define recursive macros, (works in NeoVim).[16] This allows you to create a single macro that runs forever (of course, Vim will stop the macro at the end of the document, for example). An example of this is the following key sequences: `qaV~j@a`.

[15] The other way would be to run a search and replace, such as `:s/foo/bar/g`, which would replace all occurences of `foo` with `bar`.

[16] Here's a quick introduction to recursion. Recursion is defining something in terms of what you are defining. For example, a directory can contain multiple files and multiple directories. A math example would be an equation like `x = 1 + x`, and replacing x on the right with `1 + x` giving us `x = 1 + 1 + x`, and continuing to infinity would give us `x = 1 + 1 + 1 + 1 + ....` A similar expansion can be carried out on the acronym `GNU`, left as an exercise for the reader.

7 Registers

1. Intermediate The most important part about registers is that the `"+` prefix is used to store the global clipboard, which can be accessed by any program.[17] Other actions, such as yanks and deletions can be prefixed with a register, for later retrieval.

   A useful combination is using registers for editing a macro you wrote. To continue from the macros section, you can write an incorrect macro,

   Frankly, I don't use any register other than the global one.

8 Extending Vim for yourself

To change the default behaviour of Vim, you can modify a configuration file called `.vimrc` (in Gnu/Linux) or `_vimrc` (in Windows, I think).

This will allow you to use plugins, change colorshemes, map keys to commands, etc.

8.1 Plugins

These are a few plugins that I would consider quite useful.

- Vim Plug: Vim plugin manager

To be able to use the below plugins you need to install a plugin manager, this is the one I personally use.

- Vim Sensible: set default settings for Vim

This is useful for starting off in Vim. (Not needed for NeoVim.)

- Numbers Vim: add relative line numbers to Vim (great for going n lines up or down)
- Vim Commentary: (un)comment lines of code with a text object
- Vim Surround: surround text objects with text
- Vim Vinegar: simple file browser in Vim

[17] The `"` is used to retrieve registers, with `+` referring to the name of the register to be accessed.

- Emmet Vim: create HTML easily
- Ctrlp Vim: fuzzy find files
- Targets Vim: add more text objects to Vim

More plugins for Vim can be found on `https://vimawesome.com`.

8.1.1 ColorSchemes

- Space Vim Dark
- Solarized

8.1.2 Vim in other places

- Bash/Zsh: Both Bash and Zsh have Vim modes that can be enabled
- Athame: Full Vim in the terminal, ie. when writing bash commands
- Vimium: Vim in Chrome

There are also other applications that will use Vim-like keybindings by default, such as `man`.

9 Conclusion

Congratulations on finishing this whole document! You should now know enough to be able to use vim, and look up whatever you need on the internet. To become proficient with Vim, you should use it repeatedly, until the Beginner commands come to you without much thought.