Introduction to UNIX/LINUX:

Welcome to the world of Freedom

CSE Dept.

AIET

Contents

Basic stuff on computers and OS

Introduction to Unix/Linux, why use it !

Hands on session

What is a Computer ?

- Computer is a monster device that basically deals with huge bunch of numbers

- Primarily it has
  Processor (Intel/AMD/ARM)(x86/x64)
  RAM, Hard drive, graphic card, mother board, optical drive, display, CD-rom and etc.

What manages a Computer ?

- In a nutshell, The motherboard has a chip containing a program called BIOS which has a subprogram called boot loader, which loads up a complex Software called as Operating system to RAM, and

- This is called as bootstrapping or booting and the programs which resides in chips are called as firmwares.

What is an Operating System ?

- The operating system is a software that manages the computer’s resources such as the RAM, the hard disks, the duration of processing on CPU for each running programs for multi processing etc.

What is an Operating System ?

- Basically it takes care of Networking
  (It knows how to talk to other computers),
- It knows File systems and file management
  (It knows how to store and retrieve files and docs Efficiently)
- It provides Access control and User management
  (It knows who uses the computer and who has what type of authority over the computer)

One last point on Operating System ?

- Computers are all about signal processing,they are being operated by high and low voltages, for our understanding we represents them as 1 and 0.
- Computer only understand a very low level language but highly structured, Operating system acts as a manager to this powerful monster.
- Operating system has mediators(compilers/Interpreters), these programs acts as language translators and they translate language written by humans to the language that the OS understands.

Types of OS in the beginning of time.

- FreeDOS (Open Source project)
  - Windows was developed from this.
    (Evil Bill gates made ton of money)

- UNIX and its derivatives
  - Linux (Linus Torvald's baby)
  - MacOS (Apple's)

A Little story time!

- Decades ago Unix was only available to researchers and only top few universities had the hold of it.
- A professor started a his own project called Minix, which mimics Unix and was too basic.
- A undergraduate student from Finland named Linus Torvalds studied Minix, added a little more to it, And made it public, which eventually became Linux, It was 80 KB of codes back then, currently it has more than 3 Million lines of code.

A Little story time!

- Another instance was a Professor at MIT wanted
  To modify his lab's printer to handle printing
  requests, he needed source code in order to do that.

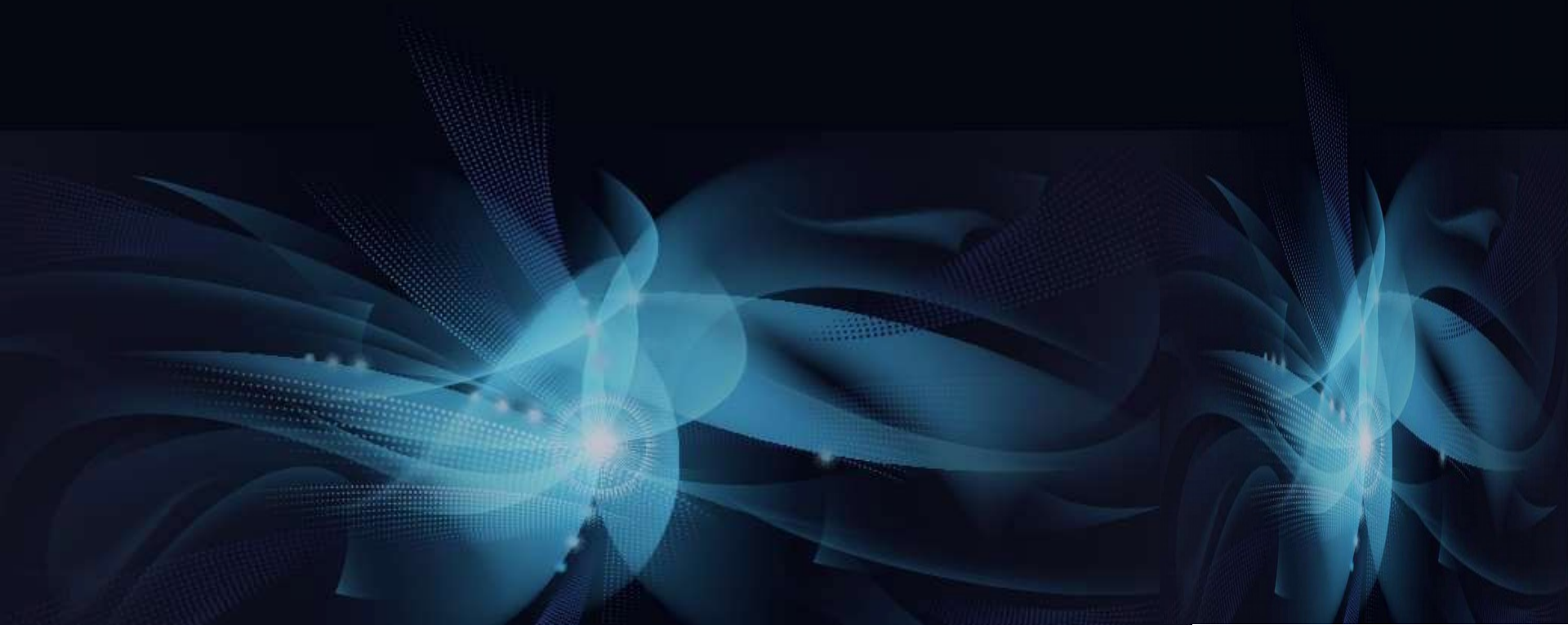
- He borrowed the code from a professor who was from
  a different university, but he insisted not to share it with
  anyone else.

