Vim en:Introduction

What is Vim?

Vim is a computer program used for writing any kind of text, whether it is your shopping list, a book, or software code.

What makes Vim special is that it is both **simple and powerful**.

Simple means it is easy to get started with. Simple means that it has a minimalistic interface that helps you to concentrate on your main task – writing. Simple means it is built around few core concepts that help you learn deeper functionality easily.

Powerful means getting things done more quickly, better, and more easily. Powerful means making not-so-simple things possible. Powerful does not mean complicated. Powerful means following the paradigm of **"Minimal effort. Maximal effect."**

What can Vim do?

I can hear you say, "So it's a text editor. What's the big deal anyway?"

Well, a lot.

Let's see some random examples to compare Vim with your current editor. The point of this exercise is for you to answer the question *"How would I do this in the editor I currently use?"* for each example.

Note: Don't worry too much about the details of the Vim commands here, the point here is to enlighten you with the possibilities, not to start explaining how these things work. That is what the rest of the book is for.

1. How do you move the cursor down by 7 lines?

- Press `7j`

2. How do you delete a word? Yes, a "word".

- Press `dw`

3. How do you search the file for the current word that the cursor is currently placed on?

- Press `*`

4. How to do a find and replace only in lines 50-100?

- Run `:50,100s/old/new/g`.

5. What if you wanted to view two different parts of the same file simultaneously?

- Run `:sp` to 'split' the view

6. What if you wanted to open a file whose name is in the current document and the cursor is placed on that name?

- Press `gf` (which means 'g'o to this 'f'ile)

7. What if you wanted to choose a better color scheme for the display?

- Run `:colorscheme desert` to choose the 'desert' color scheme (my favorite)

8. What if you wanted to map the keyboard shortcut `ctrl-s` to save the file?

- Run `:nmap :w`. Note that " means a 'c'arriage 'r'eturn, i.e., the enter key.

9. What if you wanted to save the current set of open files as well as any settings you have changed, so that you can continue editing later?

- Run `:mksession ~/latest_session.vim`, and open Vim next time as `vim -S ~/latest_session.vim`.

10. What if you wanted to see colors for different parts of your code?

- Run `:syntax on`. If it doesn't recognize the language properly, use `:set filetype=Wikipedia`, for example.

11. What if you wanted different parts of your file to be folded so that you can concentrate on only one part at a time?

- Run `:set foldmethod=indent` assuming your file is properly indented. There are other methods of folding as well.

12. What if you wanted to open multiple files in tabs?

- Use `:tabedit` to open multiple files in "tabs" (just like browser tabs), and use `gt` to switch between the tabs.

13. You use some words frequently in your document and wish there was a way that it could be quickly filled in the next time you use the same word?

- While in insert mode, press `ctrl-x ctrl-n` to see the list of "completions" for the current word, based on all the words that you have used in the current document. Switch between possibilities with `ctrl-n`ext and `ctrl-p`revious Alternatively, use `:ab mas Maslow's hierarchy of needs` to expand the abbreviation automatically when you type `m a s`.

14. You have some data where only the first 10 characters in each line are useful and the rest is no longer useful for you. How do you get only that data?

- Press `ctrl-v`, select the text and press `y` to copy the selected rows and columns of text.

15. What if you received a document from someone which is in all caps, find it irritating and want to convert it to lower case?

- In Vim, run the following:

```
:for i in range(0, line('$'))
:    call setline(i, tolower(getline(i)))
:endfor
```

  Don't worry, other details will be explored in later chapters. A more succinct way of doing this would be to run `:%s#(.)#1l1#g` but then again, it's easier to think of the above way of doing it.

  There is an even simpler method of selecting all the text (`ggVG`) and using the `u` operator to convert to lowercase, but then again that's too easy, and isn't as cool as showing off the above way of making Vim do steps of actions.

Phew. Are you convinced yet?

In these examples, you can see the power of Vim in action. Any other editor would make it insanely hard to achieve the same level of functionality. And yet, amazingly, all this power is made as understandable as possible.

Notice that we didn't use the mouse even once during these examples! This is a good thing. Count how many times you shift your hand between the keyboard and the mouse in a single day, and you'll realize why it is good to avoid it when possible.

Don't be overwhelmed by the features here. The best part of Vim is that you don't need to know all of these features to be productive with it, you just need to know a few basic concepts. After learning those basic concepts, all the other features can be easily learned when you need them.
