NUS | Computing

National University of Singapore

Eric Han
eric_han@nus.edu.sg
https://eric-han.com

Computer Science

6 August 2024

Introduction to Unix/Linux

CS1010 Workshop

Content I

Follow Along!

Figure 1: Slides available - https://eric-han.com/teaching/tools/nix_slides.pdf

Your Instructor - Dr. Eric Han

- **[Pioneer JC 2010]** Took H2 Computing, interested in Computing/research.
- You are here.
- **[B.Com. NUS 2018]** Not so long ago I was in your shoes!
  - A*STAR Scholarship, Turing Programme
  - [University of Southern California, 2016] Student Exchange
- **[PhD. NUS 2023]** PhD in AI/ML.

You are welcome to check my profile & research: https://eric-han.com.

> *Is it possible to use **only** Linux?*

My Setup

Operating Systems

1. Windows – since Primary 5, to play games that don't exist on Linux.
2. Linux – since JC, main driver.
   - Fedora Linux
   - NixOS
3. macOS – since BCom, to develop iOS/mac applications.

Tools on Linux

1. Typesetting/PPT: Google Workspace, LibreOffice, TEX, Markdown
2. Coding Tools: vim, VScode

Section 1: **Why *nix?**

Figure 2: Image taken from Linux Inside (Linux Inside 2017)

Reason 1 - Career

*nix OS is the **dominant** Operating System (OS) in the tech world, and as a computing student, you will likely have to interface with one sometime in your career.

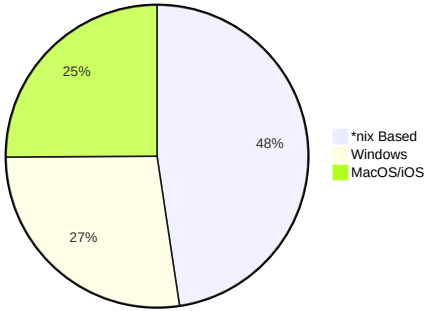

Figure 3: Primary OS (Stack Overflow 2013)

Reason 2 - SW Development

*nix is probably the most productive programming environment you can have to do most of your day-to-day tasks and to develop software as a computing professional.

Install Node.js on Linux

```
# installs nvm (Node Version Manager)
curl -o-
 ↪  https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh |
 ↪  bash

# download and install Node.js (you may need to restart the terminal)
nvm install 20
```

Reason 3 - Productivity

*nix's programmability allows for easy automation of tasks to boost productivity by composing new tools from existing ones.

Print in SoC

Problem: I need a command to print a PDF to a network printer.

```
#!/bin/bash
FILENAME=$( echo "$*" )
smbclient -U NUSSTF/username%password //nts09.comp.nus.edu.sg/psa427 -c
 ↪  "print \"$FILENAME\""
```

Solution: `$ soc-print hello.pdf`

Reason 4 - Well-Designed

The Unix philosophy serves as a great example of how good software should be designed.

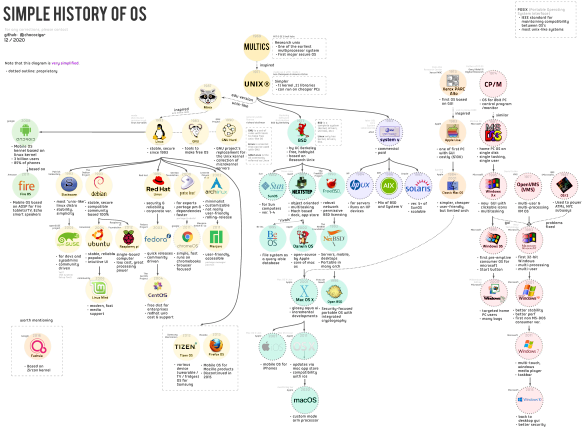

Figure 4: OS Family Tree (chococigar) – Most roads lead back to Unix.

