CS45, Lecture 5
Text Editors

Winter 2023

Akshay Srivatsan, Ayelet Drazen, Jonathan Kula

Administrivia

- **Assignment 2** was released on Monday. After today's lecture, you should have all the tools you need to complete it!
  - A2 is due next Monday the 30th at 11:59 pm
- Lectures are now being recorded as a back-up!
  - They can be found in Canvas
  - They still won't be professional-quality since we're managing them ourselves, so no guarantees, but they're there as a backup for if you can't make it to lecture.

Learning Goals

- Understand the use case of rich text editors vs plain text editors
- Understand the use case of TUI vs GUI text editors
- **Have some concrete practice using the `vim` editor**
- **Have some concrete practice using Visual Studio Code**

Rich Text

- *Rich Text* **allows you to** format your text however you'd like.
- It's extremely flexible, **but requires a ton of additional data to be associated with the text.**
- Information is structured around elements of prose: words, paragraphs, headings...
- Rich text is **for humans, not for computers.** (Computers don't need all the extra information!)
- Example rich text editors: Word, Google Docs, WordPad
- We're not going to focus on rich text during this course, but it's useful to know the difference.

Plain Text

Meanwhile...

- Plain text is how we communicate with computers (for the most part)
- Myriad applications exist for text editing
  - GUI (Graphical) applications: **Visual Studio Code**, JetBrains IDEs, TextEdit...
  - TUI (Terminal UI) applications: **vim**, emacs, nano, micro*
  - CLI (command-only) applications: ed, ex (and via scripting, as seen before!)

Learning a new editor

- New editors have learning curves! (No matter the kind!)
- Our recommendation: **Choose one visual IDE** (Visual Studio Code is what we'll be using) and **one TUI editor** (we'll be showing off vim) to learn
- It'll be slower at first, but after 10-20 hours of practice, you'll be just as fast, and then faster than others after 20!
- Look things up! Often there's a faster way to go about doing things.
  - Build up your knowledge base as you go!

Let's learn vim!

A Quick History

- **vim** was inspired by and spun off of **vi**, and stands for **VI iMitation**.
- **vi** was one of the first TUI editors, based on the editor **ed** (and the **vi**sual mode of CLI tool **ex**), which required you to edit line by line using certain commands.
- **vi**, and **vim**, continue to use that idea of **commands and modes**.

Bill Joy, the creator of **vi**

The Modal Editor

vim uses different “modes” to control editing.

- You always start in **normal mode**, used for navigating around the file.
- You press **i** to enter **insert mode**, to write text
- You press **R** to enter **replace mode**, to overwrite text
- You press **v** to enter **visual mode**, for copying or deleting lines of text at a time
- You press **:** to enter **command mode**, which allows you to do all sorts of things (like save, quit, find-replace, etc)

Demo time!

Follow along on your terminals!

```
curl -Lo vim_nav.txt https://cs45.stanford.edu/res/lec5/vim_nav.txt
```

```
vim vim_nav.txt
```

Windows & Buffers

- **vim** differentiates between “buffers” and “windows.”
- **Buffers** are an open file. A buffer can be open in one or more windows.
- **Windows** are “views” into a buffer.
    - You could have multiple windows open to the same buffer!
    - This means that your changes in one window will instantly reflect in the other
- :q closes the current window

Configuring vim

- You can customize **vim** by writing a **.vimrc** file in your home directory
  - My .vimrc makes the mouse work, adds line numbers, and makes backspace and arrow keys work like I'd expect from an editor
- You can add even more plugins either manually or using a plugin manager like **vundle**
  - I have plugins that let me search files, match braces, mark indentations, etc

Demo time!

Let's take a look at my .vimrc file

```
vim https://cs45.stanford.edu/res/lec5/.vimrc
```

From TUI to GUI

What are IDEs?

Integrated Developer Environments, or IDEs, are applications for software code editing that bundle together **lots of functionality for developer productivity into one place.**

In particular, they usually bundle code editing with syntax highlighting and autocomplete, error checking, build tools, testing tools, and the ability to run code all into one application.

Visual Studio Code

Why VSCode?

- VSCode strongly supports **remote editing**, allowing you to access and edit resources **on a server**, *without* needing a GUI shell to be installed on the server at all.
- Some other IDEs are slowly starting to support this (e.g. JetBrains, in beta), but VSCode is also **free** and has **wide language support**.

Demo time!

Let’s see some of the things VSCode can do!