Intro to VIM

Fall 2025

Luke Deany

Linux Users Group @ UIC

lug.cs.uic.edu

lugatuic

Introduction

- Goal is to get you started with bare basics of Vim
- Then get you to practicing Vim ASAP
- Ask questions if you have any, especially during practice section

Overview

1. What is Vim?
    - An overview of what vim is, and why you should know how to use it.
2. How can I use Vim?
    - An overview of how you can install vim, or simply how to use setup Vim keybinds in an editor of your choice
3. Vim Basics
    - An overview of the basic commands you will use in vim just to get around
4. Interactive Vim Tutorial
    - Finally, we'll get you set up on an interactive tutorial for vim

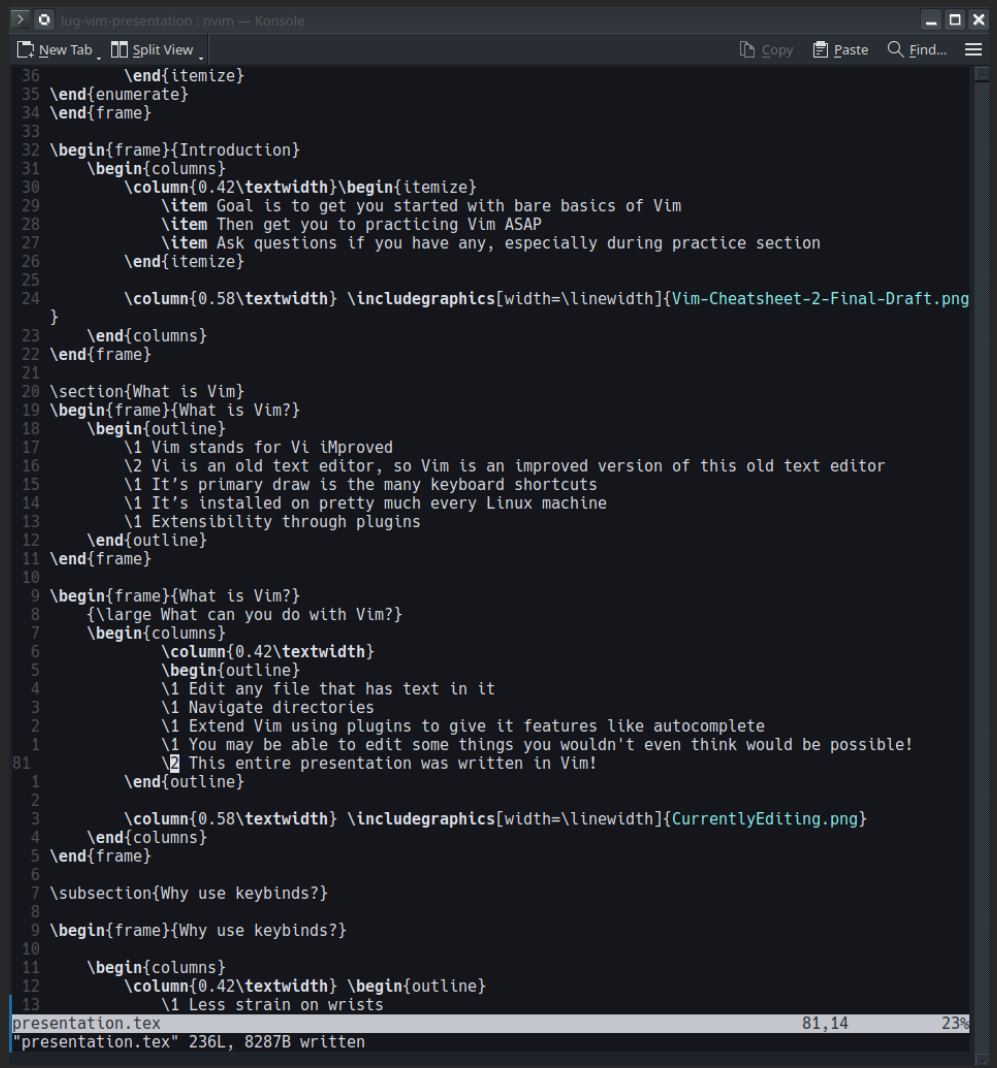
What is Vim?

- Vim stands for Vi iMproved
  - Vi is an old text editor, so Vim is an improved version of this old text editor
- It’s primary draw is the many keyboard shortcuts
- It’s installed on pretty much every Linux machine
- Extensibility through plugins

What is Vim?