- This made him to start Free software movement,
  His name is Richard Stallman, the father of GNU.
  This is what boosted Unix and Linux.

End of story!

- Linux borrowed most of it code from UNIX.
- When Linux was getting better and better, Unix in turned borrowed code from Linux.
- And this went on and on for decades, and which led to evolve more incredible technologies as well as more developments and free stuff.

Introduction to UNIX/LINUX



General Note

- Unix and Linux are both separate entities.
- But the core ideas, principles and philosophy are almost similar.
- Majority of the public uses Linux so does our labs.

Gods and the Bell Lab (Unix and C).

How was it like to See “C”.

INLINE ASSEMBLER VS. C

```
_asm
{
        mov eax, a
        add eax, b
        mov c, eax
};
```

```
c = a + b;
```

Unix/Linux History

1969
Unix was created at bell labs

1970+
Bell labs makes unix, a freeware

1980+

Solaris OS,
TCP/IP (Internet explosion)
MIT X-Windows (GUI for unix)

1990+

GNU and Richard Stallman
Linux and Linux Torvalds
Open Source Explosion

C (machine independent syntax) V/S Assembly (Horrible and wont work most of the time)

```
1  int arith(int x,
2            int y,
3            int z)
4  {
5      int t1 = x+y;
6      int t2 = z*48;
7      int t3 = t1 & 0xFFFF;
8      int t4 = t2 * t3;
9
10     return t4;
11 }
```

```
1    movl 12(%ebp),%eax          Get y
2    movl 16(%ebp),%edx          Get z
3    addl 8(%ebp),%eax           Compute t1 = x+y
4    leal (%edx,%edx,2),%edx     Compute z*3
5    sall $4,%edx                Compute t2 = z*48
6    andl $65535,%eax            Compute t3 = t1&0xFFFF
7    imull %eax,%edx             Compute t4 = t2*t3
8    movl %edx,%eax              Set t4 as return val
```

Why use UNIX/LINUX

- Windows is not for free, when the trial is over, some of the features will be unavailable, some software fails to install, and the OS keeps bargaining for activating windows.

- Linux is more secure than windows, No third party programs can run automatically in Linux, with out the permission of user, In windows this is how virus operates because they run automatically. (It is the matter of privacy)

Why use UNIX/LINUX

- Android in smart phones is basically Linux, while iOS is UNIX based.
- Linux can run in any type of hardware, such as smart-watches, modern TVs, smart-fridges almost anything even old computer hardwares.
- Microsoft runs a cloud business called as “Azure”, the whole operating infrastructure is built on Linux. Microsoft is going to release Windows which as its own Linux Kernel, that too very soon.

