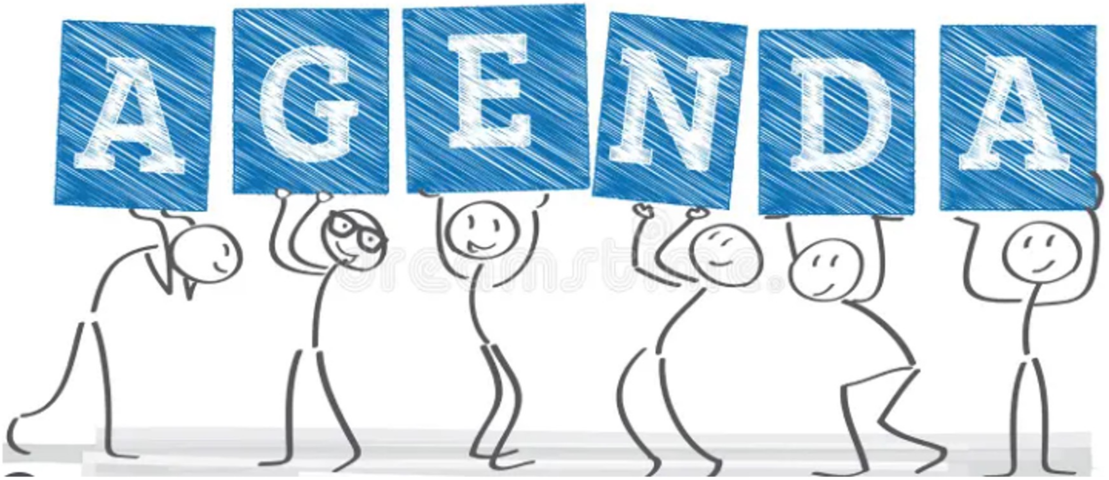
Operating Systems

Lecture 1.1

Virtual Machines and Lab Environment Setup

Lecture Agenda



- Virtualization Technology
- Overview of Linux Operating System
- Interacting with Linux OS
- Basic Linux Shell Commands
- Linux File Hierarchy Standard
- Hands-On Practice on the Shell

Virtualization Technology

(Understanding Modern Computing Infrastructure)

The Problem – Why Do we need Virtualization?

Traditional Computing Challenges:

- *Underutilized Hardware:* Physical servers run at only 15-20% capacity
- *High Costs:* Expensive hardware, power, cooling, and maintenance
- *Inflexibility:* Difficult to scale resources up or down
- *Testing Limitations:* Need multiple physical machines for different OS testing
- *Disaster Recovery:* Complex backup and restoration processes

Developer Challenges:

- Want to learn multiple operating systems but only have one computer
- Need to test software across different platforms
- Require isolated environments for safe experimentation
- Limited budget for multiple physical machines

The Solution: Virtualization

Virtualization is the process of creating virtual instances of computing resources (such as operating systems, servers, storage, or networks), allowing multiple isolated environments to run concurrently on a single physical machine.

- **Key Benefits:**
  - **Resource Optimization:** Increase hardware utilization to 70-80%
  - **Flexibility:** Easy to create, modify, and destroy virtual environments
  - **Isolation:** Problems in one virtual system don't affect others
  - **Rapid Deployment:** Create new systems in minutes, not hours

- **Real-World Impact:** Modern data centres run 80% virtualized workloads

How to Experience Multiple Operating Systems?

Option 1: Physical Network

- Multiple physical machines with different OS installations
- Pros: Maximum performance, complete hardware control
- Cons: High cost, power consumption, maintenance overhead

Option 2: Multi-Boot Systems

- Dual/triple boot on single machine with careful partitioning
- Pros: Native performance, access to multiple OSs
- Cons: Only one OS active at a time, no network simulation, complex setup

Option 3: Windows Subsystem for Linux (WSL)

- Linux environment directly in Windows 10/11
- Link: https://learn.microsoft.com/en-us/windows/wsl/about
- Pros: Native integration, lightweight, no dual boot needed
- Cons: Windows-only, limited to Linux environments

How to Experience Multiple Operating Systems?

