Computer Systems and Networks

ECPE 170 – Instructor Dr. Pallipuram– University of the Pacific

Linux Basics

These slides are credited to Dr. Jeffrey Shafer

Pre-Lab

↗ Everyone installed Linux on their computer

↗ Everyone launched the command line ("terminal") and ran a few commands

↗ **What problems were encountered?**

  ↗ **Virtualization support in processor not enabled (BIOS)**

  ↗ VMWare Player (current version) only runs on Windows 64

  ↗ 3D graphics virtualization incompatible with specific hardware

  ↗ Old virtual machine software

  ↗ Others?

↗ Tip: If you have problems maximizing your VM to full screen, or doing copy-and-paste between Linux and Windows, make sure you installed the VM tools

Person of the Day: Linus Torvalds

↗ Creator of **Linux Kernel**

  ↗ Started in 1991

  ↗ First developer – hobby project (for fun!)

  ↗ Modern kernel is product of work by thousands of programmers

  ↗ Currently "final authority" on what is included in the kernel

↗ Creator of **Git version control system**

  ↗ Initially for Linux kernel dev

Operating System Tasks

➚ **What does the OS need to do?**

➚ Schedule processes to run

➚ Memory management

➚ Interrupt handling (manage hardware in general)

➚ Security (between processes)

➚ Network access

➚ Storage management (filesystem)

➚ Graphical user interface

➚ May be a **middleware** layer on top of the OS

Operating Systems – Processes

➔ **Process management** is a key operating system task

➔ OS must initially **create processes** when you run your program

➔ OS can allow processes to **access resources**
  ➔ Must *schedule* access to *shared* resources (e.g., CPU)

➔ OS can allow processes to **communicate** with each other

➔ OS must **clean up** after process finishes
  ➔ Deallocate resources (e.g. memory, network sockets, file descriptors, etc...) that were created during process execution

Operating Systems – Scheduling

↗ The operating system schedules process execution

　↗ What processes are allowed to run <u>at all</u>?

　↗ What processes are allowed to run <u>right now</u>?

↗ **Context switches** occur when the CPU is taken from one process and given to another process

　↗ CPU *state* (registers, current PC, etc…) is preserved during a context switch

Operating Systems – Scheduling

➚ **Preemptive Scheduling**

➚ Each process is allocated a timeslice

➚ When the timeslice expires, a context switch occurs

➚ A context switch can also occur when a higher-priority process needs the CPU

Discuss

What to expect from OS w.r.t Security when there are several Processes?

Operating Systems – Security

↗ Process A is forbidden from reading/modifying/writing the memory of Process B

  ↗ **Virtual memory** is a huge help here!

  ↗ Each process has a separate *virtual* address space that maps to different regions of *physical* memory

↗ Process A has other limits besides which memory pages it can access

  ↗ **What are some other limits?**

  ↗ Amount of memory consumed

  ↗ Number of open files on disk

  ↗ Which files on disk can be read/written

Operating Systems – Filesystem

➚ OS is responsible for managing data on persistent storage

➚ Job of the **filesystem!**

   ➚ What files exist? (i.e. names)

   ➚ How are they organized? (i.e. paths/folders)

   ➚ Who owns and can access them? (i.e. usernames, permissions)

   ➚ Where are individual file blocks stored on the disk?

      ➚ *i.e. filename "database.dat" is really composed of 15823 blocks, of which block 1 is located at logical block address #... on the hard drive.*

Operating Systems – Device Management

- ↗ Manage devices
  - ↗ How do we send data to the NIC for transmission?
  - ↗ How do we render an image for display on screen?
  - ↗ How do we read a block of data from our RAID disk controller?

