Vim

Introduction

Vim was published in november 2nd, 1991 as an implementation of Vi (1976) by Bram Moolenar.

VIM is everywhere

For starters, Vim is installed pretty much everywhere. I can't remember ever logging into a system and not being able to use the Vim editor. Most Linux distributions install it by default. With other editors such as nano or Emacs, they may or may not be available. If you're working on a system and your only editor choice is Vim, then you need to at least be able to use it to make simple edits.

Also, many programs rely on an external editor. By default, in most cases, that external editor is Vim. Typically this can be overridden by setting an environment variable such as EDITOR. However, if you execute one of these commands and find yourself looking at the Vim editor, you'll be happy to at least know how to make simple edits and exit out of Vim. Just a couple of commands that rely on an external editor such as Vim include crontab, visudo, and git.

How learning VIM improves your productivity

Typing speed is irrelevant up to a certain level. Your ability to navigate through code is much more important. This is where Vim along with its keybindings, layout and set up can help you speed up the process.
The skills that you build while learning Vim can be used in many different editors, such as VSCode (with the Vim plugin) or Sublime Text and so on…

Vim modes

Normal Mode

By default, Vim starts in “normal” mode. Normal mode can be accessed from other modes by pressing Esc.
Normal mode is where one should spend most of their time while using Vim because in this mode you can navigate in the text with these commands and change blocks of code with other commands.
In normal mode, there are multiple ways to move around in an open file. In addition to using the cursor keys to move around, you can use movement commands as well.
But in the normal mode you can’t insert text with the keyboard because you need to enter in the “Insert Mode”.

Insert Mode

This is the second most used mode, and will be the most familiar behavior to most people. Once in insert mode, typing inserts characters just like a regular text editor. You can enter it by using an insert command from normal mode.
To leave insert mode and return to normal mode, press Esc.

Visual Mode

Visual mode is used to make selections of text, similar to how clicking and dragging with a mouse behaves but without a mouse. Selecting text allows commands to apply only to the selection, such as copying, deleting, replacing, and so on.
In Visual Mode you can use normal mode commands to edit text and other visual mode specific commands.
Press ‘v' to enter visual mode, this will also mark a starting selection point..
Move the cursor to the desired end selection point; vim will provide a visual highlight of the text selection.
Visual mode also has the following variants:
V to enter visual line mode, this will make text selections by line
<Ctrl+V> to enter visual block mode, this will make text selections by blocks.

Command Mode

Command mode has a wide variety of commands and can do things that normal mode can’t do as easily. To enter command mode type ’:’ from normal mode and then type your command which should appear at the bottom of the window. For example, to do a global find and replace type :%s/foo/bar/g to replace all ‘foo’ with ‘bar’.“”

Replace Mode

Replace mode allows you to replace existing text by directly typing over it. Before entering this mode, get into normal mode and put your cursor on top of the first character that you want to replace. Then press ‘R’ (capital R) to enter replace mode. Now whatever you type will replace the existing text. The cursor automatically moves to the next character just like in insert mode. The only difference is that every character you type will replace the existing one.

Commands

Movement Commands

`j` =move down;
`k` = move up;
`l` = move right;
`h` = move left;
`<N>j`;
You can also use the above command with ‘k’, ‘l’, ‘h’.
You can also move with the arrow keys.
`gg` = move to the first line of the file;
`G` = move to the last line of the file;
`:<N>`;
`H` = top line on screen;
`M` = middle line on screen;
`L` = bottom line on screen;
`e` = the end of the current word;
`b` = beginning of current word;
`w` = beginning of next word;
`0` = beginning of current line;
`^` = first non-whitespace character of current line;
`$` = end of current line;
`%` = move to matching parenthesis, bracket or brace;

Utility Commands

`I` = insert at the first non-whitespaced character of the line;
`a` = insert after current character;
`o` = insert a line under the current one and enters insert mode;
`O` = insert a line above the current one and enters insert mode;
`r` = overwrite the current character and enters command mode;
`U` = undo;

`Ctrl+R` = redo;

Commands for opening, closing and saving files

`:w` = save the current file;
`:wq` = save the current file and close it;
`:w newname` = save a copy of the current file as newname, but continue editing the original file;
`:sav newname` = save a copy of the current file as newname, but continue editing the file newname;
`:q!` = close a file without saving;

Changing mode Commands

`esc+esc` = return to normal mode from command or visual mode
`esc = return to normal mode from insert or replace mode`
`i` = insert before current character;
`:` = enters command mode;
`R` = enters replace mode;
`v` = enters visual mode all along the line;
`V` = enters visual mode from the cursor;

Commands for copy and pasting

`y` = copy the entirely line;
`c` = cut and enters insert mode;
`C` = cut the rest of the current line and enters insert mode;
`d` = cut but remains normal mode;
`D` = cut the rest of the current line;
`p` = paste in the line after the cursor;
`P` = paste in the line before the cursor;
`x` = delete a character after the cursor;
`X` = delete a character before the cursor;

How to create and use macros

1. Enter command mode and press ‘q’
2. Press any key you want the macro to be assigned to
3. Press any key you want the macro to reproduce
4. Press ‘q’ and the key you assigned the macro to, to stop the macro
5. Press ‘@’ and ‘shift+<macro-key>’ to start the macro

What's a .vimrc file? (https://github.com/amix/vimrc)

Vim allows for really awesome customization, typically stored inside a .vimrc file. Typical features for a programmer would be syntax highlighting, smart indenting and so on.
In this file you can also save your macros.
This configuration file is located under every user home directory like. To get there you simply type ~/.vimrc in the terminal.

Basic version (https://github.com/amix/vimrc/blob/master/vimrcs/basic.vim)

This version of the .vimrc file already has basic plugins stored in it.

Awesome version (https://github.com/amix/vimrc)

This version of the file .vimrc includes a ton of useful plugins, color schemes, and configurations.

Example of plugins(https://github.com/amix/vimrc) (https://github.com/vgod/vimrc)

- vim-commentary is a plugin that allows you to comment stuff out.
- vim-indent-objects is a plugin that allows you to define a new text object representing lines of code at the same indent level. Useful for python/vim scripts.
- vim-indent-guides is a plugin for visually displaying indent levels in Vim.
- AutoClose is a plugin that inserts matching brackets, paren, brace or quotes.
- YankRing is a plugin that maintains a history of previous yanks, changes and deletes.
- ack.vim is a plugin that runs ack (a better grep) from vim, and shows the results in a split window.