Option 4: Desktop Hypervisors (Most Popular for Learning)

- Oracle VirtualBox (Free): https://www.oracle.com/virtualization/virtualbox/
- VMware Workstation/Fusion Pro: https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion
- Microsoft Hyper-V (Windows built-in): https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/
- Parallels Desktop (Mac): https://www.parallels.com/products/desktop/
- UTM (Mac): https://mac.getutm.app/

Option 5: Container Technology

- Docker Desktop: https://docs.docker.com/desktop/install/windows-install/
- Run Linux distributions inside lightweight containers

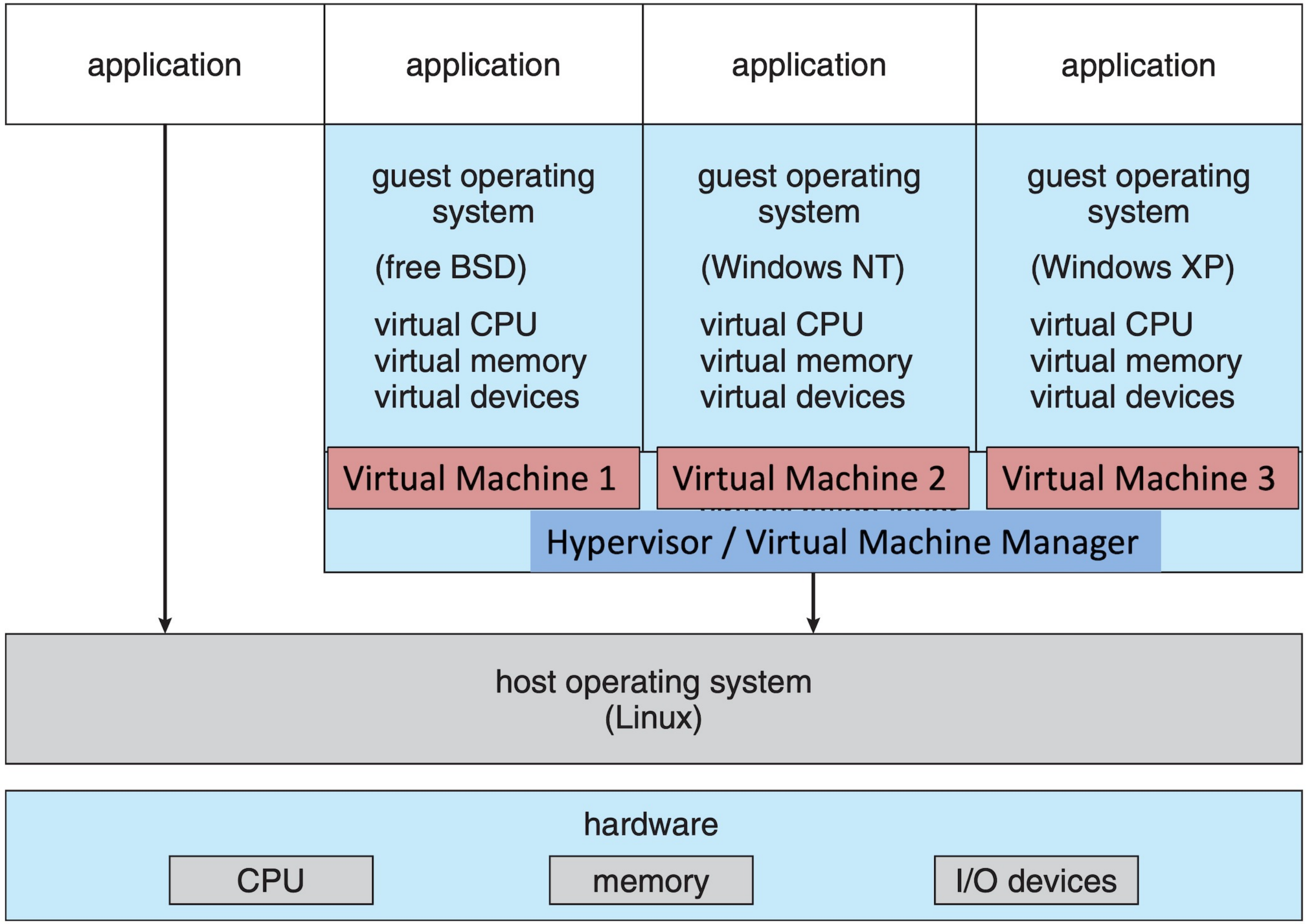
Option 6: Cloud Virtualization

- Use AWS, Azure, or Google Cloud (leading cloud computing platforms that provide on-demand services like computing, storage and networking over the internet)
- Getting Started: https://www.linkedin.com/advice/3/how-can-you-use-linux-cloud-computing-skills-system-administration