Section 2: **What is *nix?**

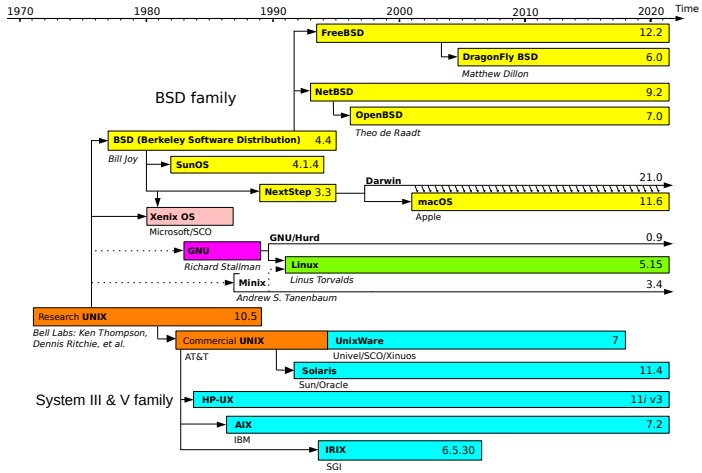

Figure 5: Abridged Timeline of Unix and Linux (Guillem 2021)

Unix is an OS that was developed in the late 1960s and the early 1970s, by Dennis Ritchie and Ken Thompson from Bell Laboratories.

Figure 6: Ken thompson (sitting) and Dennis Ritchie

Unix Philosophy

Summarized by (Salus 1994) – Write programs

- that do one thing and do it well.
- to work together.
- to handle text streams, because that is a universal interface.

Although it was well used by businesses & universities, it was **not free**.

> *CLI is a text-based (simple and fast) way to instruct the computer to perform a task. We will use CLI as opposed to GUI to interface with nix in our exercises!*
>
> Command-Line Interfaces (CLI)

Linux is actually GNU/Linux:

- Linux kernel – initially developed from scratch as a hobby, by Linus Torvalds
- GNU project (tools) – which aims to replace Unix, by Richard Stallman

Since, Linux it follows many Unix principles, we will now refer to the family as *nix.

Figure 7: Linus Torvalds & Richard Stallman – edited (Brys 2012).

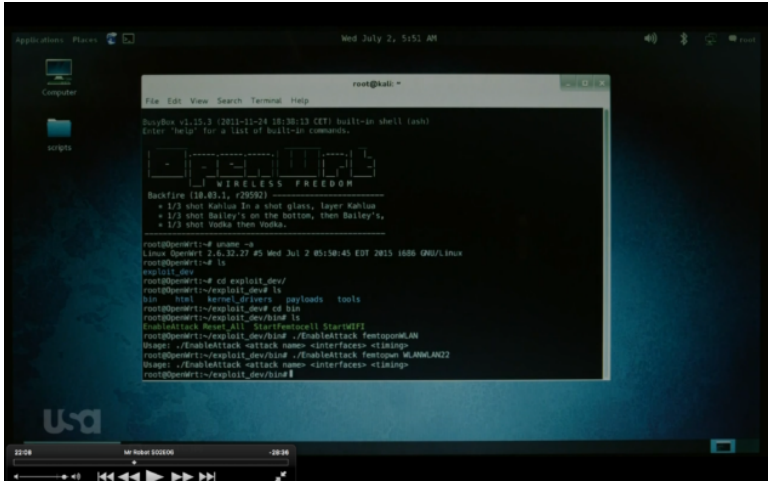

Figure 8: Screengrab from Mr. Robot, eps2.4_m4ster-s1ave.aes

Linux Command Prompt

Terminal (emulated) / Console (physical)

Program that takes keyboard inputs and prints text output to the user.

- Windows Terminal
- Terminal
- xterm, Console, konsole

Shell

Program that interacts with the OS to executes commands (from terminal or scripts).

