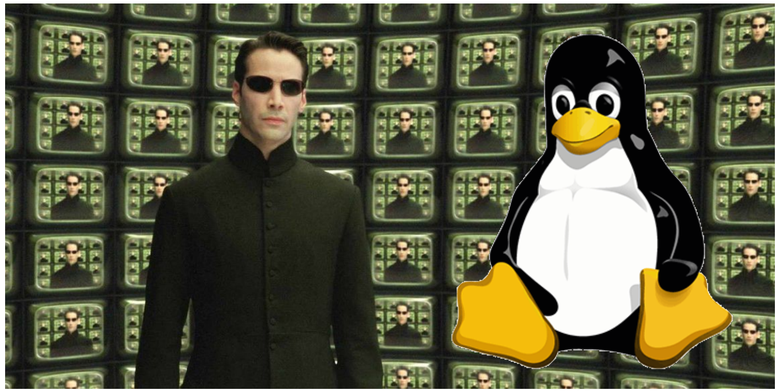
26.05.20 10:52

CSCI 125
Introduction to Computer Science and Programming II
Lecture 0: Introduction to Linux



Jetic Gū

2020 Summer Semester (S2)

Overview

- Focus: Introduction to Linux
- Architecture: Linux/Unix OS
- Core Ideas:
  1. Introduction to Linux
  2. Common commands
  3. Lab0: `vim`, `"Hello World!"`, Compiling a programme, submitting on OJ

Linux/Unix

Yes, they are different, and
it is recommended that you turn on your linux now

Summary

Popular Computer OS

- Windows NT
  - Windows XP, Windows 7, 8, 10
- Unix
  - BSD, OSX, macOS, iOS
- Linux
  - Ubuntu, Fedora, Android[1], etc.