- ↗ Operating systems can be extended through **device drivers** to manage new hardware
  - ↗ Hardware vendors write software to manage their devices
  - ↗ OS provides a fixed interface (API) that driver must follow

- ↗ Common task for a device driver is **responding to interrupts** (from that device)

Operating Systems – The Kernel

- Who does all this essential work in the operating system? (besides the GUI)
    - The **kernel** (i.e. the heart or core of the OS)

- Kernel performs:
    - Scheduling
    - Synchronization
    - Memory management
    - Interrupt handling
    - Security and protection

Operating Systems – GUI

➚ Operating systems with **graphical user interfaces** (GUI) were first brought to market in the 1980s

Apple Mac OS 1.0 (released 1984)
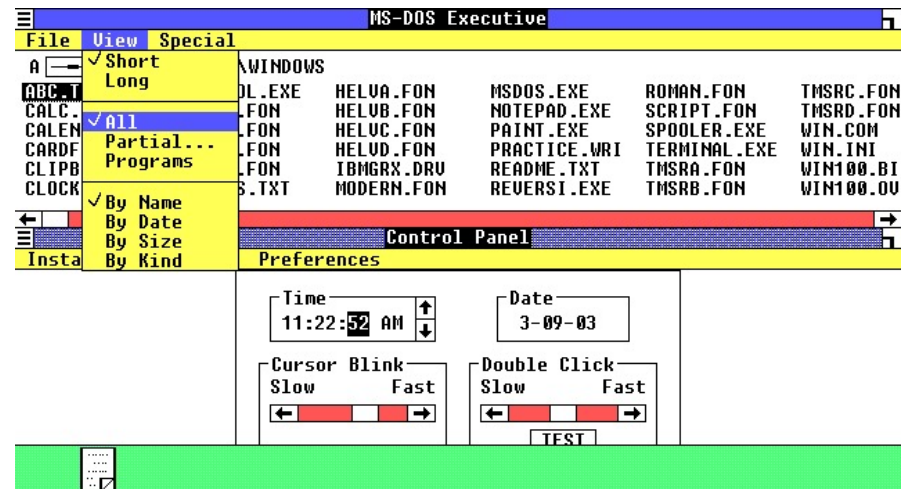
Microsoft Windows 1.0 (released 1986)

Captures from http://www.guidebookgallery.org/screenshots

↗ Significant evolution in GUI design in subsequent decades

Operating Systems – GUI

→ Technical perspective:

    → The GUI is one of the **least important parts** of the operating system

→ A GUI does not even have to be part of the *true* OS at all

    → Windows 1.0 was just a **program that ran on top** of MS-DOS, the *true* operating system (of that era)

→ *But to a user, the GUI is one of the most important parts of the OS!*
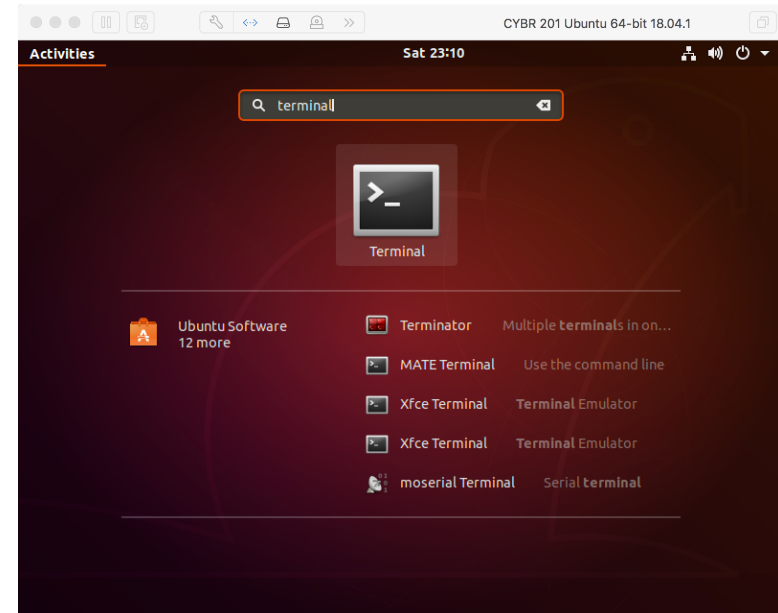
Command-Line