Hypervisors

A Hypervisor also known as Virtual Machine Monitor is a software layer that creates and manages virtual machines, acting as an interface between physical hardware and virtual systems. The two main types of hypervisors are:

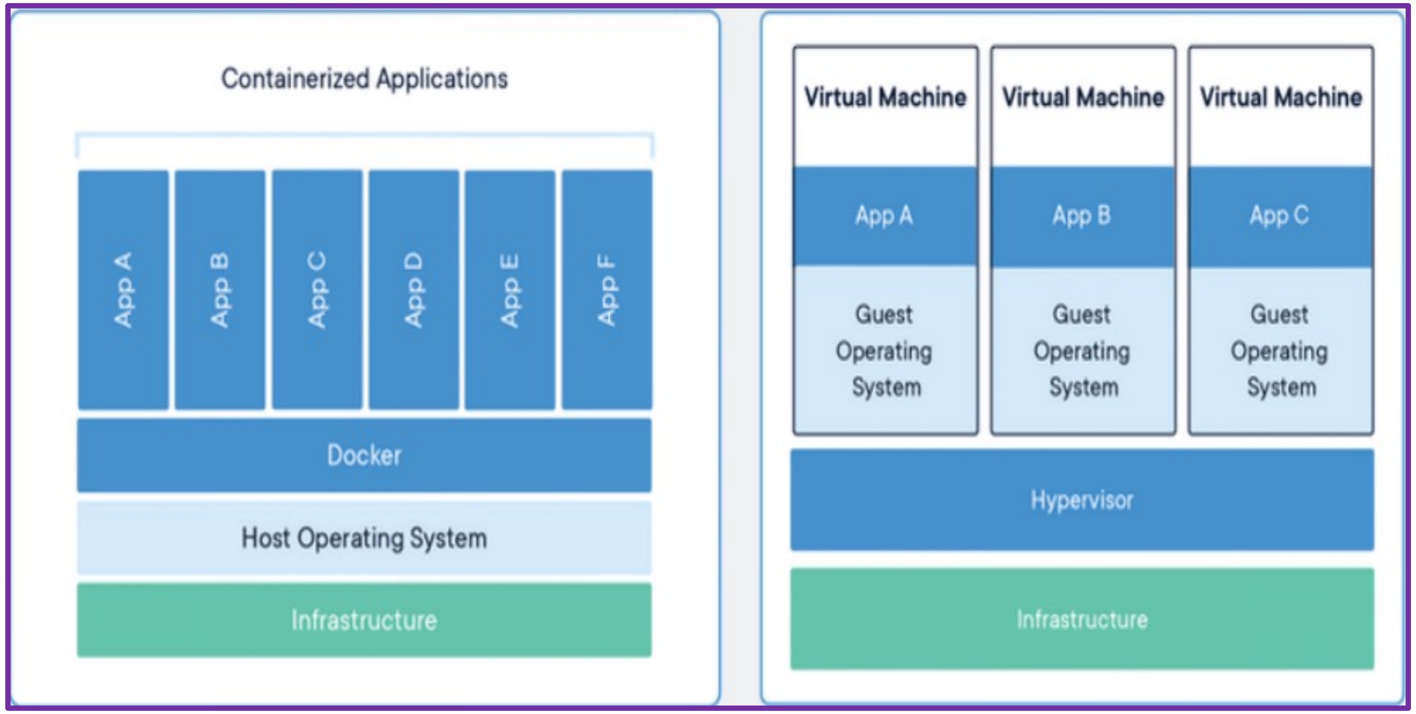
- **Type-1 (Bare-Metal):** Runs directly on hardware (VMware ESX, Citrix XenServer)
- **Type-2 (Hosted):** Runs on host OS (Oracle VirtualBox, VMware Fusion, Parallels Desktop)



For this course, install Oracle VirtualBox on your host OS (Windows 11), and as a guest OS install either Kali Linux or Ubuntu Desktop for practicing the shell commands and programming assignments.

