Linux Fundamentals Agenda

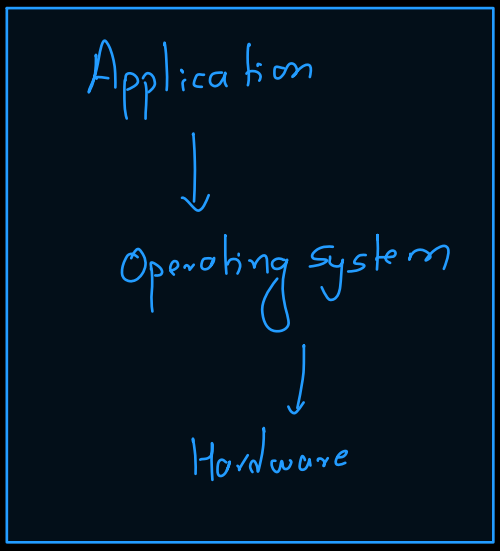
1. Fundamentals of Linux
2. Different shells in Linux
3. Linux commands on ubuntu
4. Linux commands for Devops
5. Linux Filesystem
6. Package management in Linux
7. Linux Admin - Configuring a DNS server
8. Shell scripting
9. Linux vs Windows
10. Linux vs Unix
11. Linux Interview Q A

Operating Systems - Linux

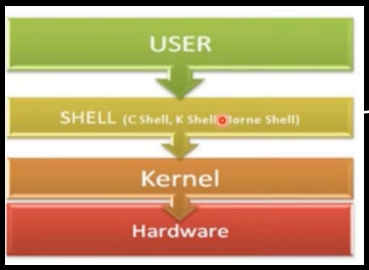
Operating System

- An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for hardware programs. The OS is a vital component of the system software in any computer system. Applications usually require an OS to function.

Ex: Windows
Linux
IBM AIX
Soloris
HP-US



Linux is Organized



→ most OS have their own shell
(Windows - Powershell ... and so on)

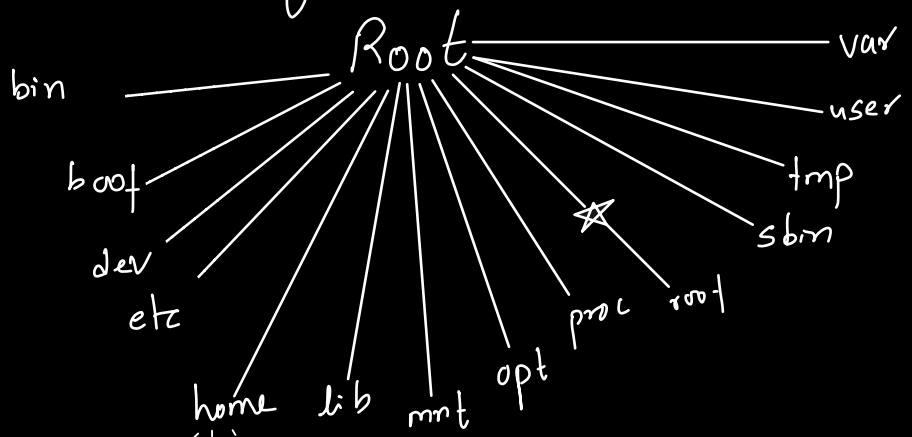
→ Linux Run-Level

The init process, by default, runs the system in one eighth of the run levels

	Run Level
Runlevel 0	Halt System - to shutdown
Runlevel 1	Single user mode
Runlevel 2	Basic multiuser mode (No NFS) → Network file system
Runlevel 3	Full multiusermode (CLI) → AWS works here
Runlevel 4	Unused
Runlevel 5	Multiusermode with (GUI)
Runlevel 6	Reboot system

In Linux OS

Linux Directory



— these are common linux directories

User Mangement

useradd <username>
passwd <userpswd>

Linux Fundamentals

Linux is a Kernel not an operating system. Linux when combined with GNU project in 1990s became an OS. A Kernel is a central component of an operating system that manages operations of a computer & hardware. It basically manages memory & CPU time. Kernel is the core component of an OS.