- Powershell (ps), MS-DOS (cmd)
- Bourne Shell (sh), Bourne Again Shell (Bash)
- Z shell (Zsh)

Command Prompt

Where the user (`$`) or root (`#`) types a command, for the shell to execute a task:

```
[eric_vader@thinkpad:~]$
```

Section 3: **Linux Navigation**

Key differences

Since macOS is also part of the *nix family, we focus on Windows vs *nix:

.	Windows	*nix
Organization	Usually `C:\`	Single 'root' `/`
Case	Case-insensitive	Case-sensitive
Path separator	`\`	`/`
Forbidden Characters	`<>:"/|?*`	`/`
File System	NTFS,exFAT	UFS(Unix),Ext4(🐧),APFS(🍎)
Hidden Files	File Property	Names starting with `.`

Dir Structure (vs. Windows)

- Home directory (user `ehan`):
  - `C:\Documents and Settings\ehan`
  - `/home/ehan`, `/home/e/ehan`
- External media:
  - Drives – `C:\`, `D:\`
  - `/mnt`, `/media`, `/cdrom`
- Programs:
  - `C:\Program Files`, `C:\Program Files (x86)`
  - `/opt`, `/usr`, `/sbin`, `/bin`
- Temporary directory (user `ehan`):
  - `C:\Users\ehan\AppData\Local\Temp`
  - `/tmp`, `/var`
- Libraries:
  - `C:\Windows\System32`, `C:\Windows\SysWOW64`
  - `/lib`, `/lib64`

Dir Structure

- Devices: `/dev`
- Kernel (pseudo): `/proc`, `/sys`, `/dev`
- Boot: `/proc`
- Configuration: `/etc`
- Site-Specific: `/srv`
- Recovered files: `/lost+found`
- Root home: `/root`

Remark

Not all nix systems have every folder, but most include the essential ones.

Path in *nix

Path is a unique location to a file/directory.

Absolute Path

- Always starts with `C:\`
  - `C:\Windows\calc.exe`
- Always starts with `/`
  - `/bin/bash`

Relative Path

- `calc.exe`
- `bash`

Symbol	Meaning
`~`	home directory
`.`	current working directory
`..`	parent directory

Basic Navigation Commands I

Print current Working Directory `pwd`

```
$ pwd
/home/eric_vader
```

```
# pwd
/root
```

Remark (User vs Root prompt)

User prompt is denoted by ('$') and root's is ('#'). Root can execute anything user can execute.

Basic Navigation Commands II

LiSt content of a directory `ls`

```
$ ls
brother-HLL6200DW-cups-en.ppd  Desktop  Documents  Downloads  Music
 ↪  Pictures  Public  Templates  Videos  Workspace
```

MaKe a subDIRectory `mkdir`

```
$ mkdir hello-world
$ ls
brother-HLL6200DW-cups-en.ppd  Desktop  Documents  Downloads
 ↪  hello-world  Music  Pictures  Public  Templates  Videos  Workspace
$ ls -F
brother-HLL6200DW-cups-en.ppd  Desktop/  Documents/  Downloads/
 ↪  hello-world/  Music/  Pictures/  Public/  Templates/  Videos/
 ↪  Workspace/
```

Basic Navigation Commands III

Change Directory `cd`

```
$ cd hello-world/
$ ls
$ cd ..
```

ReMove a subDIRectory `rmdir`

```
$ rmdir hello-world/
$ rmdir hello-world/
rmdir: failed to remove 'hello-world/': No such file or directory
```

Remark (Rule of Silence)

Commands will not print unnecessary output.

Basic Navigation Practice

Demo – We will be using a jslinux virtual machine in your (Chrome) browser, either:

- jslinux - Fedora33 [Try this first]
- jslinux - Buildroot [backup]

Useful commands:

- Print current Working Directory `pwd`
- LiSt content of a directory `ls`
- MaKe a subDIRectory `mkdir`
- Change Directory `cd`
- ReMove a subDIRectory `rmdir`

In-lecture Activity (10 mins)

Create a folder 'hello-world' in the '/mnt' directory and delete the folder. Then figure out what is 'ls -l' displaying.