1. Modified Linux kernel

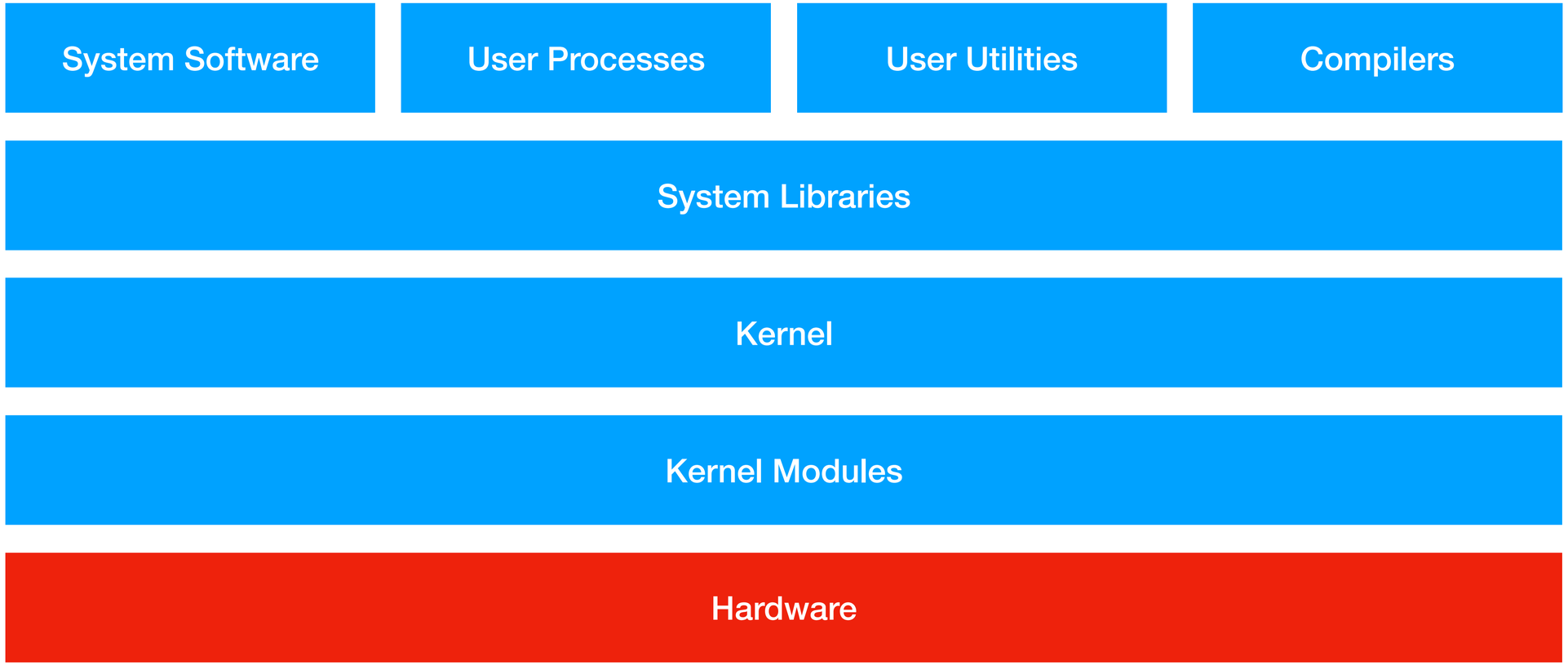
Linux

- Linus Torvalds’ Undergraduate Thesis
  - he was frustrated by Unix license issues
- Open-source Unix-like OS
- Until 2018, the kernel is maintained by Torvalds alone

Concept

1. Modified Linux kernel

Linux Architecture



1. More Knowledge: Operating System Course (SFU: CMPT300; UBC: CPEN331; BCIT: COMP4731)

Concept

File System

Windows

Letter assignments of drives

- C:\
- D:\
- ...

Linux/Unit

Root drive

- Root: `/`
- Others: `/mnt/usb1`

Concept

Linux/Unix File Structures

- `/`: root directory
- `/bin`: essential executables (commands)
- `/sbin`: essential system programmes
- `/dev`: physical devices
- `/etc`: system configuration files
- `/lib`: essential libraries
- `/tmp`: temporary stuff
- `/usr`: Secondary hierarchy

- Linux
  - `/home`: User folders
    e.g. `/home/jetic`
- OSX/macOS
  - `/Users`: User folders
    e.g. `/Users/jetic`

Users and User Groups

- Most powerful user: `root`
- Superuser: `jetic` (user with administrative privileges)
- Normal user: `cocoa` (user without administrative privileges)

Concept

Permissions

- 2 Attributes: Owner, Group
  - Any file/folder must have an Owner(User), and a Group(User Group)
- 3 permission types:
  - Read/Write/Execute, expressed by 3 binary bits (e.g. $(111)_2=7$)
- 3 permission categories:
  - Owner, Group, Everyone else

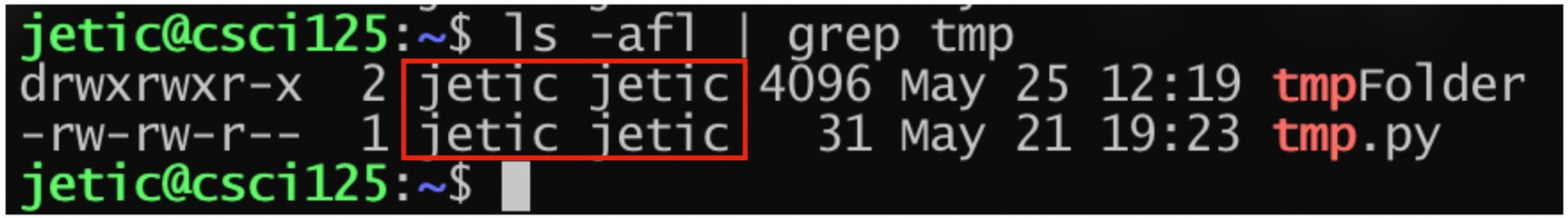
Concept

Permissions

```
jetic@csci125:~$ ls -afl | grep tmp
drwxrwxr-x  2 jetic jetic 4096 May 25 12:19 tmpFolder
-rw-rw-r--  1 jetic jetic   31 May 21 19:23 tmp.py
jetic@csci125:~$
```