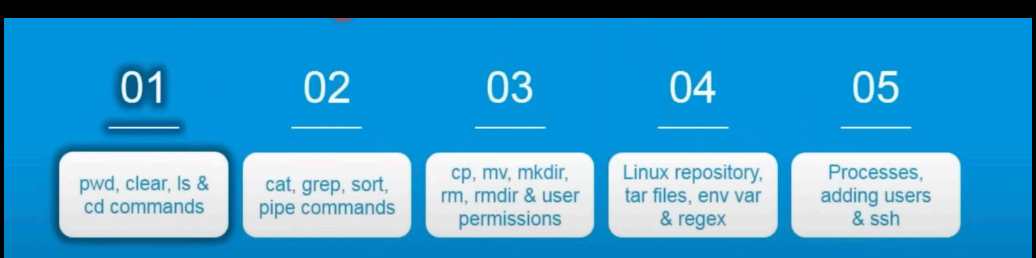
Objectives of Kernel:

- To establish communication between user level application & software
- To decide incoming processes
- To control disk management
- To control memory management
- To control task management

Features of Linux

- Simple updates
- Free software licensing
- Access to source code
- Multiple distributions (red hat, fedora etc)
- Better malware

Hands-on Linux Commands



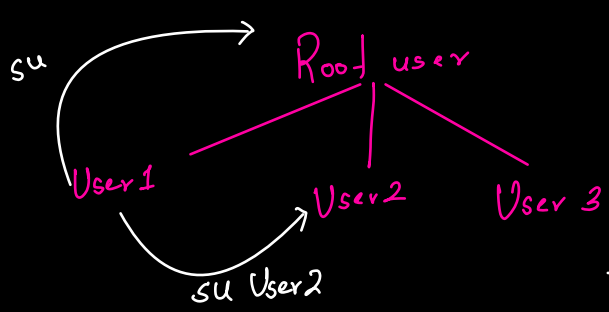
Working with Linux Commands

1. pwd → print the current directory
2. clear
3. echo → command that writes its argument to standard output

```
ayushpoudel@eduroam-226-92-153-checkoutpage % echo hi, my name is Ayush Poudel. I am learning Linux Fundamentals currently.
hi, my name is Ayush Poudel. I am learning Linux Fundamentals currently.
```

printed as output

4. su → switch user



Working in cloud CLI like AWS CLI and so on

Now, we know root has permission on anything, so whenever we want to do something that the user is not allowed: we use `sudo` command.

5. sudo → super user do

⑥ ls ⟶ list

- → ls path : write ls followed by path to see files in that folder
- → ls -l : list all contents
- → ls -a : list all hidden contents
- → ls -S : sorts and lists all the dirs
- → ls *.html : only lists .html files

⑦ cd ⟶ change directory

⑧ cat → display content of text files or concatenate two files

- cat -b : to display content of text files & concatenate several files
- cat -s : squeeze all blank spaces
- cat -E : show EOF

Using >, >> in cat

cat _____ >> a.txt : will concatenate a & b
(input in terminal / add b.txt)

cat _____ > b.txt : will touch b.txt & concatenate anything key or > to b.txt

Why to use cat over vim or nano?

→ cat command can display multiple files at once

→ cat command can be used to append as well as create

- >> for append
- > for create & append

} directly in the terminal

⑨ Grep → to search for a particular string / word
↳ similar to cmd+f in GUI, we use grep in CLI

- grep "options" file1.txt
  - options → string
  - file1.txt → path
- grep -i : returns results for case insensitive
- grep -n : matching strings along with line no.
- grep -v : returns lines not matching the string
- grep -c : returns no. of lines of matched strings

Very useful to edit codes

⑩ sort → used to sort contents of file

flags:
- -r : in reverse
- -f : case insensitive
- -n : returns results per num order

(11) cp ⟶ copy command in CLI

$ cp [options] source destination

Flags:

-i	enters interactive mode, CLI asks before overwriting
-n	Does not overwrite the files
-u	Updates the destination file only when source is different
-R	Recursive copy, even copies hidden files
-v	Verbose: Prints informative message

(12) mv ⟶ move command in CLI

mv [options] source destination

i	Interactive mode
u	Updates if only source file is different
v	Verbose: print source and destination file

(13) mkdir ⟶ make directory

mkdir -p file1/{file01, file02, file03}

used to create multiple directories in parent all at same time

User Permissions in Linux

Any user permission if given the command

ls -l

↳ which just means long is given as

```
[ayushpoudel@eduroam-226-92-153 LinuxFirstInstance % ls -la
total 0
drwxr-xr-x   5 ayushpoudel  staff  160 Jun 28 15:00 .
drwx------@ 15 ayushpoudel  staff  480 Jun 28 15:01 ..
drwxr-xr-x   2 ayushpoudel  staff   64 Jun 28 15:00 bin
drwxr-xr-x   4 ayushpoudel  staff  128 Jun 28 15:00 home
drwxr-xr-x   2 ayushpoudel  staff   64 Jun 28 15:00 root
```

↳ these are the user permissions

How to interpret it?

