Vim Text Editors

1 minute read

- Edit text files in a terminal
- Quit Vim `:wq` to save, `:q!` to not save

Excellent introduction lecture to Vim (thanks Matthew!) The lecturer uses my preferred setup, Vim with tmux (terminal multiplexer).

Concepts:

- Vim is a modal text editor. The modes are:
  - normal for navigation and edits
  - insert for writing large blocks of text
  - visual for making visual selections (highlighting)
  - command for program commands
- Vim is a programming language. It shares syntax with other shell tools, such as `sed`.
- Keep your fingers on the home keys (asdf jkl;). You will develop muscle memory, and be able to edit “at the speed of thought”. Other programs can use the same key bindings, even web browsers.

Why Vim?

It’s faster. You will be more productive, and you will never outgrow the capabilities.

Other reasons not highlighted in video:

Vim works in a terminal, and everywhere else.

Why is this important for our class?

Why don’t we have a desktop GUI on our Amazon EC2 instances?

We don’t need the overhead. What kinds of overhead?

1. Network bandwidth to send high resolution video of a remote desktop.
2. Server CPU and memory. If we’re using a micro instance with 1 CPU and 0.5 GB memory, can we even run a conventional desktop?

Emacs

Most people who program a lot in a terminal eventually stick with one of two programs: Vim or Emacs. Emacs has similar capabilities to Vim.

One high level difference between Emacs and Vim is that Emacs uses “chords” for commands: pressing several keys simultaneously, while Vim uses “melodies” for commands: pressing several keys in order.

Updated: January 22, 2021