Hackers Toolbox: Vim

Taufiq Mohammed, Toh Li Heng

06 Feb 2024

Slides at `https://hckr.cc/ht2324s2-w4-slides`

Where are we?

NUS Hackers

http://nushackers.org

Hackers Toolbox

Hack & Roll

Friday **Hacks**

Hackerschool

About Me

Hi, I'm Taufiq!

Year 2 CS undergrad. Interested in finding wacky problems to solve.

I've used Vim (casually) for 3 years.

Required Software

You need **vim** (that's what we're learning today)

Download instructions:
`https://www.vim.org/download.php`

Why do we want to learn an editor?

- We don't have workshops on how to learn a web browser, so why are editors important?
- Writing code, or editing files on a computer has a lot of moving parts: you spend a lot more time switching files, reading, navigating, editing code compared to writing a long stream of words sometimes.
- As a power user, you probably will spend a lot of time doing these things, it might be good to find an editor that will **speed up your workflow.**

How to learn an editor

- Start with the **fundamentals** (this is what we'll cover)
- **Practice** as much as possible
- Within about 10-20 hours of use, you'll be back to your normal speed
- Look things up as you go: find out shortcuts to doing things
- After that your new editor should start to save you lots of time

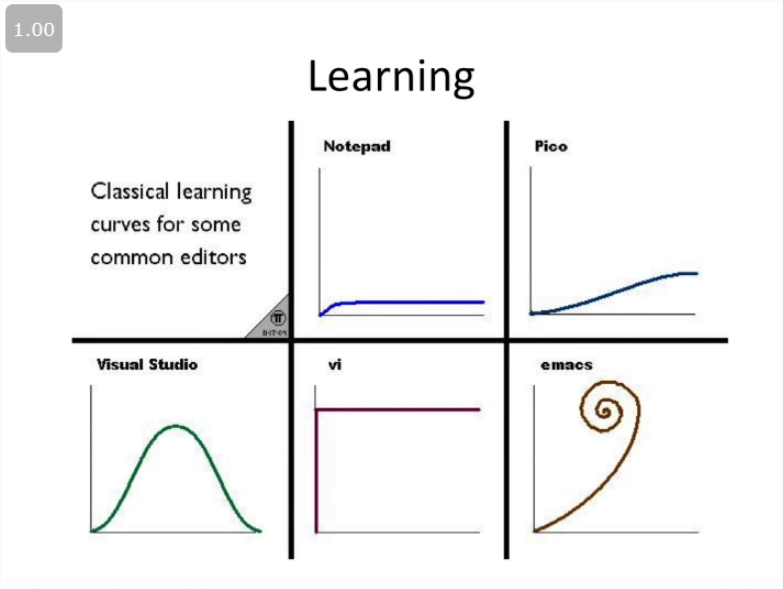
What to expect today?

Today we'll be covering one such editor, **vim**.

The idea is to kickstart you knowledge, and give you the fundamentals and resources to go off on your own.

Which editor to learn?

- People tend to have very strong opinions on this. See `https://en.wikipedia.org/wiki/Editor_war`
- Vi(m) vs Emacs
- It's completely up to you! we'll cover one of the popular options out there, vim but you can also look up Emacs and choose one that interests you most



Origins of Vi(m) (1/2)

- Vim stands for vi iMitation, later changed to vi iMproved
- Created by the late Bram Moolenaar in 1991.
- It is based off another text editor, **vi**, created by Bill Joy in 1976.

Origins of Vi(m) (2/2)

- Bill Joy was trying to create an editor that was usable with a 300 baud modem. (approximately 0.3 kbit/second today). Essentially, you could only type one letter a second, so the commands had to be really short.
- As it turns out, when you optimize heavily to minimize the number of keystrokes, you have a **really efficient editor**

Benefits of Vim

- Vi(m) is the de facto default editor in any Unix-based system. You can probably find vi/vim in any and every Unix-based system today
- Vim is also extremely customizable and programmable
- There is a huge community and plugin support for almost anything imaginable

Philosophy of Vim

- Vim is a **modal text editor**
- We have different modes for inserting text vs manipulating text
- The idea here is that you tend to spend more time reading/making smaller edits, instead of writing big essays in one big go.

Where are we?