→ It is an array

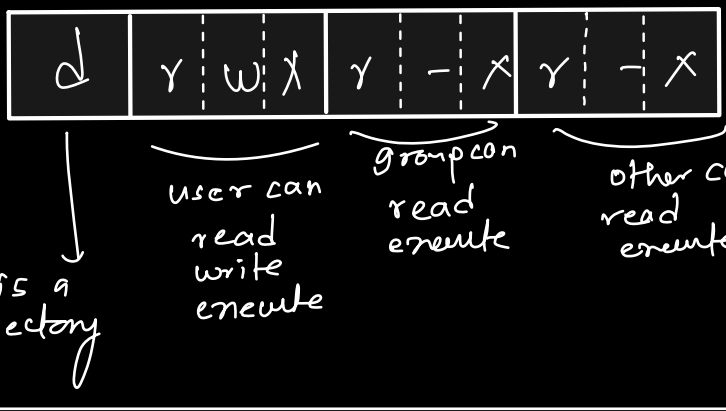
This part has
- : normal file
d : directory
c : char special file
b: binary file

This part has r, w, x and -

r = read
w = write
x = execute
- = nothing

Now let's interpret the above user permissions in pmk

`drwxr-xr-x` if is to be placed in our array above:



Now, how do we modify such user permissions?

→ chmod : to change access permissions of files & directories

→ chown : to change owner of files and directories

→ chgrp : to change group ownership of files & dirs.

To change the rwx permissions is very easy:
you need to know that we have

① user → u
② group → g
③ others → o

so, if you want to add r, w to g then just do

chmod g+rw file1.txt

if you want to remove r, x from o then

chmod o-rx file1.txt

then file 1 perm is changed from

-rwxr--x r-x ⟶ -rwxrwx---

Environment Variables

printenv → print the list of all environment variables

echo $HOME → print the path of home directory

echo $path → a colon seperated list of directories in which shell looks for commands

echo $ hostname

echo $ username

echo $ lang

echo $ BASH_VERSION

Regular Expressions

RegEx are use to search through data. It can be piped along with 'grep' command to find patterns of text in a file.

Symbol	Explanation
.	Replaces any character
^	Matches the start of string
$	Matches the end of string
*	Matches the preceding character zero or more times
?	Matches the preceding character one or more times
()	Groups regex
\	Represents special characters

Processess

Process is an instance of running program.

1. Definition

A process is program in execution. It consists of the program code, data and resources (like open files and network connections) allocated by OS.

2. Purpose

Processes allow the OS to manage multiple tasks concurrently, providing the illusion that multiple programs are running simultaneously even on a single-core processor

3. Attributes

Each process has its own unique ID, its own memory space (though some parts may be shared via memory mapping), and its own set of resources allocated by OS.

4. Lifecycle:

They typically go through several states, → creation - running - waiting - terminated

The OS schedule manages the execution of processes, determining which process gets CPU time in what order.

5. IPC (Inter Process Communication)

They can communicate with each other using various IPC mechanisms provided by the OS, such as pipes, shared memory, signals & sockets

6. Control

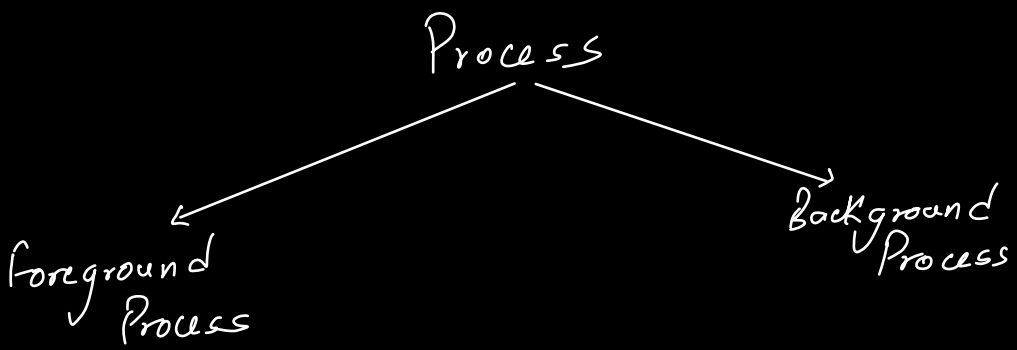
OS provides utilities and commands (like ps, top, kill etc) to monitor processes, manage their execution, and terminate them if necessary.

In sum,

→ An instance of a program is called process

→ Any command given to linux kernel starts a new process

→ There can be multiple process/instances of the same program



A process when viewed using `top` command looks like :

PID	USER	PR	NI	VRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND

- PID → Process ID
- PR → Priority
- NI → Niceness value
- VRT → Virtual memory
- RES → Physical memory
- SHR → Shared memory
- S → status
- %CPU → CPU time
- %MEM → Physical memory used
- TIME+ → Total CPU time
- COMMAND → command