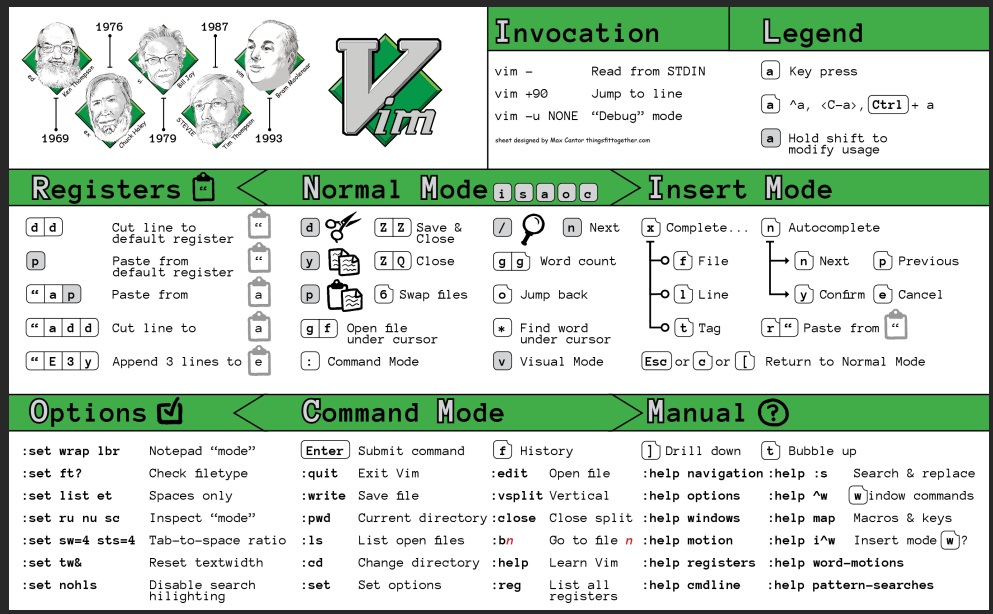
What can you do with Vim?

- Edit any file that has text in it
- Navigate directories
- Extend Vim using plugins to give it features like autocomplete
- You may be able to edit some things you wouldn’t even think would be possible!
  - This entire presentation was written in Vim!



Why use keybinds?

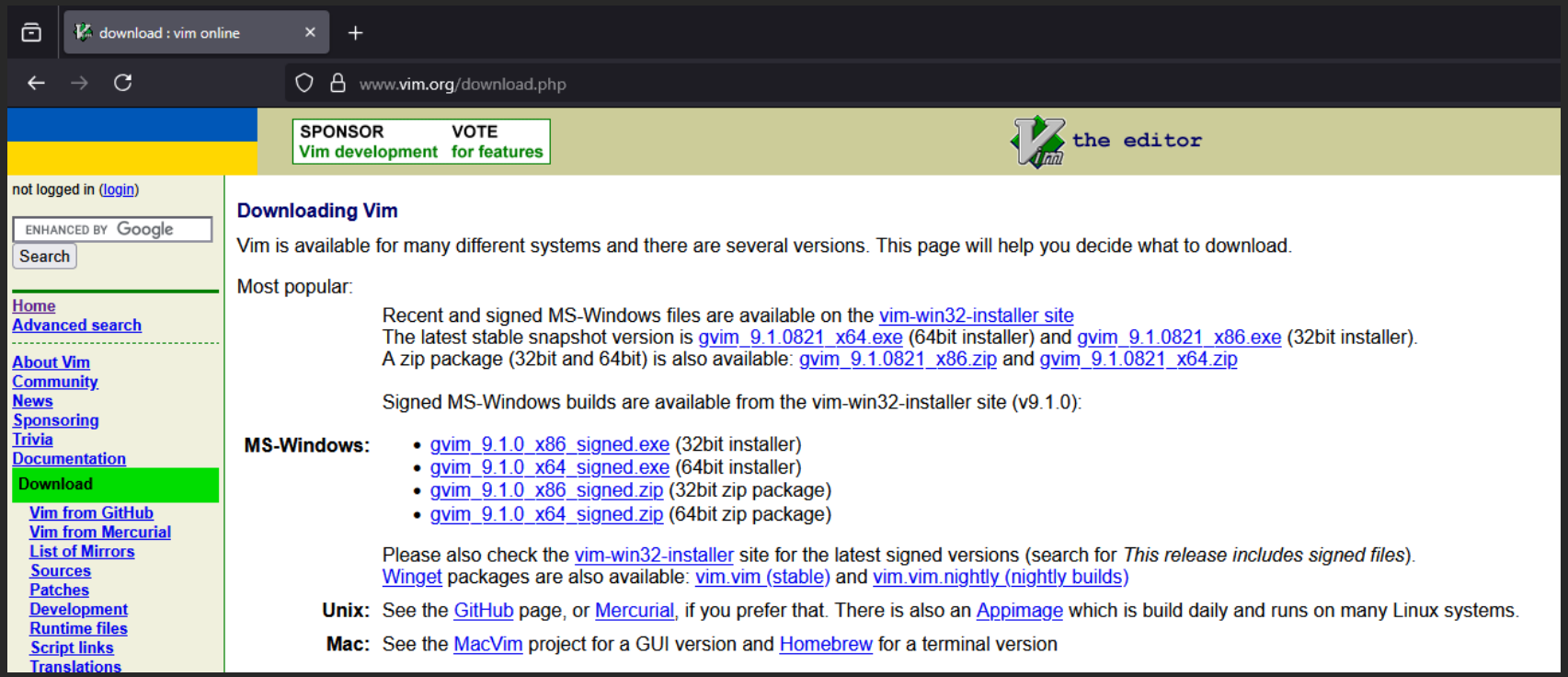
- Less strain on wrists
  - Do not have to switch back to mouse
- Increase in speed
- Uniform across editors