Docker's Containers

Docker is like a smart toolkit for building, sharing, and running software in tiny, portable packages called containers. It makes sure your app works the same no matter where you run it—on your laptop, in the cloud, or on someone else's machine. With Docker, developers can package everything their app needs and ship it out smoothly, quickly, and reliably.



https://docs.docker.com/desktop/setup/install/windows-install/

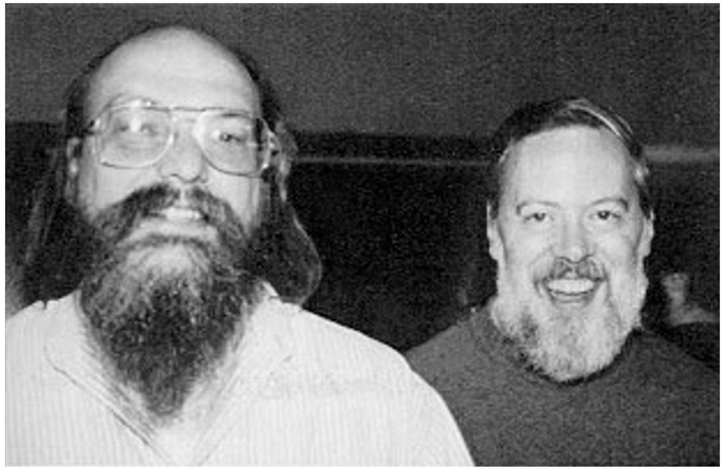
- **Docker Image:** A Docker image is like a blueprint or recipe for creating containers. It contains everything needed to run an app, including the code, tools, and settings. Containers are built from images, making them reusable and consistent.

- **Docker Container:** A Docker container is like a lightweight, portable mini-computer that runs only the apps and tools you need. It uses shared resources from your actual computer but stays isolated, so it doesn't interfere with other programs. Think of it as a neat, self-contained box for running software consistently anywhere!

Linux - The OS

History of Unix

- All modern operating systems have their roots in 1969, when Dennis Ritchie and Ken Thompson developed the C language and the UNIX operating system at AT&T Bell labs.



- Since the source code of UNIX was widely available, various organizations developed their own versions, which led to chaos as far as UNIX history is concerned.
- Two major versions developed:
  - System V, from AT&T.
  - BSD (Berkeley Software Distribution from UC Berkeley). Minor variation includes FreeBSD, OpenBSD and NetBSD.
- To make it possible to write programs that could run on any UNIX system, IEEE developed a standard for UNIX, called POSIX and later SUSv3, that most versions of UNIX now support.

UNIX is basically a Simple Operating System

But

You have to be a *GENIUS* to understand the Simplicity

~ Dennis Ritchie

The Linux

- In 1991, Linus Torvald, a student of university of Helsinki Finland, bought a 386 computer and tried to write a brand new POSIX compliant kernel, which became what we call Linux today.
- Todays Linux run on:
  - **100%** of all world's top 500 supercomputers run Linux.
  - **80%** of all smart phones.
  - 96% of the top one million **web servers** run on Linux.
  - **Millions** of desktop computers.
  - Embedded Systems (routers, Raspberry Pi boards, self driving cars, washing machines etc.)

- Source code of latest stable kernel (6.16) can be downloaded from https://www.kernel.org
- For an annotated, cross-referenced view of the Linux kernel source across versions, explore **Elixir** by **Bootlin**: https://elixir.bootlin.com/linux/v6.15.4/source a must-have tool for kernel hackers and contributors.