Modes of Vim

There are a few primary modes of vim:

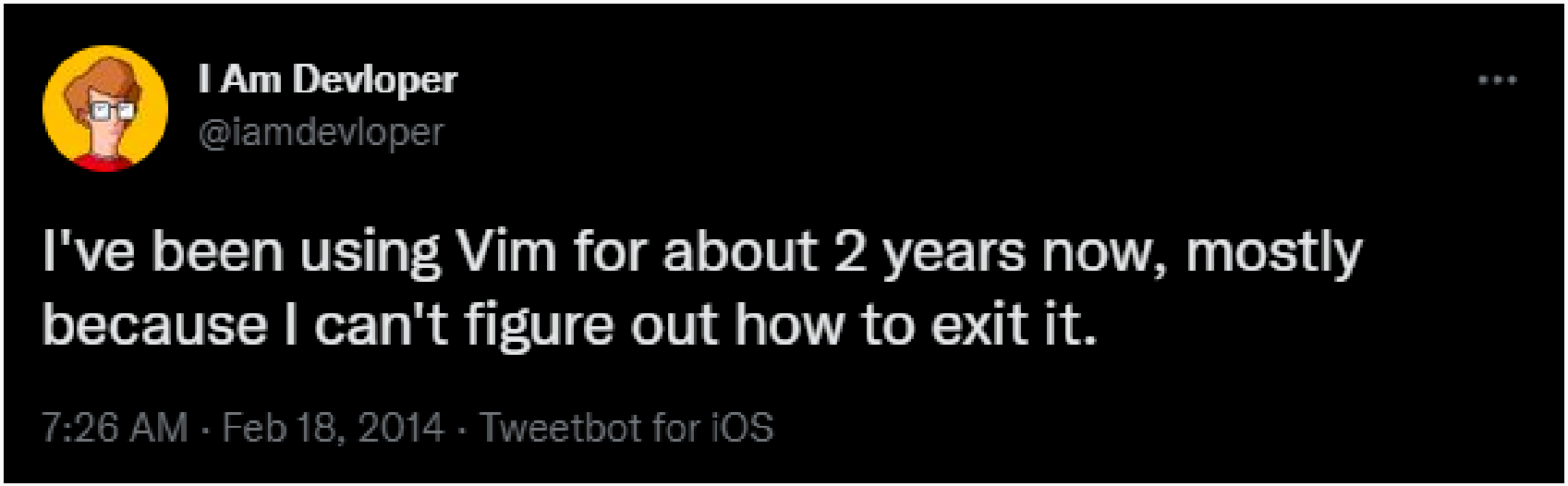
- Normal Mode - For moving around a file and making small edits
- Insert Mode - For inserting text
- Visual Mode - For selecting blocks of text (plain, line or block)
- Command Mode - For entering commands

Modes of Vim

Keystrokes have different meanings in different operating modes. for example, x in insert mode will insert a literal character ‘x’, but in normal mode, it will delete the current selection

Opening Vim and quitting it (1/2)

There's a common joke that if you give a web designer a computer with vim loaded up, and ask them to quit vim, you get a random string generator



Opening Vim and quitting it (2/2)

To open vim, just type `vim` in the terminal. If you are using gVim, you can just open the executable.

To close vim, type `:q`. We'll go through what this means later

Changing modes

By default, you start vim in **normal mode**. For most cases, you will transition from normal mode to another mode, based on your use case, and then return back to normal mode after.

Normal to Command Mode

Command mode is where we run commands similar to a command line in vim. To go to this mode, simple press `:`. A text bar should appear at the bottom of the screen. From there, we can execute several vim commands.

- `:` - go to command mode
- `q` (in command mode) - quits the file
- `w` (in command mode) - saves the file
- `!` - force an action
- `:wq` - save the file then quit
- `:q!` - force quit file without saving

Once you are done, vim should automatically put you back in normal mode

Any mode to normal mode

Normal mode is your default mode where you should spend most of your time. In vim, if you ever get lost or are not sure what is happening, always **reset to normal mode**

Esc will bring you to normal mode from any of the other modes. You will be using this a lot.

Why Escape?

It might seem quite counterintuitive to use escape since it's quite out of place on your keyboard. However, vi was created using an ADM-3A terminal. It looks like this:

Some programmers also map Caps Lock to Esc or other mappings for convenience

Changing modes

Mode	Description	Hotkey
Normal	Navigate the file	Esc
Insert	Inserting text	i, I, a, A, o, O
Command	For entering commands	:
Visual	For selecting text	v, V, Ctrl + v

Where are we?

Navigating in normal mode

First let's open up a file using vim by using `vim (filename)`.

We can navigate the file by using `hjkl` (left, down, up, right respectively)

Why not arrow keys (or a mouse)? Historically, it's because the old keyboards did not have arrow keys or a mouse.

However, in practice, it is extremely efficient as you don't need to move you hands away from the alphanumeric keys to do anything.

A simple vim config

Notice that by default, vim looks pretty bad. Let's spice it up a bit with some simple configs.