https://thingsfittogether.com/product/vim-cheat-sheet-basics-print/

How can I use Vim?

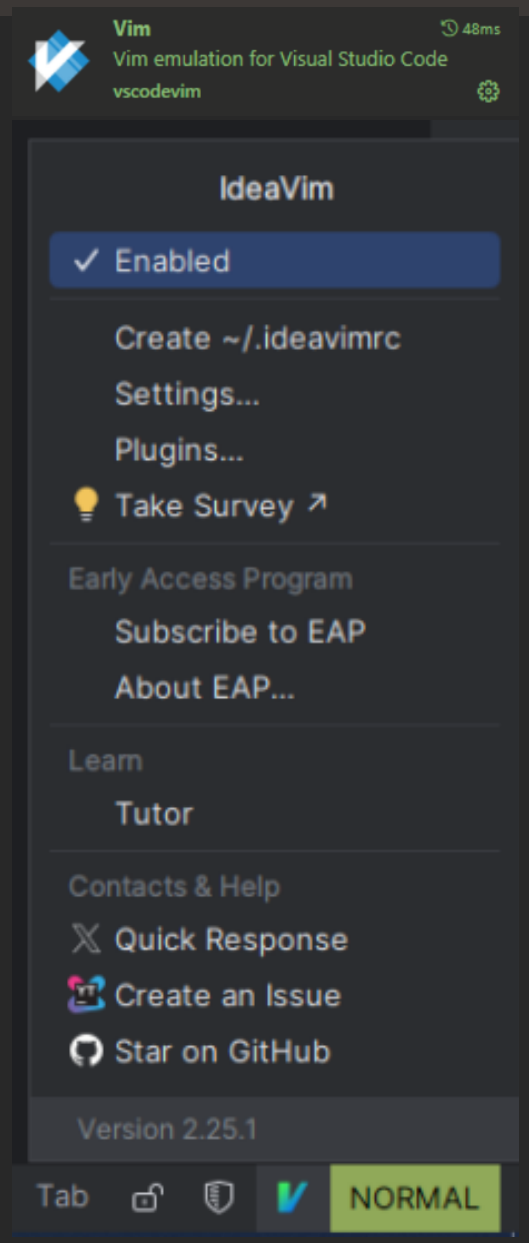
1. On Windows you can download Vim from their website, or using WSL (Windows Subsystem for Linux) will also have Vim installed
2. If you're on a Linux machine, you already (most likely) have it installed, run "vim" in the terminal
3. MacOS has Vim installed by default, but it is a limited version, you can install the full version using homebrew



How can I use Vim?

How can I use Vim keybinds?

- For pretty much every major editor, you have two options
  - Enable Vim mode if built-in
  - Install a Vim keybinds plugin



Vim Basics

Vim has many concepts that are useful to learn, but we will boil it down to three for this presentation.

1. Buffers
2. Modes
3. Keybinds

Buffers

- Buffers are like the clipboard on your computer
  - The buffer is separate from your clipboard
  - Things you copy in vim will not copy to your clipboard, and vice versa
- You can have as many buffers as you want
  - For this example we will stick to only 26, the characters in the English alphabet.
- Before inputting a command that uses buffers (you will see some in a second), you input double quote (") followed by the name of the buffer.