Section 4: **File Management**

Output of `ls -l`

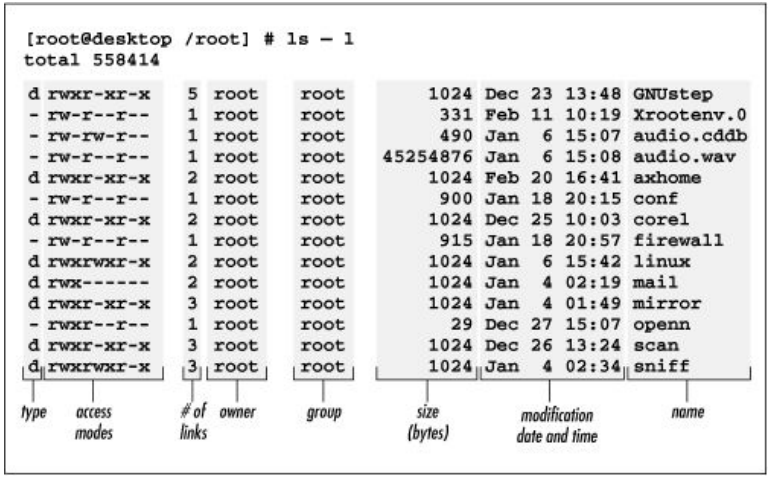

Figure 9: Image taken from (McCarty 2017)

Output of `ls -l`

- Type: Directory(`d`), File(`-`), Link(`l`)
- Access Modes (User, Group, Others)
  - User: Read Write Execute
  - Group: Read Write Execute
  - Others: Read Write Execute
- Number of Links
- Owner, eg. ehan
- Group, eg. staff
- Size (bytes)
- Modification date and time
- Name

Remark (Links)

Links in linux is similar (but different) to shortcuts in Windows.

<permission>	effect on file	effect on directory
r	reading the content of a file	read the names of the files in the dir
w	writing into a file	create/delete/rename files in the dir
x	executing a file	access contents and meta-info of dir

Changing File Permissions `chmod <mode><action><permission> <name>`

`<mode>` (`u`)ser, (`g`)roup, (`o`)thers, (`a`)ll, with `<action>` `+`, `-`
Example: `$ chmod u+w hello.c`

Remark (<>)

Don't key <>; it is common to use <> to specify an item!

File Management Commands I

CoPy `cp`

```
$ cp hello.c hello1.c
$ ls
bench.py  hello1.c  hello.c
```

MoVe / rename `mv`

```
$ mv hello1.c hello2.c
$ ls
bench.py  hello2.c  hello.c
```

File Management Commands II

ReMove `rm`

```
$ rm hello2.c
$ ls
bench.py  hello.c
```

CATenate file to screen `cat`

```
$ cat hello.c
/* This C source can be compiled with:
   gcc -o hello hello.c
*/
#include <stdlib.h>
...
```

Executables

In windows, executable binaries (usually) end with `.exe` but on linux it doesn't.

1. Create our own executable `$ gcc -o hello hello.c`. (Don't worry about this)
2. Check the executable created:

```
$ ls -l
total 148
-rw-r--r-- 1 root root    114 Dec 26  2020 bench.py
-rwxr-xr-x 1 root root  11656 Jul 12 16:27 hello
-rw-r--r-- 1 root root    185 Sep  9  2018 hello.c
```

3. Notice that `hello` is green and the eXecute permission is enabled.
4. Run the binary (Notice that `./hello` works but not `hello`):

```
$ ./hello
Hello World
```

File Management Practice

Useful commands:

- Print current Working Directory `pwd`
- LiSt content of a directory `ls`
- Change Directory `cd`
- CoPy `cp`
- MoVe / rename `mv`

In-lecture Activity (5 mins)

Copy the 'ls' binary from '/bin/ls' to your home folder. Rename the binary to 'myls' and use 'myls' to list files in your home directory to examine the modification dates.

Section 5: **Terminal Basics**

Control Sequences

Special (key) commands to control programs (sorted by frequency):

- Ctrl + d Exit/end of input to a program
- Ctrl + c Terminate current program
- Ctrl + z Suspend current program
  - `fg` Bring suspended program into foreground
  - `bg` Bring suspended program into background

Check `stty` for others, but these 3 are most used.