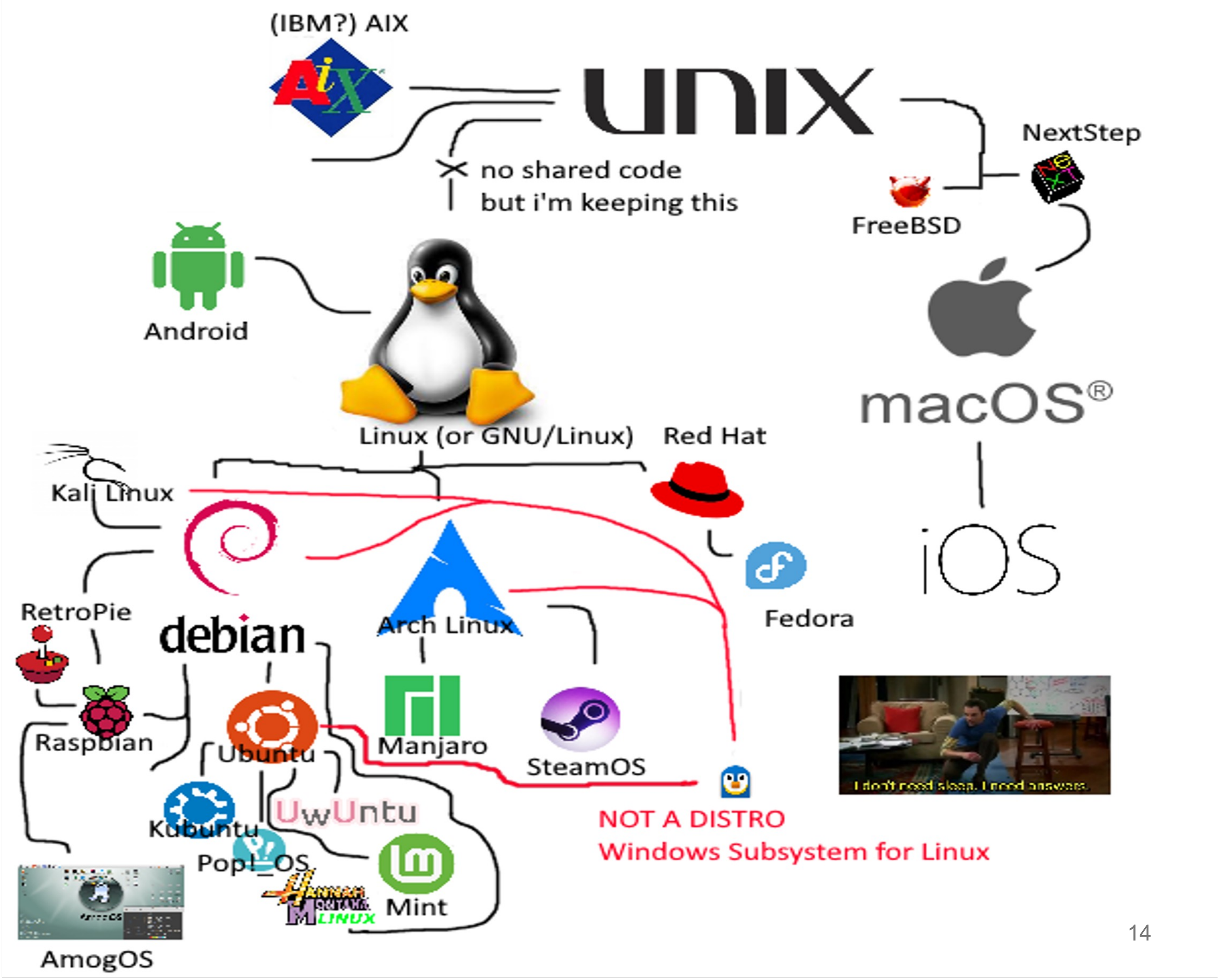
Linux Distributions

A Linux distribution is a compilation of Linux Kernel bundled with:

- System management tools (systemd, ufw)
- Server software (apache, ssh, mysql)
- Desktop applications (LibreOffice, VLC)
- Documentations (man/info pages)

Some popular Linux distributions are:

- Kali Linux (https://www.kali.org)
- Red Hat (https://www.redhat.com/en)
- Ubuntu (https://www.ubuntu.com)
- CentOS (https://www.centos.org)
- Debian (https://www.debian.org)
- Linux Mint (https://www.linuxmint.com)
- OpenSuSe (https://www.opensuse.org)



Interacting with Linux OS

LIVE FREE OR DIE
http://www.UNIX-systems.org
UNIX®
UNIX is a Registered Trademark of The Open Group
00B-0248

Interacting with Linux OS

For a user of an operating system there are two types of interfaces, using which a user can give commands to perform various operations:

- **Graphical User Interface:** GNOME, KDE, Unity, Xfce, Enlightenment, Sugar
- **Command Line Interface:** Also called a shell. A Linux shell is an interactive program that accepts commands from user via key board, parse them from left to right and execute them. Most of the shells available in todays Linux provides the features of executing user commands and programs, I/O handling, programming ability (scripts and binaries). Example shells are
  - **Bourne shell (`/bin/sh`):** First widely-used shell, scripting support, control structures, and pipelines.
  - **C Shell (`/bin/csh`):** C-like syntax, history mechanism, aliases, and job control.
  - **Korn Shell (`/bin/ksh`):** Combines features of Bourne and C shells.
  - **Bourne Again Shell (`/bin/bash`):** Introduced enhanced scripting, command completion, and job control.
  - **Z Shell (`/bin/zsh`):** Highly customizable, advanced tab completion, spelling correction, and powerful scripting.