Modes

What are modes?

- Modes are how you operate using vim, and each mode does different things. You can switch modes at pretty much any time.
  - This may be kind of confusing to think about at first, but you already are familiar with this concept if you use another development environment!
- For instance, if you are writing code in visual studio code, and you then use your mouse to highlight text, you can think of it as "switching" into visual mode.
- You can see what mode your in by checking the bottom left, but this may be changed by different vim configurations.

Modes

What modes are there?

- Normal Mode
  - Vim “home base”, allows you to switch to different modes
  - Also used for things like rearranging text (copying and pasting)
- Insert Mode
  - The most common mode, in this mode any text you write will actually be written to the file
- Visual Mode
  - Allows you to select larger blocks of text visually, useful for copying and pasting, or deleting large sections
- Command Mode
  - Allows you to enter commands to Vim, which is used for things like saving, among plenty of other things
- Replace Mode
  - Like Insert Mode, but will directly write over text rather than adding new text

Modes

Insert Mode

- Most common way to enter insert mode is by pressing ‘i‘ or ‘a‘
  - ‘i‘ inserts before cursor, ‘a‘ inserts after cursor

Modes

Visual Mode

- Most common way to enter visual mode is by pressing ‘v‘
- Allows you to move around your cursor with hjkl to select different portions of text
  - Can be used to yank (copy) large portions of text to other areas
- Press escape to exit visual mode

Modes

Command Mode

- Entered by press ‘:‘, you will see a the command in the bottom left
- Once you’ve typed a command, press enter
  - Example, ‘:wq‘ to enter command mode, then "write" and "quit"

Modes

Replace Mode

- Entered by pressing ‘R‘
- Allows you to type directly over existing text, replacing it

Keybinds

- Keybinds are used in Vim to tell the program what you want to do
- They use "context" from where your cursor is or what you have selected
- Some change modes, some move the cursor, and some edit text
  - There are hundreds of keybinds! You don’t have to learn them all to use Vim
- Some keybinds can be combined to do new things

Keybinds

HOW DO I EXIT HELP

- In command mode, type ‘w‘ to signify you’d like to write, or save, the file.
- In command mode, type ‘q‘ to signify you’d like to exit, or close, the file.
- You can combine these commands into "wq" to signify you’d like to save and exit the file!
    - If you have modified a file since you’ve opened it, ‘q‘ will not work, you must either write ‘wq‘ to save and quite, or q! to quit without saving

Keybinds

Basic text manipulation commands

- In normal mode, use ‘x‘ to delete one character
- In normal mode, use ‘r‘ to change the character under the cursor
- In normal mode, use ‘c‘ to change large groups of words using different contexts
- In normal mode, use ‘d‘ to delete large groups of words using different contexts

Keybinds

Context, how we tell Vim what we want

- Most basic context is a number, ‘10x‘ will delete 10 characters
- Other easy context is simply repeating the same command to indicate a whole line
  - Using ‘dd‘ will delete the entire line under the cursor
- ‘f{character}‘ indicates find, and execute the command until the next occurance of the character you specify, inclusive
  - You can use a number modifier on this! ‘2fi‘ will find the 2nd occurance of i from the cursor!
- ‘t{character}‘ indicates until, meaning it will execute the command until the next occurance of the character you specify, exclusive
  - You can once again use the number modifier on this!

Keybinds

Context Notes

- There are a lot more things you can do with context, but outside scope of beginner.
- If you didn’t understand this, that’s okay! It will come with time and using Vim.
- Once you get a grasp on how this works you can edit large groups of text very quickly.

Keybinds
Moving Text

- By default, you can’t just copy and paste
- Instead, commands like ‘d‘ for delete will store the line in the main buffer.
  - Buffers mean the text is saved and can be pasted in later
- To store text into a buffer without deleting it, use ‘y‘ in normal mode with context. It stands for ‘yank‘
- To paste text from a buffer, use ‘p‘ for paste
  - For all copy, paste, and delete commands you can specify a buffer before the command to put the data into a different buffer.
- ‘"add‘ deletes an entire line from the file, and places it into buffer a (also places it in main buffer)

Keybinds

What does this keybind do?
‘"ap‘

Interactive Tutorial

- Now we will do an interactive tutorial for Vim using vimtutor
- vimtutor allows us to practice editing text using Vim
- Feel free to do this on your own by running ‘vimtutor‘ in a shell
    - If this doesn’t work, we will complete it as a group as well