Advantages of
Command Line

Advantages of
Windows / GUI

Linux Command Line

In-Class Activity

1. Launch your Linux virtual machine!

2. Open the **Terminal** – a text-based interface that accepts your commands (Applications button -> Terminal)

3. Open Canvas and today's *In-Class Participation* assignment
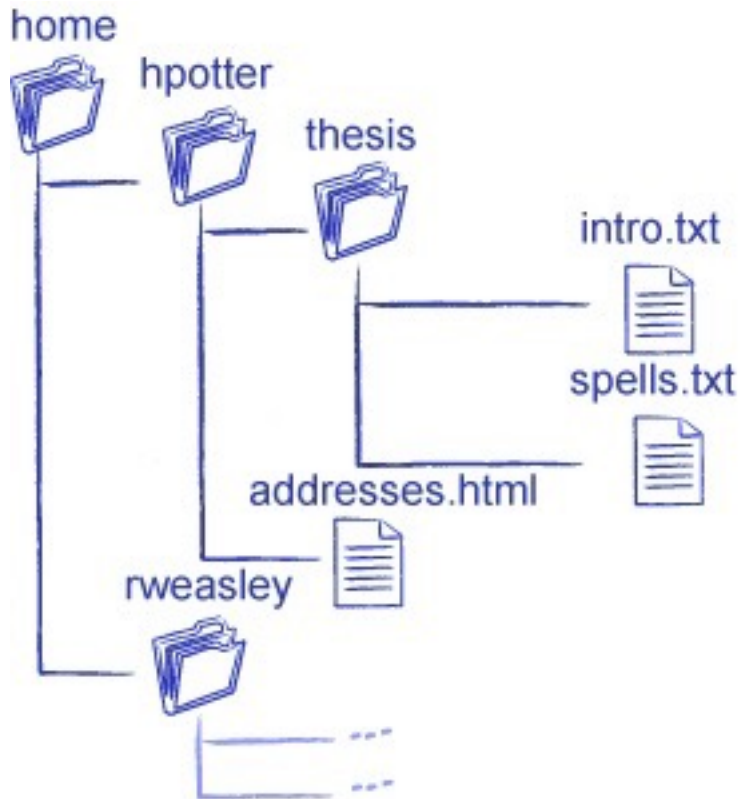
Problem 1 – Which Shell?

↗ A shell is a user program that defines how your terminal window behaves for input commands

   ↗ Command-line interpreter

   ↗ Parses user input and carries out commands

↗ Many types exist: `sh`, `bash` (Bourne again), C syntax motivated: `csh`, `tsh`, etc.

↗ Find out what shell is being used:

```
$ echo $SHELL
```

P1

Directory Structure

home
hpotter
thesis
intro.txt
spells.txt
addresses.html
rweasley

~L 2000

http://osl.iu.edu/~pgottsch/swc2/lec/shell01.html

➚ **Root directory:** /

➚ **Absolute** path:
   ➚ `/home/hpotter/thesis/intro.txt`

➚ **Relative** path:
   ➚ If I am already in `/home/potter/`
   ➚ `addresses.html`

Problem 2 – Navigation Skills

↗ Where are we?

```
$ pwd
```
Print Working Directory

↗ What items exist here?

```
$ ls
```
List items

```
$ ls [options] [location]
```

P2

Basic Operations

➚ Tilde (~) sign refers to <u>your</u> home directory. You can perform either

```
$ ls /home/you/Documents
```

```
$ ls ~/Documents
```

Navigation Skills

➔ Dot (.) sign refers to current directory. Try:

```
$ ls .
```

➔ Double dot (..) refers to the parent directory of your current directory. Try:

```
$ ls ..
```

Problem 3 – Navigation Skills

↗ Change directory

```
$ cd [location]
```

↗ Hint: There's a very easy shortcut to change directory to your home directory...

