Vim Tutorial

Alex Antonakakis, Stelios Malamas

University of Crete CSD

February 2024

Introduction

- Vim is a free and open-source text editor
- Released in 1991
- Available on all popular package managers
  - apt-get install [Debian/Ubuntu/Mint]
  - brew install vim [macOS]
  - dnf install vim [Fedora22+/CentOS]
  - pacman -S vim [Arch]
  - zypper install vim [openSUSE]

Why Vim

- Vim is a free and open-source text editor
- Preinstalled on many Unix-like systems
- Complete functionality over SSH
- Offers tons of plugins and extensions
- Fully configurable via .vimrc file

Modes

- Vim supports 4 modes
  - **NORMAL**: This is the default where you can navigate and switch between modes.
  - **COMMAND**: This is where you type commands.
  - **INSERT**: In this mode, you can directly insert and edit text.
  - **VISUAL**: This mode allows you to visually select blocks of text, which can then be manipulated in various ways.

Package managers

- Vim supports several package managers that streamline the process of installing, updating, and managing plugins and extensions:
  - **Pathogen**: One of the earliest package managers for Vim, it simplifies plugin management by allowing each plugin to reside in its own directory.
  - **Vundle (Vim Bundle)**: Vundle simplifies plugin installation by specifying them in a configuration file and fetching them directly from GitHub repositories.
  - **Vim-plug**: A modern and minimalist plugin manager for Vim, Vim-plug supports lazy loading, parallel installation, and update mechanisms.

Entering Vim Modes

- **Normal Mode**: Default mode in Vim.
  - To enter Normal Mode, press the **Esc** key.
  - You have to be in normal mode to run any of the commands shown below.
- **Command Mode**: Default mode in Vim.
  - To enter Command Mode, press the **Shift** + **:** .
- **Insert Mode**:Insert or edit text.
  - To enter Insert Mode:
    - ★ Press **i** to insert before the cursor.
    - ★ Press **a** to insert after the cursor.
    - ★ Press **I** to insert at the beginning of the line.
    - ★ Press **A** to insert at the end of the line.
    - ★ Press **o** to open a new line below the current line.
    - ★ Press **O** to open a new line above the current line.
- **Visual Mode**: Useful for selecting text.
  - To enter Visual Mode:
    - ★ **v** to enter character-wise visual mode.
    - ★ **V** to enter line-wise visual mode.

Useful Vim Commands: Navigation

- **h, j, k, l**: Move left, down, up, and right respectively.
  - You might as well use the arrows, it's up to you
- **w, b**: Move forward and backward by a word.
- **gg**: Go to the first line.
- **G**: Go to the last line.
- **#G**: Go to line number #.

Useful Vim Commands: Editing

- **i, a, o**: Insert before the cursor, after the cursor, or on the next line respectively.
- **dd**: Delete line.
- **dw, db**: Delete the next word, delete the previous word.
- **yy**: Yank (copy) the current line.
- **p**: Paste yanked or deleted text
- **u**: Undo
- **Ctrl+r**: Redo

Introduction to Bash

- **Bash**: Command language interpreter
- **Shell**: Macro processor which allows for interacting or non-interacting command execution
- **Script**: Allows for an automatic command execution

Basic Shell Commands

- **ls**: Lists the contents of the current directory
- **cd foo**: Change current directory to foo
- **pushd/popd**: Push/Pop directory into directory stack
- **mkdir foo**: Create a new directory named foo
- **touch bar**: Create a new file named bar
- **rm bar**: Delete the file named bar
- **cat bar**: Print the contents of the file named bar

I/O Redirection

- $>$: Standard Output (stdout)
  - **ls** $>$ **files.txt**: Writes output of the ls command to files.txt
- $<$: Standard Input (stdin)
  - **cat** $<$ **files.txt**: Takes input from files.txt and redirects it to cat
- **2**$>$: Standard Error (stderr)
  - **ls 2**$>$ **files.txt**: Writes the error output of the ls command to files.txt

Printing files

- **more foo.txt**: Filter for paging through the contents of foo.txt one screenful at a time
- **less bar.txt**: Similar to more command, but it allows backward as well as forward movement
- **head -N foo.txt**: Prints the first N lines of foo.txt
- **tail -N bar.txt**: Prints the last N lines of bar.txt

Search

- **grep foo bar.txt**: Search for "foo" in bar.txt
- **find ~ "foo.txt"**: Search for foo.txt in the directory tree that starts from the home directory

Pipes

- Unix allows multiple command chains.
- The output of the command on the left becomes the input of the command on the right.
- **ls** | **sort**: Directories and files are listed and then sorted.

Useful notes

- $VARIABLE to access a variable in bash.
- $? to access the return value of the command previously executed.
- **cd** - will take you back to the previous directory
- Each directory always has 2 subdirectories
  - **.** which is the **current directory**
  - **..** which is the **previous directory**
  - Useful when you have walked through a long path and want to go back to the previous directory fast
  - **cd some/long/path/a/b/c/d** *work on dir* **cd -**
- Enclose command within

  

  to run a command within a command.
  - ls /home/‘whoami‘/