Copy this config file to `~/.vimrc`, or modify your existing config: `https://hckr.cc/ht-vim-vimrc`

Normal Mode

Type	Description
Basic	hjkl: left, down, up, right
Word	w: next word, b: back a word
File	gg: top of file, G: bottom of file
Line	0: beginning of line, $: end of line, ^ first non-whitespace of line
Line Numbers	34G: to go to line 34 in the file
Screen	H igh part of screen, M iddle of screen, L ow part of screen
Braces	% to go to corresponding braces
Repeating	10j: to go down 10 times
Scroll	Ctrl + d: to scroll down, Ctrl + u to scroll up

Searching

Searching is slightly different in vim, there are two main ways to search something:

- Find inline: `f` to find further up in the line, `F` to find in everything before the cursor
- Search in file: `/` + `query` to search forward in the file from the cursor, `?` to do the opposite. `n` to go to next result and `N` to go to previous result.

Where are we?

Insert Mode

1. Make sure you are in normal mode. Esc
   - 1.1 i to insert before cursor
   - 1.2 I to insert at the start of line
   - 1.3 a to insert after cursor
   - 1.4 A to insert at the end of line
   - 1.5 o to start a next line and insert
   - 1.6 O to start a line above the current selection and insert
2. Get out of insert once done. Esc

Making small edits - delete

In normal word, you can quickly delete a portion of text

- d + modifier, deletes a certain portion based on the modifier
- dw - delete word
- 6dw - delete 6 words
- dd - delete the entire line
- d$ - delete till end of line
- dt + char - delete till character

Making small edits - change

Similarly, change allows you to quickly delete and change a certain portion of text

- `c` + `modifier` - deletes then puts you into insert mode
- `cw` - change word
- `7cw` - change 7 words
- `c$` - change till end of line
- `ct` + `char` - change till certain character

Making small edits - misc. (1/2)

- y + modifier, yank a certain portion and puts it in a put buffer (think of ctrl c)
- yy - yank entire line
- yw - yank word
- 6yw - yank 6 words
- yt + char - yank till character
- p - put/paste whatever was in the buffer
- P - put/paste in the line above

Making small edits - misc. (2/2)

- x to delete a certain character
- r to replace a character
- . - do last action
- u - undo action
- Ctrl + r - redo action

Practice 1

`https://hckr.cc/ht-vim-p1`

Try and fix the typos, and small errors here!

Where are we?

Visual Mode

There are a few kinds of visual modes:

- Visual [v]
- Visual Line [V]
- Visual Block [Ctrl]+[v]

We can use these selections along with the above commands: [y]ank, [d]elete, and [c]hange.

Opening Files and other commands

Aside from saving and quitting, there are a couple of pretty important commands to know:

- `:enew` - opens a new file
- `:e` + `filepath` - open the file at path
- `sp` - open a new split
- `vsp` - open a new vertical split

Macros

Macros are one of the really powerful features in vim that can significantly speed up your workflow.

- `q` + `keystroke` to start recording a macro, `q` again to stop recording
- `@keystroke` to apply the macro

Practice 2

`https://hckr.cc/ht-vim-p2`

It seems like the data from the first 3 columns are corrupted, let's remove them!

`https://hckr.cc/ht-vim-p3`

Let's use a macro to extract the names from these emails!

Extending Vim

One of the greatest features is that it is extremely customize and very easy to customize too! There are many great configs and plugins created for different workflows, which we can't cover in this session. Here are a few recomendations:

- `https://github.com/amix/vimrc`
- `https://vimconfig.com`
- `https://github.com/gunbux/dotfiles`

Getting productive quickly

There are many (neo)vim-based distributions that are pre-configured with sane defaults and plugins to get you productive immediately:

- `https://www.lunarvim.org/`
- `https://astronvim.github.io/`
- `https://nvchad.com/`

Others

We covered the base fundamentals of operating vim. Here's some things we did not cover, but are really useful:

- `di(` - delete everything inside the nearest brackets
- find and replace
- buffers, windows, and tabs
- ...a lot more!

Where are we?

Using vim outside of vim

Vim's hotkeys are very well loved, and many people love putting vimbindings on everything (myself included). Here are some you can look at:

- Vimium - Vim for Chromium based browsers
- IdeaVim - Vim emulation for JetBrains
- Homerow - Use your keyboard for your whole Mac

I need help!

Vim is a really powerful editor if you are able to master it. Don't worry, there are plenty of resources!

- `:help command` - get manual for a specific command
- `vimtutor` - built in vimtutor, give it a read, it shouldn't take more than 30 mins
- VimGolf - really good practice: edit the file in the minimum strokes
- There are tons of vim wikis, guides and cheatsheets out there, just search for them!

Talk to us!

- **Feedback form**: `https://hckr.cc/ht-feedback`
- **Upcoming Hackerschool**: Tomorrow – Introduction to Rust: `https://hckr.cc/hs2324s2-w4-signup`