P3

Problem 4 – Documentation

↗ Documentation ("manual") on commands

```
$ man [command]
```

↗ Example usage

↗ Hidden files have a (.) before the filename

↗ `.secret`, `.bashrc`, …

↗ Type `ls` – Do you see any hidden files?

↗ Using the man command, find out what option you need to use with `ls` to list the hidden files

```
$ ls -a
```

P4

Basic Operations

Do these steps in your VM!

↗ Create a directory called `Linux_tutorial` inside your home directory

```
$ mkdir [options] [dirname]
```

↗ Change to the `Linux_tutorial` directory

↗ Create a blank file called `example1`

```
$ touch example1
```

Basic Operations

↗ Put something in the file via output redirection

```
$ echo "Tiger Roar" > example1
```

Do these steps in your VM!

↗ Copy file `example1` to `example2`

```
$ cp example1 example2
```

↗ Move the `example2` file to your home directory

```
$ mv example2 ~
```

Basic Operations

Do this step in your VM!

➚ Remove the file `example2`

```
$ rm ~/example2
```

➚ General form of command

```
$ rmdir [options] [dirname]
```

```
$ rm [options] [filename]
```

Piping

Do this step in your VM!

↗ Change to `/etc` directory and count the number of files in that directory. You only have 60 seconds. Tick tock!!

↗ Tip: Combine list tool with another tool that will count the number of words (or <u>lines</u>)

```
$ cd /etc
$ ls -l | wc -l
```

Pipe Word   Option: Count
        Count   number of lines

Problem 5 – Wildcards

↗ Directory listings can use wildcards to search for matching file names

↗ Example: In `/etc` directory, list all files with `.conf` extension

```
$ ls *.conf
```

↗ Example: In `/etc` directory, list all files where second letter is d and with `.conf` extension

```
$ ls ?d*.conf
```

* – Zero or more characters
? – Single character
[] – Range of characters

P5

File Permissions

➚ Linux provides you privacy with files via permissions

   ➚ **r** read – the contents of the file can be viewed

   ➚ **w** write – something can be written to the file

   ➚ **x** execute – the file can be executed if an executable or script

➚ Permission is granted to three types of people

   ➚ **owner** – the one who created the file, also called user (u)

   ➚ **group** – the file belongs to a single group (g)

   ➚ **others –** everyone else (o) but the group or the owner

Problem 6 – File Permissions

↗ *Create the requested file with the requested contents, and obtain a directory listing…*

```
-rw-rw-r-- 1 shafer shafer 18 Sep  4 14:40 example3
```

owner has read and write permissions, but not execute

group has read and write permissions, but not execute

others have read only permissions

P6

File Permissions

↗ The `example3` file can't be executed – try it:

```
$ ./example3
```

Problem 7 – File Permissions

```
$ chmod [permissions] [file]
```

↗ Changing the file permissions requires answers some questions

　↗ Whose permissions are we changing?

　　↗ `[ugoa]`: owner, group, others, or all

　↗ Are we granting or revoking permission?

　　↗ **+**: providing　　**-**: revoking

　↗ What are we providing?

　　↗ **r** (read), **w** (write), or **x** (execute)

```
$ chmod u+rwx file
```
Provides rd/wr/ex to owner

```
$ chmod g-x file
```
Removes ex for group