Why use UNIX/LINUX

- Every clouds are running on Linux,
- To use Windows, it must be fully installed at first into the system. This is not the case in terms of Linux.
- Linux can run in RAM (Elite operating systems), they can also be used straight away without even installing them, by booting from pendrives, CDs, floppies as well.

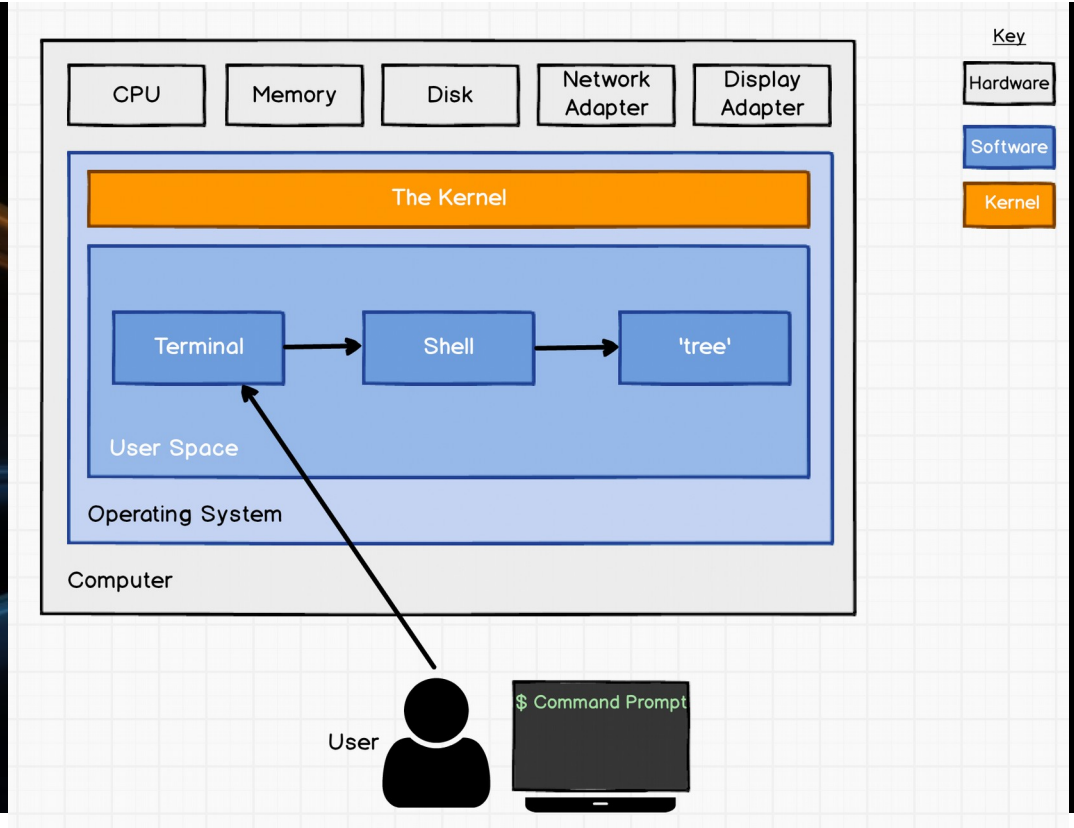
Why use UNIX/LINUX

- To be more general, Linux is truly transparent
  It is almost everywhere.

- It is full of free software, for example, one can make use of freely available software such as Open-Office or LibreOffice rather than Paid Microsoft Office.