Remark (Terminate vs Suspend)

Ctrl + c vs Ctrl + z are fundamentally different! Do not use suspend to terminate!

Equivalent to ⊞ Ctrl + Alt + Del. but on Linux:

- `ps` List all processes
- `jobs` List all jobs - processes by current shell
- `fg` Bring suspended program into foreground
- `bg` Bring suspended program into background
- `kill` Send (kill) signal to a process
  - `pkill` Kill processes by name.
  - `kill` Kill processes by pid (Process ID).

Command Documentation

MANual `man`

The interactive manual for `ls`, `q` to quit.

```
$ man ls
```

INFOmation `info`

Alternative to `man` but not widely adopted

```
$ info ls
```

Help command `help` **or Flag** `-h`, `--help`

Many commands have implemented the help flag:
`ls --help` or `help ls`

Remark (There is always Google)

You can also Google the command to check its documentation.

Name Completion

Completing Commands

 when typing a command will autocomplete / list commands that matches:

```
$ a
addgroup   adduser    ar          arping      ash
addr2line  animate    arp         as          awk
```

Completing File Path

 when typing a file path will autocomplete / list paths that matches:

```
$ ls hello.
hello.c   hello.js
```

There are several ways to compose commands, the main ways is via `;` and `&&`.

Sequential Command Execution `;`

```
# cd /mnt; mkdir hello-world; ls; rmdir hello-world; cd ~
hello-world
...
```

AND Short-circuit Execution `&&`

Run if the previous command is successful:

```
# cd /hello-world && ls
sh: cd: can't cd to /hello-world
```

Basic I/O Redirection

Output Redirection to file `>`

```
$ ls > out.txt
$ cat out.txt
```

Input Redirection from file `<`

```
$ cat < out.txt
bench.py
hello.c
out.txt
```

Output to Input for another program `|`

```
$ ls | cat
```

Practice on `sleep` and `echo`

What is the difference between the 2 commands?
Try Suspending.

Command 1

```
$ sleep 10m && echo "Hello World" | cat > hw.txt
```

Command 2

```
$ sleep 10m ; echo "Hello World" | cat > hw.txt
```

In-lecture Activity (10 mins)

Find out what the commands ‘sleep’ and ‘echo’ does, and answer the question.

Section 6: **Vim Editor**

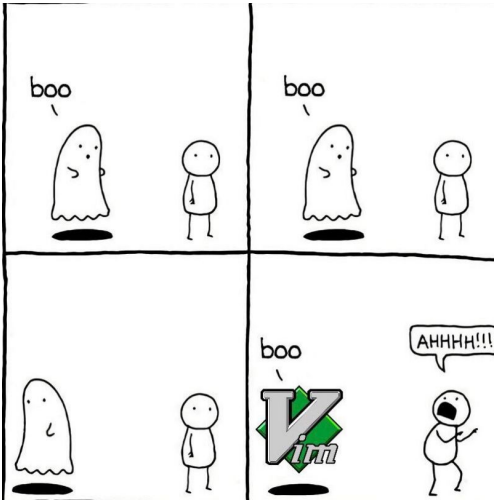

Figure 10: Image taken from devRant (devRant 2018)

Philosophy of Vim

We will not comment on the other *respectable* text editor `emacs` but focus on `vim`:

1. Minimizing Hand Movements (ie. lazy..)
2. Multi-modal Editor
   a. Normal mode: where you navigate, read / invoke other modes.
   b. Insert mode: where you insert the text
   c. Visual mode: where you highlight and manipulate text. (Not covered)
   d. Command mode: where you issue commands.
3. Tell `vim` What You Want To Do; Don't Do It Yourself
4. Be A Good Unix Citizen

Remark (Vim Advice)

Learn the smallest possible subset of vim well then moving on to more complex features.