Linux Shell Commands

- A shell command can be internal/built-in or External.
  - Internal commands (built-ins) are part of the shell itself, so they execute faster because they don't require spawning a new process. e.g., **`cd`, `dot`, `echo`, `pwd`**.
  - External commands are separate executable files located in the filesystem in the form of a binary executable or a shell script, e.g., **`cat`, `ls`, `mkdir`, `more`**.
- The general syntax of a shell command is

```
command    [option(s)]     [argument(s)]
```

- After reading the command the shell determines whether the command is internal or external.
- It processes all internal commands by using the corresponding code segments that are within its own code.
- To execute an external command, it searches the command in the **search path**. Directories names stored in the `PATH` variable. [`echo $PATH`]

Linux Basic Shell Commands

Basic Commands	Description
`bash, sh, csh, ksh, zsh`	Different types of UNIX/Linux shell
`who, whoami, finger, users`	User information lookup programs
`logout, exit, ^D`	Terminate the current shell session
`alias, unalias`	Used to create/remove pseudonyms for commands
`passwd, chfn`	Used to change user password, user info
`date`	Prints or sets the system date and time
`cal`	Displays calendar for specific month or year
`Clear <Ctrl + l>`	Clear the terminal screen
`hostname`	Display/set the system hostname
`uname -a`	Prints system information
`man [-k], apropos, whatis`	Displays manual and summary of commands
`whereis, which, info, type`	Locate binary (-b), source (-s), man pages (-m)
`shutdown, poweroff, reboot`	Power off / Restart system

Linux Basic Shell Commands (cont…)

Command for Directories only	Description
`cd`	Change directory
`mkdir -[p], rmdir -[p]`	Create and remove a directory
`pwd`	Display present working directory

Commands for Files only	Description
`cat, less, more, head, tail`	View contents of a file
`file`	Determines file type
`wc`	Displays line, word, character count of file(s)
`uniq`	Report or omit repeated lines
`sort`	Sort lines of files
`cut`	Remove column(s) from tabular files (tab, colon, space)
`paste`	Horizontally concatenate contents of two or more files
`grep`	Prints lines of files where a pattern is matched
`gzip, gunzip, bzip2, bunzip2`	Compression and un-compression software

Linux Basic Shell Commands (cont…)

Basic Commands	Description
`cp -[rpif]`	Copy files and directories
`mv`	Move/rename files/directories (No `–r` option required)
`rm -[rfi]`	Removes files/directories
`stat`	Displays file/directory statistics
`touch`	Update timestamp of a file/dir (access, modification, status change time)
`find / -name mv`	Search a file based on attribute in a dir hierarchy
`locate, updatedb`	Searches for the string in database(s)
`ls [-aldihFvStr]`	Displays listing of contents of a directory
`tar`	Archiving utility
`source`	Execute a script by current interpreter
`export`	Export a variable into the environment
`df, du, free`	Disk space and memory usage commands

Linux Manual Pages

Don’t expect to remember everything… I don’t!

Use **`man`** program to display help pages from `/usr/local/share/man/` dir having further sub-directories each for following:

- 1 – Shell commands; e.g., *`intro(1)`*, *`mv(1)`*, *`ls(1)`*, *`cat(1)`*, *`read(1)`*, *`write(1)`*
- 2 – System calls; e.g., *`intro(2)`*, *`read(2)`*, *`write(2)`*, *`open(2)`*
- 3 – Library calls; e.g., *`intro(3)`*, *`printf(3)`*, *`scanf(3)`*, *`read(3)`*
- 4 – Special files; e.g., *`intro(4)`*, *`tty(4)`*, *`null(4)`*, *`zero(4)`*, *`hd(4)`*
- 5 – File Formats; e.g., `intro(5)`, `passwd(5)`, `shadow(5)`, *`services(5)`*, *`interfaces(5`*
- 6 – Games & demos; (Manual pages for games available on system)
- 7 – Miscellaneous; e.g., `intro(7)`, `glibc(7)`, *`hostname(7)`*, `ip(7)`, `inode(7)`
- 8 – Admin functions; e.g., `intro(8)`, `shutdown(8)`, `fsck(8)`, *`mkfs(8)`*, `tune2fs(8)`
- 9 – Kernel routines; (Not available by default, mostly useful for kernel module developers)