- Unix/Linux is highly customizable, it size typically varies from Kilobytes to Gigabytes, for instance DSM (damn-small-linux) is Linux which is just 50 MB in size, with a GUI

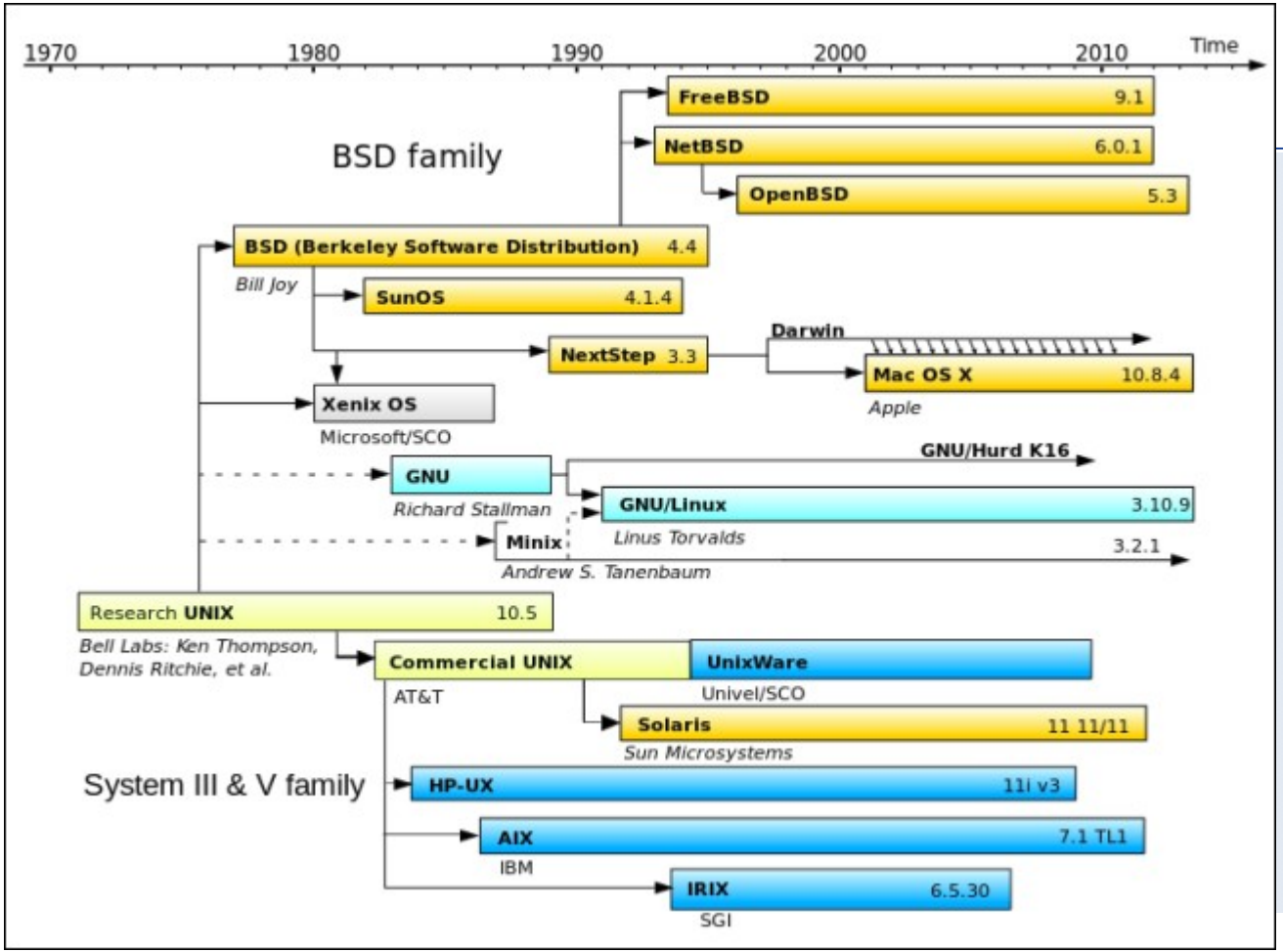
Unix/Linux Architecture



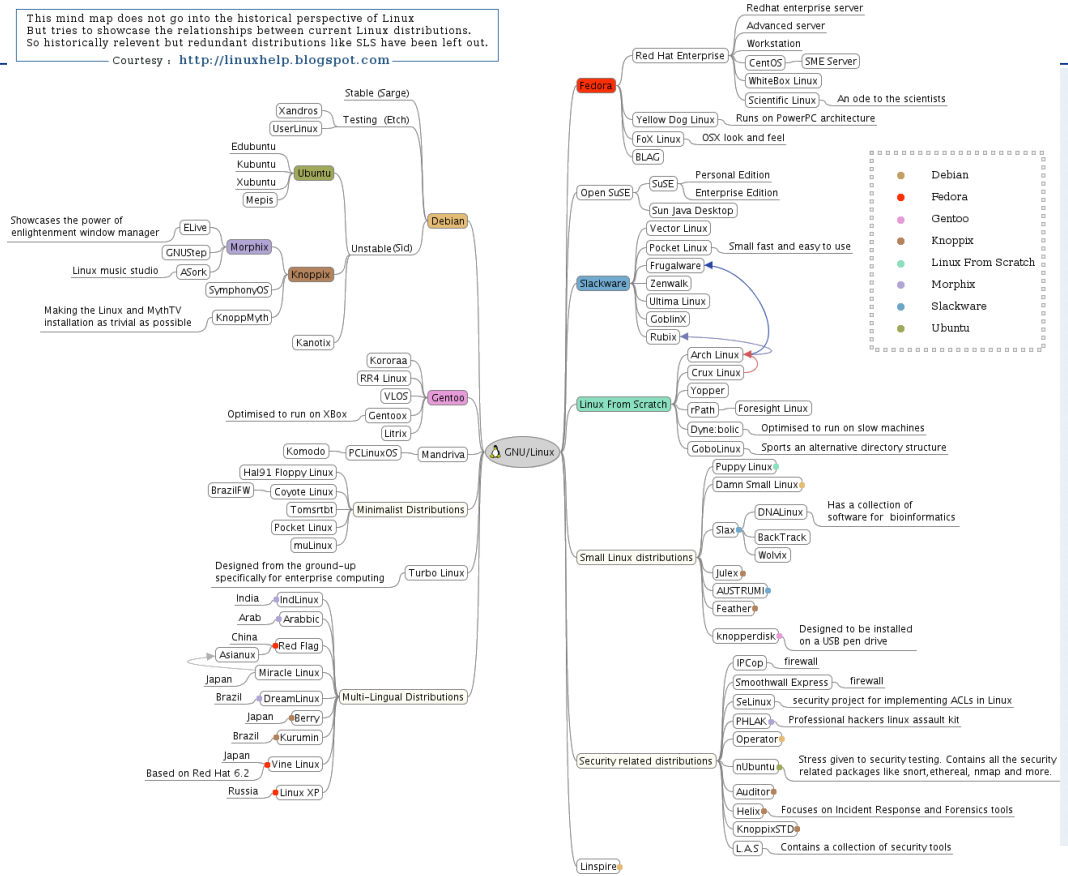
UNIX/LINUX is actually the Kernel

- The **Shell** is a program that takes keyboard commands and passes them to the operating system to carryout operations.
- When using a graphical user interface (GUI), we need another program called a **Terminal** emulator to interact with the shell.
- A **Kernel** is the central part of an operating system. It manages the operations of the computer and the hardware. Shell tells The kernel what to do, users interacts with Kernel via shell.

Flavors of Unix



Flavors of Linux



Key concepts of Unix/Linux

- In Unix/Linux every thing is referred as files,
  It means Hard-drives, Network devices, Documents and the all hardware are all files.

  (ordinary files, device files, directory files)

- In Unix/Linux the user “root” is equivalent to “Administrator” in Windows. If a user in Linux needs to do anything beyond his user space, he needs root privileges.