- "-rw-rw-r--"
  first character: [-, d], '-' for file, 'd' for folder (directory)
  Owner "rw-" read+write;
  Group "rw-" read+write;
  Others "r--" read-only;
  Also written as 110110100 or 664

Permissions



- first "`jetic`": owner
- second "`jetic`": group

Linux/Unix Commands

Just the common ones for now

Summary

P2
Command Line

Command Line Environment

This is also called `shell`

- `jetic@csci125:~ $`
  - `jetic`: username
  - `csci125`: computer name, also know as `HostName`
  - `~`: Current directory, '`~`' stands for the home directory for current user
    as in `/home/jetic` in Linux or `/Users/jetic` in mac
  - `$`: current user is not root
    for root it's '#'

Technical

P2
Command Line

1. Software Installation

- Linux comes with different **Package Managers**, sorta like App Stores
  - Ubuntu uses **APT**
  - Reference manual: `$ man apt`

`man` stands for manual, most commands have such things

Technical

P2
Command Line

1. Software Installation

- APT maintains a "list" of all software locally, if you want to install something, it will look at the list for it.
  - Update local APT "list":
    `$ sudo apt update`
  - Upgrade all installed software:
    `$ sudo apt update`
  - Install something
    Install vim: `$ sudo apt install vim`
    Install C compiler: `$ sudo apt install build-essential`
    Install C++ compiler and debugger: `$ sudo apt install g++ gdb`

`sudo` stands for "superuser do", only users with administrative privileges can execute these commands. Password also required, and `shell` remembers your password for a while.

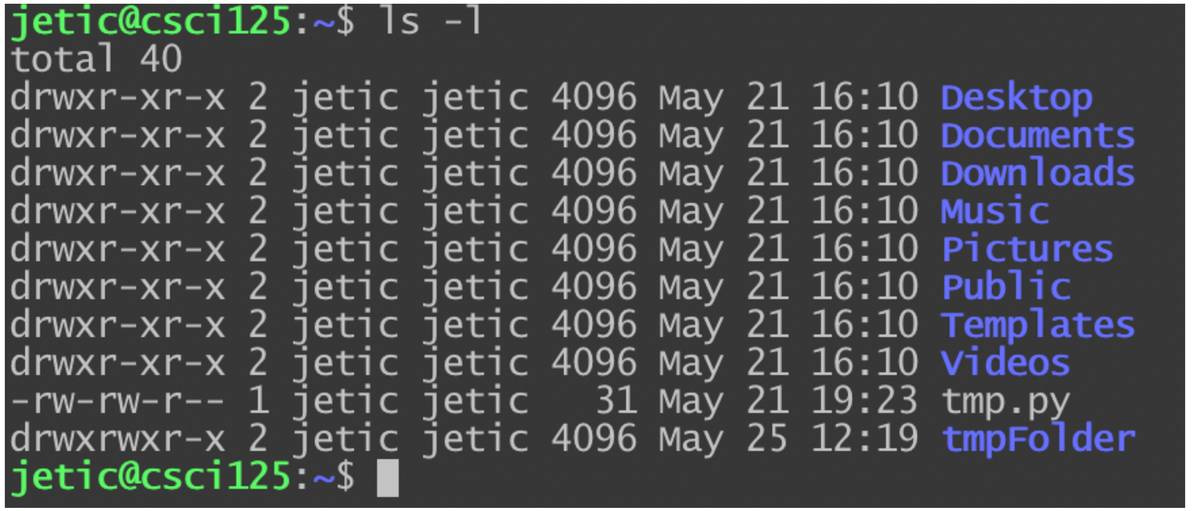
Technical

2. Change Directory

- Command `pwd`
  - Working directory: everything you do is going to be w.r.t. this directory (e.g. create a new file)
  - Show current working directory:
    ```
    $ pwd
    ```
- Command `cd`
  - Relative: go into a folder named `Desktop`:
    ```
    $ cd Desktop      # in the current working directory
    $ cd ./Desktop    # in the current working directory
    $ cd ../Desktop   # in the parent working directory
    ```
  - Go into a folder named `Desktop` in your home:
    ```
    $ cd /home/jetic/Desktop
    $ cd /Users/jetic/Desktop
    $ cd ~/Desktop
    ```