Normal mode

Navigation

- LDUR: ← , ↓ , ↑ , → – h , j , k , l
- Word: previous, next – b w
- Sentence: forward, backward – ( )
- Paragraph: forward, backward – { }

Search

Search for a particular keyword: / , ? followed by the keyword; n for the next match.

Undo/Redo

u to undo, Ctrl + r to redo.

Going to/fro Normal mode

Quit vim

To exit vim, ⇧+z+z to save and quit; ⇧+z+q to quit without saving.

Other modes

Return to Normal mode: Esc.

- Insert mode `-- INSERT --` i, s, a, o
- Visual mode `-- VISUAL --` v
- Command mode `:` :
- Macro recording mode `recording @<a-z>` q+a-z
  - ie. q+a – `recording @a`
  - q to return to normal mode

Command mode

- Turn on line numbers `:set number`
- Jump to line number `:<line number>`
- Open another file `:e <filename>`
- Help `:help <keyword>`
- Save file `:w`
- Save and quit `:wq`
- Quit without saving `:q!`
- Delete current line `:d`
- Delete current line `:<start>,<end>d`
- Run a command and insert the std output `:r! <command>`

Insert mode

There are different ways to enter insert mode:

- i – insert before cursor
- I – insert at start of line
- a – insert after cursor
- A – insert at end of line
- o – add new line below cursor
- O – add new line above cursor
- s – substitute character
- S – substitute line

Subset I personally use regularly

- LDUR: ←, ↓, ↑, →
- u to undo, Ctrl + r to redo.
- i – insert before cursor
- Turn on line numbers `:set number`
- Jump to line number `:<line number>`
- Delete current line `:d`
- Save and quit `:wq`
- Quit without saving `:q!`
- Search ? followed by the keyword; n for the next match

Remark (Vim Advice)

Ignore the overwhelming list of shortcuts and learn a subset that you can quickly use!

Edit Source Practice

```c
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello World!!!");
    return 0;
}
```

In-lecture Activity (10 mins)

Edit 'hello.c' to as shown. Navigate to edit the line(s) via 3 ways: (1) use line number, (2) use search, (3) use LDUR. After every way, revert to original, save only on the last edit.

Section 7: **Connect to Remote Server**

What is SSH?

Intuition (Remote Desktop): It is like TeamViewer but (mainly) for CLI.

Intuition (Layman): It is like astral projection (ie. Luke Skywalker vs Kylo Ren in SW.)

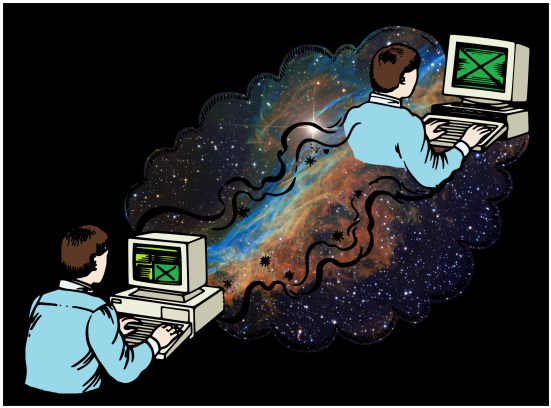

Figure 11: Image taken from Reddit (ProgrammerHumor 2022)

SSH Command

On your *local* computer, connecting to a *remote* computer:

```
$ ssh <username>@<hostname>
```

- `<username>` is your *remote* username
- `<hostname>` is the address or hostname of the *remote* computer (server/host).

Here is an example of `ehan` connecting to `pe112.comp.nus.edu.sg`.

```
$ ssh ehan@pe112.comp.nus.edu.sg
The authenticity of host 'pe112.comp.nus.edu.sg (192.168.48.114)' can't
 ↪  be established.
ED25519 key fingerprint is
 ↪  SHA256:tLCvBMgrxecaHKi1vHefU8lRH7r2PvIINmjqSVFAz68.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
 ↪  yes
```

SSH from outside SoC

The PE Hosts are only accessible in SoC networks, if you are outside:

```
$ ssh ehan@pe112.comp.nus.edu.sg
ssh: Could not resolve hostname pe112.comp.nus.edu.sg: Name or service
 ↪  not known
```

However, you may connect into SoC networks from outside by using SoC VPN.