P7

Problem 8 – Wrap-up

↗ Open-ended questions based on what we've learned today

↗ *Take 5 minutes and complete…*
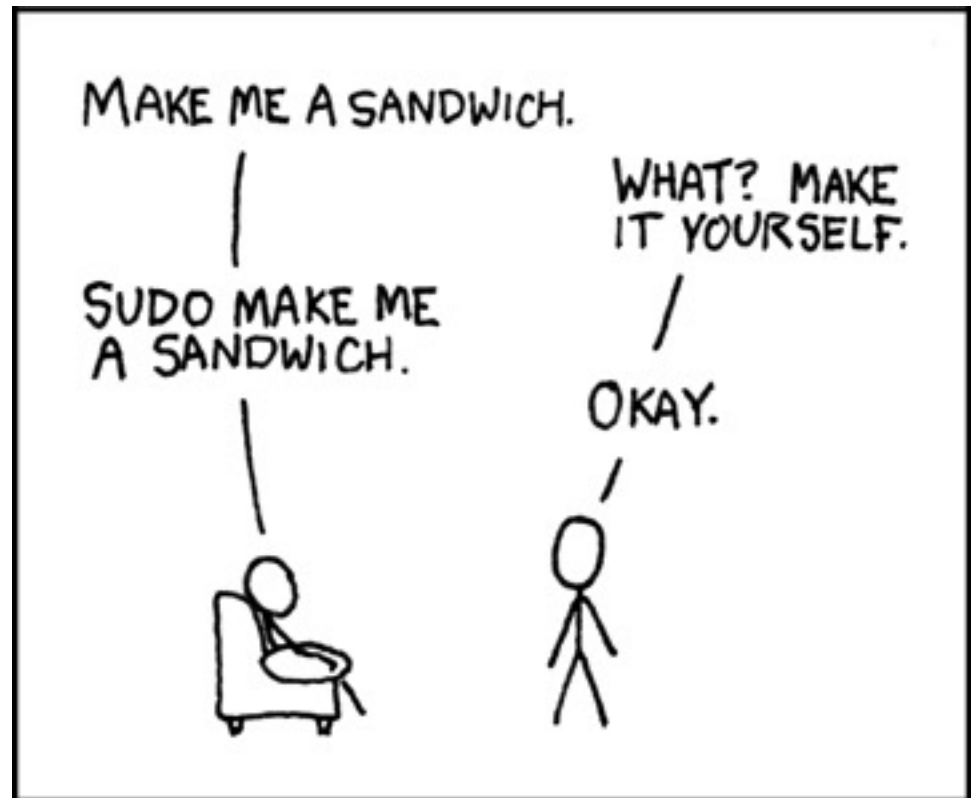
P8

Shell Shortcuts

↗ <TAB> key to auto-complete commands

↗ <UP ARROW> key to cycle through previous commands

These two tips make your life so much <u>easier</u>!

Linux: Sudo Command

↗ `sudo <<command>>`

↗ Command is run as root user

↗ root = "Administrator"

MAKE ME A SANDWICH.

SUDO MAKE ME A SANDWICH.

WHAT? MAKE IT YOURSELF.

OKAY.

http://xkcd.com/149/

Labs

Labs

↗ Labs have (at most) two graded elements:

1. **Pre-Lab "checkpoint"** – quick verification that pre-lab *appears* to be done

   1. Due at start of first day of lab

2. **Lab Report**

   1. Submit all source code used with lab report

   2. Due by posted date after lab

Lab Reports

➚ Not really "reports", more like "worksheets"

➚ Create in LibreOffice (aka *OpenOffice*) using example template on website

➚ Export in **PDF format**

➚ Submit

  ➚ Via Canvas *Assignments* section for Lab 1 <u>only</u>!

  ➚ Via Version control for Lab 2 and beyond

Upcoming Schedule

- ↗ Today
  - ↗ **Lab 1 – Linux Basics**

- ↗ Thursday
  - ↗ **Lab 2 – Version Control**

- ↗ Deadlines
  - ↗ **Lab 2 pre-lab checkpoint – Start of class Thursday**
  - ↗ **Lab 1 Report – Sep 6th, 2021 by 11:59 PM**
    - ↗ Submit via Canvas
  - ↗ **Lab 2 Report – Sep 8th, 2021 by 11:59 PM**