Key concepts of Unix/Linux

Ordinary Files

- Source code of program
- Files that are generated by compiling source code

Directory Files

- Files that keep track of Files that are present in them

Device Files

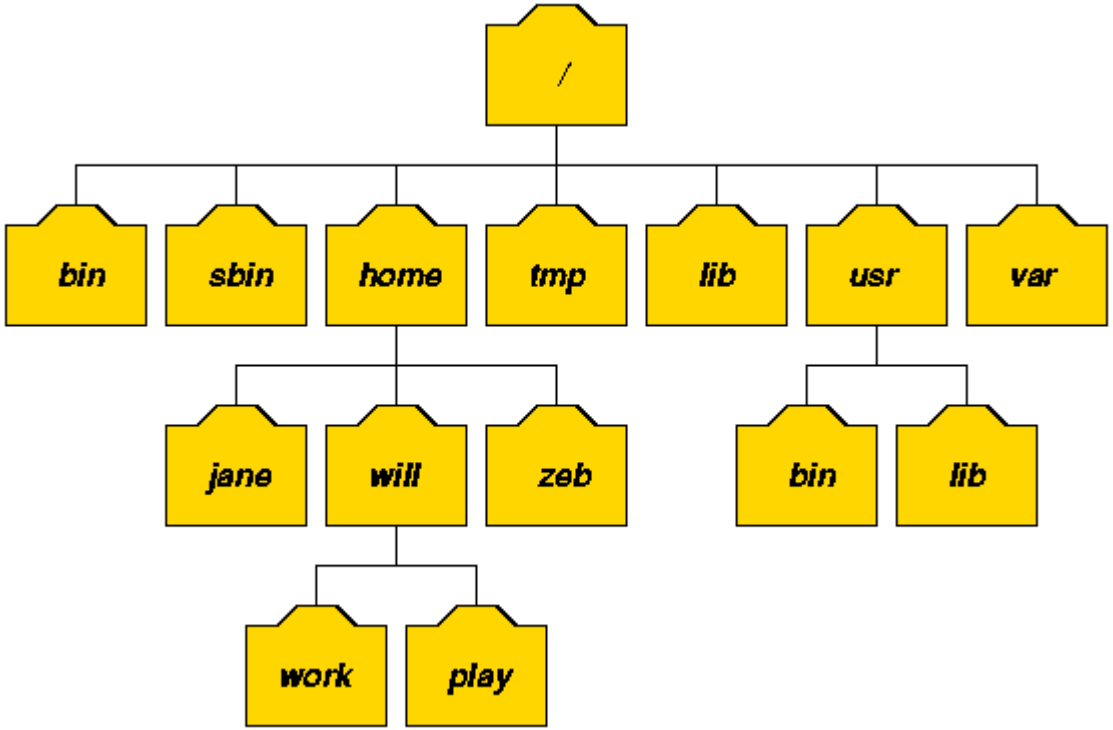
- Files that refers to Hard drive or pendrive or Cd-rom