3. List Directory

- Command `ls`
  - List everything in current directory:
    ```
    $ ls
    $ ls .
    ```
  - List everything in some directory:
    ```
    $ ls /usr
    ```
- Options for Command `ls`
  - All (include hidden files):
    ```
    $ ls -a
    ```
  - Long format:
    ```
    $ ls -l
    $ ls -al
    ```



4. Make Directory

- Command `mkdir [FOLDER] ...`
  - Create a **folder** named `myFirstFolder`:
    ```
    $ mkdir myFirstFolder
    ```
  - Create a **folder** named `my First Folder`:
    ```
    $ mkdir my\ First\ Folder
    ```

5. Delete Stuff

- Command `rm [OPTION] ... [FILE] ...`
  - Delete a **file** named `tmp.cpp`:
    ```
    $ rm tmp.cpp
    ```
  - Recursively delete a **folder** named `tmpfolder`:
    ```
    $ rm -r tmpfolder
    ```
  - Delete any **file** matching pattern `../tmp.*`:
    ```
    $ rm ../tmp.*
    ```
- Command `rmdir [FOLDER] ...`
  - Delete an empty **folder** named `tmpfolder` (warning if not empty):
    ```
    $ rmdir tmpfolder
    ```

6. Move stuff

- Command `mv SRC TGT`
  - Rename `tmp.cpp` to `tmp1.cpp`:
    ```
    $ mv tmp.cpp tmp1.cpp
    ```
  - Move `tmp.cpp` to folder `~/cheese`:
    ```
    $ mv tmp.cpp ~/cheese
    ```
- Copy file `cp [OPTION] SRC TGT`
  - Copy file `tmp.cpp` to `another tmp.cpp`:
    ```
    $ cp tmp.cpp another\ tmp.cpp
    $ cp tmp.cpp "another tmp.cpp"
    ```

7. Cat

- Command `cat [FILE] ...`
  - Display a **file as text** named `tmp.cpp`:
    ```
    $ cat tmp.cpp
    ```

8. Download

- Command `wget URL`
  - Install wget using APT
  - Download a vim configuration file:

```
$ wget jetic.org/download/vimrc
$ wget --no-check-certificate https://jetic.org/download/
vimrc
```

Exercise

- List everything under `/etc`
- List in long format everything under the root directory `/`
- Create a new folder called `csci125lab0` under your home directory
- Download https://jetic.org/download/vimrc
  - Look at its content using `cat`
  - Put it in your home directory, and rename it as `.vimrc`

Hello World!

How to `Vim` it?

Actual Code In Python

```
if __name__ == '__main__':

    print("Hello World!\n")
```

- Indentation is important: not in any C or extension of C (C++) though
- Main programme: the whole thing is the main programme

Concept

Actual Code In C++

```
#include <cstdio>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

- Indentation not important: you can remove all "\n" and make it a single line
- `#include <cstdio>`: import library `cstdio`, which is C standard IO library supplying you `printf` function.

Actual Code In C++

```
#include <cstdio>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

- `int main() { ... return X;}`: function definition
  `main()` function must be of type `int`, must have return value

- `0` from `main()` means the programme exits normally

Actual Code In C++

```
#include <cstdio>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

- `printf("Hello World!\n");`
  - Same as `print("Hello World!\n")` in python
  - Every function call ends with `';'`
  - Apostrophe `'x'` for characters
    Double quotes `"xxx"` for strings (character sequences)

Concept

P3
Hello World

Typing the code in Vim

```
#include <cstdio>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

- `$ vim lab0p1.cpp`
  - Press key `'i'` to enter interactive mode, so that you can type
  - Press key `'esc'` to exit interactive mode
    - enter "`:w`" to save
    - enter "`:q`" to quit, or "`:wq`" to write and save, or "`:q!`" to force quit

Concept

Compile

```
#include <cstdio>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

- Go back to SHELL
  - Compile using `g++ $ gcc lab0p1.cpp -o lab0p1`
  - Execute the programme `$ ./lab0p1`