Linux File Hierarchy Standard

Linux File Hierarchy Standard

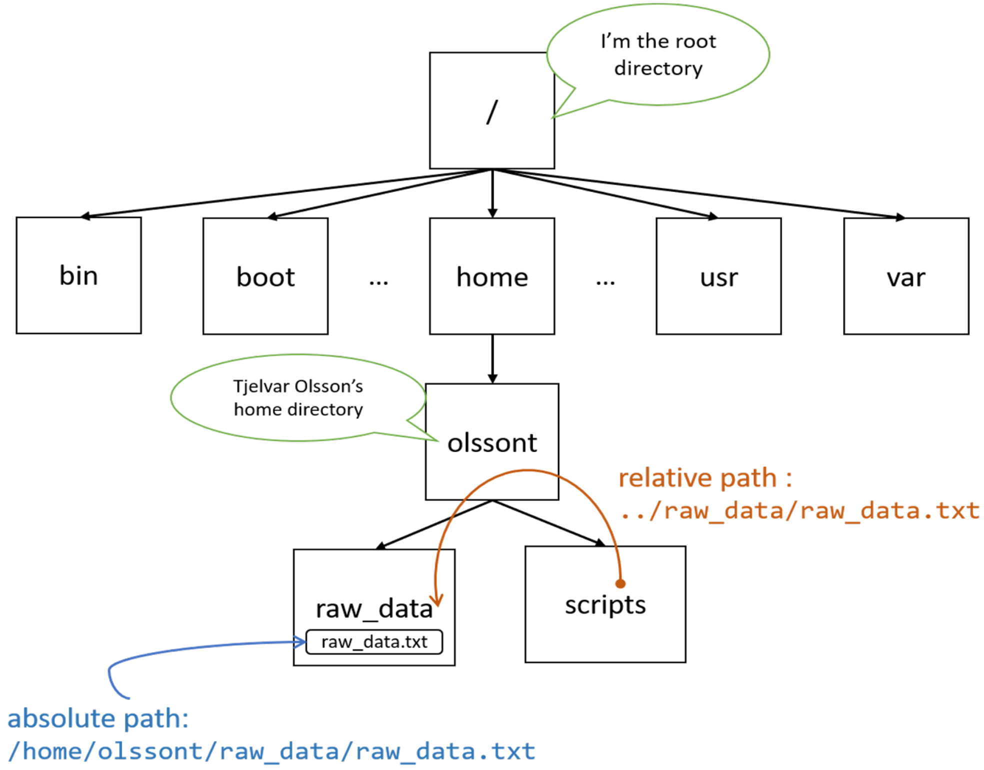
All UNIX based OSs normally follow the FHS. To get info of your file system hierarchy you can give the command **`$man 7 hier`** or can visit the following link

http://www.pathname.com/fhs/pub/fhs-2.3.pdf

Everything that exist on your Linux system can be found below the root (/) directory. Some important directories are:

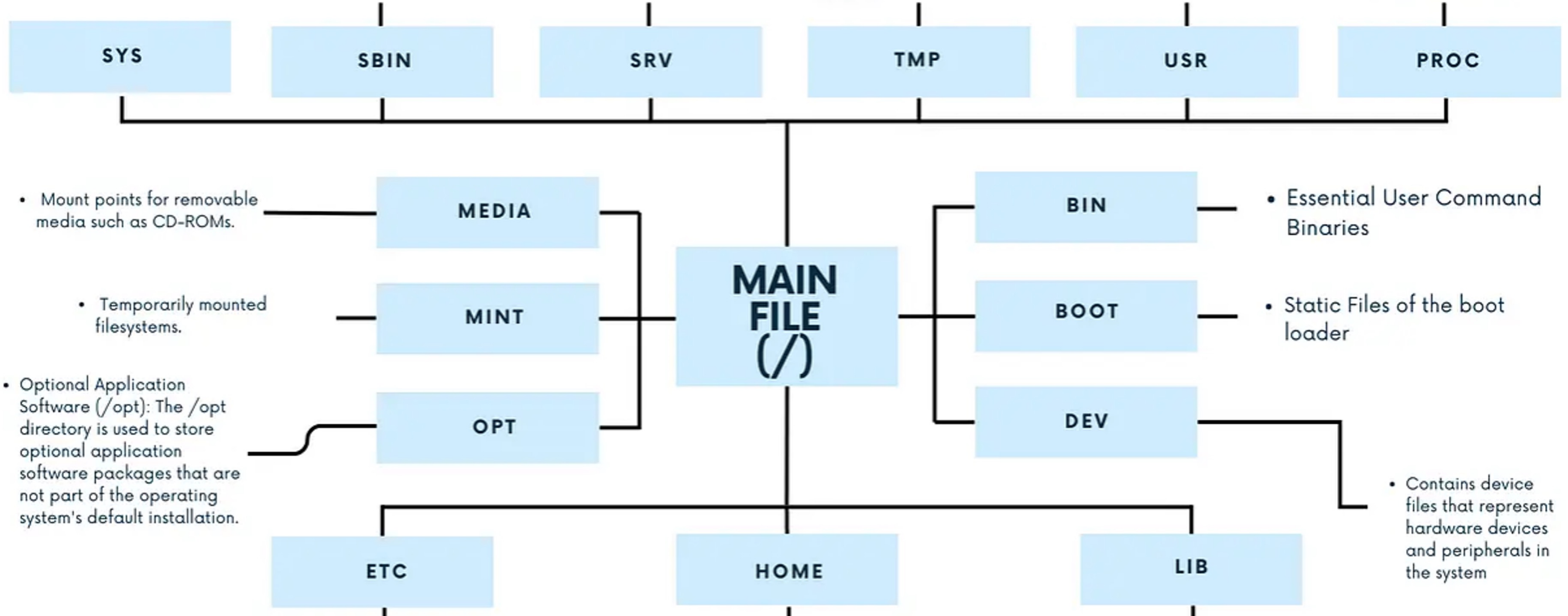
- Binary Directories: `bin, sbin, lib, opt`
- Configuration Directories: `boot, etc`
- Data Directories: `home, root, media, mnt, tmp`
- In-memory Directories: `dev, proc, sys`
- UNIX System Resources: `usr`
- Variable data: `var`

A filename in Linux is case sensitive, can be up to 255 characters long, and can contain special characters as well.



Absolute path always begins with a slash ("/")
Relative path starts from your present location

LINUX FILE STRUCTURE



MAIN FILE (/)

SYS

SBIN

- Essential system binaries, e.g., fsck, init, route.
- /sbin: Contains essential system administration binaries for system maintenance.

SRV

- Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.

TMP

- Temporary files. Often not preserved between system reboots and may be severely size-restricted.
- Files under this directory are deleted when the system is rebooted.

USR

- Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.
- Contains binaries, libraries, documentation, and source code for second-level programs.

PROC

- Virtual filesystem providing process and kernel information as files. In Linux, it corresponds to a procs mount. Generally, automatically generated and populated by the system, on the fly.

MEDIA

- Mount points for removable media such as CD-ROMs.

MINT

- Temporarily mounted filesystems.

OPT

- Optional Application Software (/opt): The /opt directory is used to store optional application software packages that are not part of the operating system's default installation.

BIN

- Essential User Command Binaries

BOOT

- Static Files of the boot loader

DEV

- Contains device files that represent hardware devices and peripherals in the system

ETC

- Host-specific system-wide configuration files.
- Contains configuration files required by all programs.

HOME

- Users' home directories contain saved files, personal settings, etc.
- Home directories for all users to store their personal files.
- example: /home/kali, /home/kv

LIB

- Libraries essential for the binaries in /bin/ and /sbin/.

To Do

- Install the appropriate hypervisor (VirtualBox/VMWare) and install Kali Linux on it. Go through the provided Linux Commands Cheat Sheet and practice shell commands.
- Watch OS video on Setting-up Linux Environment:
  https://www.youtube.com/watch?v=wO0Y1IJIajM&list=PL7B2bn3G_wfBuJ_WtHADcXC44piWLRzr8&index=2
- Watch OS video on Basic Shell Commands (Part-I):
  https://www.youtube.com/watch?v=_YPvqThyM4c&list=PL7B2bn3G_wfBuJ_WtHADcXC44piWLRzr8&index=4
- Watch OS video on Basic Shell Commands (Part-II) :
  https://www.youtube.com/watch?v=dUkskLi70nI&list=PL7B2bn3G_wfBuJ_WtHADcXC44piWLRzr8&index=5
- Watch OS video on Linux **`vim`** Editor (If you do not know vim you do not know Linux):
  https://www.youtube.com/watch?v=7tFniseSLzM&list=PL7B2bn3G_wfBuJ_WtHADcXC44piWLRzr8&index=6

Coming to office hours does NOT mean that you are academically weak!