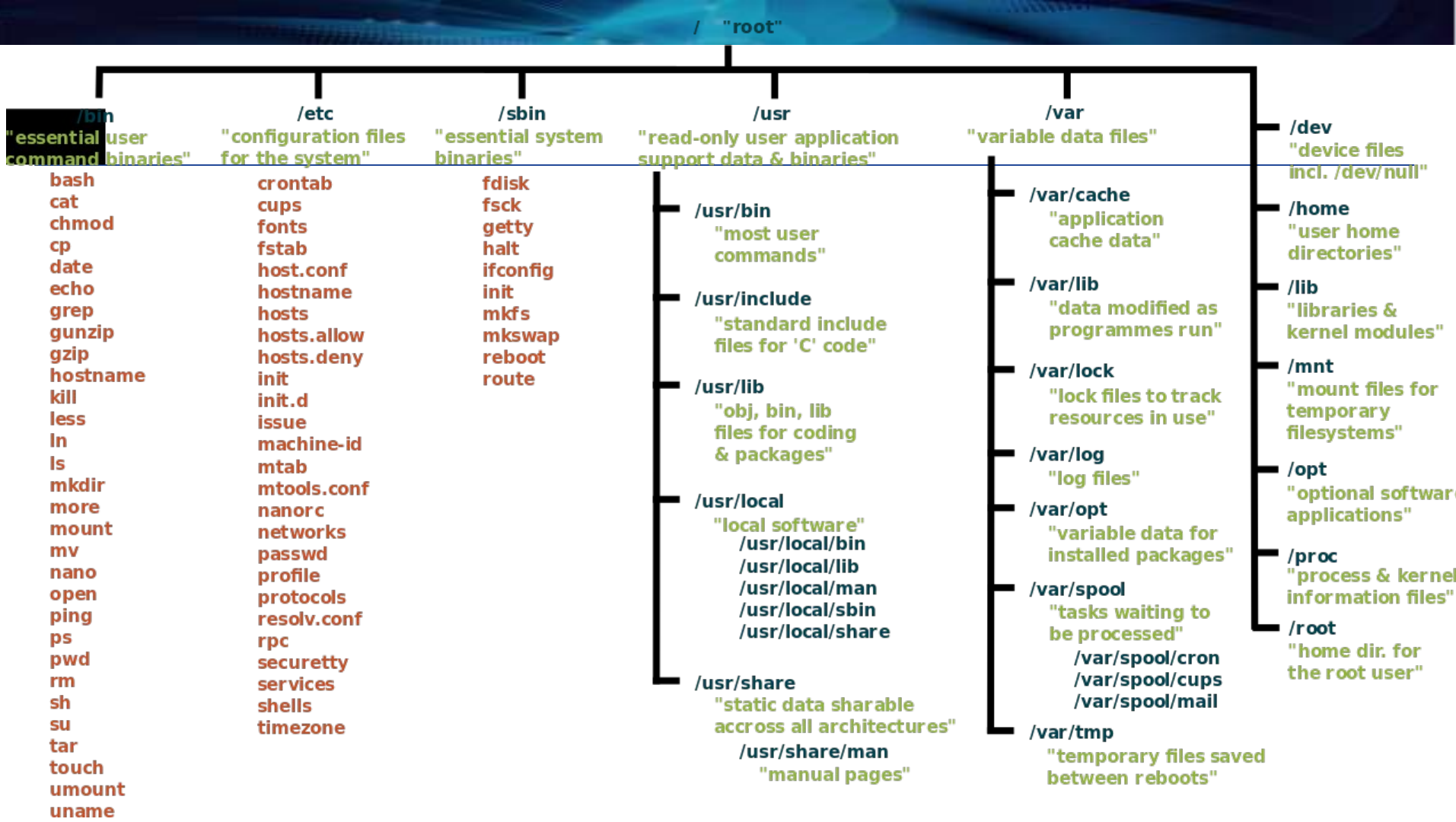
The File system Hierarchy in Unix/Linux

- As Unix/Linux relates everything as files, the files which are common in nature are grouped together and are put in folders (directories).

- A normal user can access only a set of files and directories, but root can access everything.

The File system Hierarchy in Unix/Linux




/ "root"
/bin
"essential user command binaries"
bash
cat
chmod
cp
date
echo
grep
gunzip
gzip
hostname
kill
less
ln
ls
mkdir
more
mount
mv
nano
open
ping
ps
pwd
rm
sh
su
tar
touch
umount
uname
/etc
"configuration files for the system"
crontab
cups
fonts
fstab
host.conf
hostname
hosts
hosts.allow
hosts.deny
init
init.d
issue
machine-id
mtab
mtools.conf
nanorc
networks
passwd
profile
protocols
resolv.conf
rpc
securetty
services
shells
timezone
/sbin
"essential system binaries"
fdisk
fsck
getty
halt
ifconfig
init
mkfs
mkswap
reboot
route
/usr
"read-only user application support data & binaries"
/usr/bin
"most user commands"
/usr/include
"standard include files for 'C' code"
/usr/lib
"obj, bin, lib files for coding & packages"
/usr/local
"local software"
/usr/local/bin
/usr/local/lib
/usr/local/man
/usr/local/sbin
/usr/local/share
/usr/share
"static data sharable accross all architectures"
/usr/share/man
"manual pages"
/var
"variable data files"
/var/cache
"application cache data"
/var/lib
"data modified as programmes run"
/var/lock
"lock files to track resources in use"
/var/log
"log files"
/var/opt
"variable data for installed packages"
/var/spool
"tasks waiting to be processed"
/var/spool/cron
/var/spool/cups
/var/spool/mail
/var/tmp
"temporary files saved between reboots"
/dev
"device files incl. /dev/null"
/home
"user home directories"
/lib
"libraries & kernel modules"
/mnt
"mount files for temporary filesystems"
/opt
"optional softwar applications"
/proc
"process & kernel information files"
/root
"home dir. for the root user"