Remark (SoC VPN vs NUS VPN)

NUS VPN gets you into NUS networks, not SoC's. So, use SoC VPN.

Access SSH?

1. – Linode Guide
   - » Recommended Command Prompt (or PowerShell)
   - » Windows Subsystem for Linux
   - » PuTTY
2. / – `ssh` via Terminal/Console

Access an Unix Terminal?

1. – Depending on your objective/constraints.
   - » Follow Access SSH guide to connect to a remote host.
   - » Windows Subsystem for Linux

Remark (jslinux virtual machine)

Use jslinux for learning purposes, do not use it for your PE assignments.

Copying to/from local/remote

We can copy (`cp`) files to/from local/remote: `$ scp <source> <destination>`

`<source>` / `<destination>` is the location of the resource:

- If local, `<path>`, same as `cp`
- If remote, it will be of the form `<username>@<hostname>:<path>`

Local, Push File to Remote

```
[eric_vader@thinkpad:~]$ scp hw ehan@pe112.comp.nus.edu.sg:~
```

Local, Pull File from Remote

```
[eric_vader@thinkpad:~]$ scp ehan@pe112.comp.nus.edu.sg:hw ./hw2
```

Remote, Pull/Push to/from Local

```
ehan@pe112:~$ scp eric_vader@<local-hostname>:hw ./hw3
```

Demo of Objective

Recap of what we learnt

1. Create a file using `vim`
2. Transfer to remote using `scp`
3. Remote connect to remote using `ssh`
4. Compile a program on remote
5. Run the program and pipe output to a file
6. Transfer the output to local using `scp`
7. Open locally using `vim`

Some Improvements (Advanced)

Read/Watch and learn more:

1. Setting up SSH Keys
2. Use `screen` to create terminal 'tabs'

Post Lecture Tasks

This lecture is based on; read in more detail:

- NUS SoC Unix@Home Workshop, 2021
- CS1010 Lab Guides Documentation

Setting Up for Class

1. Apply for SoC Unix account
2. Enable access for the compute cluster
3. Setup SoC VPN - enable VPN when outside SoC
4. How to access SSH - Slide 58

References I

Brys, Carlos. 2012. https://www.slideshare.net/slideshow/datos-abiertos-opendata/14913843.

chococigar. https://www.reddit.com/media?url=https%3A%2F%2Fpreview.redd.it%2Fe21s2vb3a2a81.png%3Fwidth%3D4613%26format%3Dpng%26auto%3Dwebp%26s%3Da2bc24827b9caf90eb0c85dae83893976a6d8183.

devRant. 2018. "DevRant - Every Non Vim Users (or Emacs!) — Devrant.com." https://devrant.com/rants/1607005/every-non-vim-users-or-emacs.

Guillem, Hotmocha, Wereon. 2021. "UNIX - Simple English Wikipedia, the Free Encyclopedia — Simple.wikipedia.org." https://simple.wikipedia.org/wiki/UNIX.

Linux Inside. 2017. "Log into Facebook — Facebook.com." https://www.facebook.com/photo/?fbid=1832237733461345&set=a.781153315236464.

McCarty, Bill. 2017. "Learning Debian GNU/Linux, 1999, o'rilley."

ProgrammerHumor. 2022. “How I Explain SSH to People That Don’t Have a Software Background.” https://www.reddit.com/r/ProgrammerHumor/comments/y20xpa/how_i_explain_ssh_to_people_that_dont_have_a/.

Salus, Peter H. 1994. *A Quarter Century of UNIX*. ACM Press/Addison-Wesley Publishing Co.

Stack Overflow. 2013. “Stack Overflow Developer Survey 2023 — Survey.stackoverflow.co.” https://survey.stackoverflow.co/2023/.