Accessing files

- Lets take an example that a file called “program_1” resides in “bin” directory, it is can be accessed as “/bin/program_1”, This is called as **Absolute path**, as it starts from root (top-level) which is “/”

- Assume that you have two directories “photos” and “videos”, you can access “photo_1.jpg” as “./photos/photo_1.jpg”

- Whenever we compile a C program, It creates a file called as “a.out” in the present directory, generally we access it by “./a.out”, This is **Relative Path.**

How Unix relates users

- In UNIX, every file has a user tag and the group tag. User tag to a file defines that he is the owner of that file and only he has permission to read or modify etc.

- Set of users form a group in Unix, each group has group tag (a name), If a file has group tag → (Group_A). that implies all the users of Group_A has common privileges (permission) on that file. It could be only to read the file, read and write, or only to execute the file.

The Shell and the shell prompt

- Shell is a program that tells the kernel what to do, receives the output from kernel and gives back to the user.
- Terminal is a program used to communicate with the shell via commands, shell prompts to enter commands, when commands are entered it becomes idle until the kernel process it.
- It gets back with the output, or just prompts again indicating the completion of previous command.

The Shell

- If you are a root user, the shell would look like with ‘#’,

  root@My_computers_fancy_name: # ./bin/run_web_server

- For a normal user, with ‘$’.

  username@his_computer_name: $ ./my_C_program

The command and its general structure

- Unix has Internal and External commands, Internal commands are primitive, they are used commonly than external commands.

- A typical command would look like. (Ignore till ‘$’)

  user@host :$ **command_name [arguments]**

The command and its general structure

user@host :$ **command_name [arguments]**

- In some cases arguments are must, generally

  [arguments] – > optional arguments

  arguments can be short/longvariants with values.

- Short → -s [value] // [value] depends on command
- Long → --long-arg [value]

A command example

The command “ls” is used for listing directory contents, an can be used as follows. (ignore ‘$’)

- $ ls
- $ ls -l #single short arguments
- $ ls -l -a # multiple short arguments
- $ ls -la # short args can be merged
- $ ls --long-listing --all # long args, cant be #merged, same as ‘ls -la’

Getting help for commands

- If you are using a unknown command, read a little description about it on the internet, make sure it is not powerful to harm the computer, when it is mistakenly used.

  You can try the following as commands
- help **command_name**
- **command_name** -h
- **command_name** --help
- man **command_name**

man (Manual reference pages for commands)

- When one needs more detail about a command, man provides comprehensive information about it, applies only if a manual page exists for that command

- It consists
  a) Name, name of the command
  b) Synopsis,   syntax of the command, optional args if ‘[]’ exists
  c) Description,   little detail on what command does
  e) Options,   all arguments that command has and little detail
  d) Example and Author references

Multiple command, Pipes and Re-directions

- Multiple commands can be executed by adding ‘;’ between commands. Example **‘$ command1 ; command2’**

- Pipes in unix is all about, output of one command acts as input to the other command (we’ll explore later).

- Unix has (STDIN, STDOUT, STDERR) (standard input, output and error), we can handle each of them be redirecting them. Example (we can give input to a program from file rather than typing from keyboard) (we can redirect output if we just want the command to run), (Demo will be in hands on session)

Hands-On Session


Thank You!!