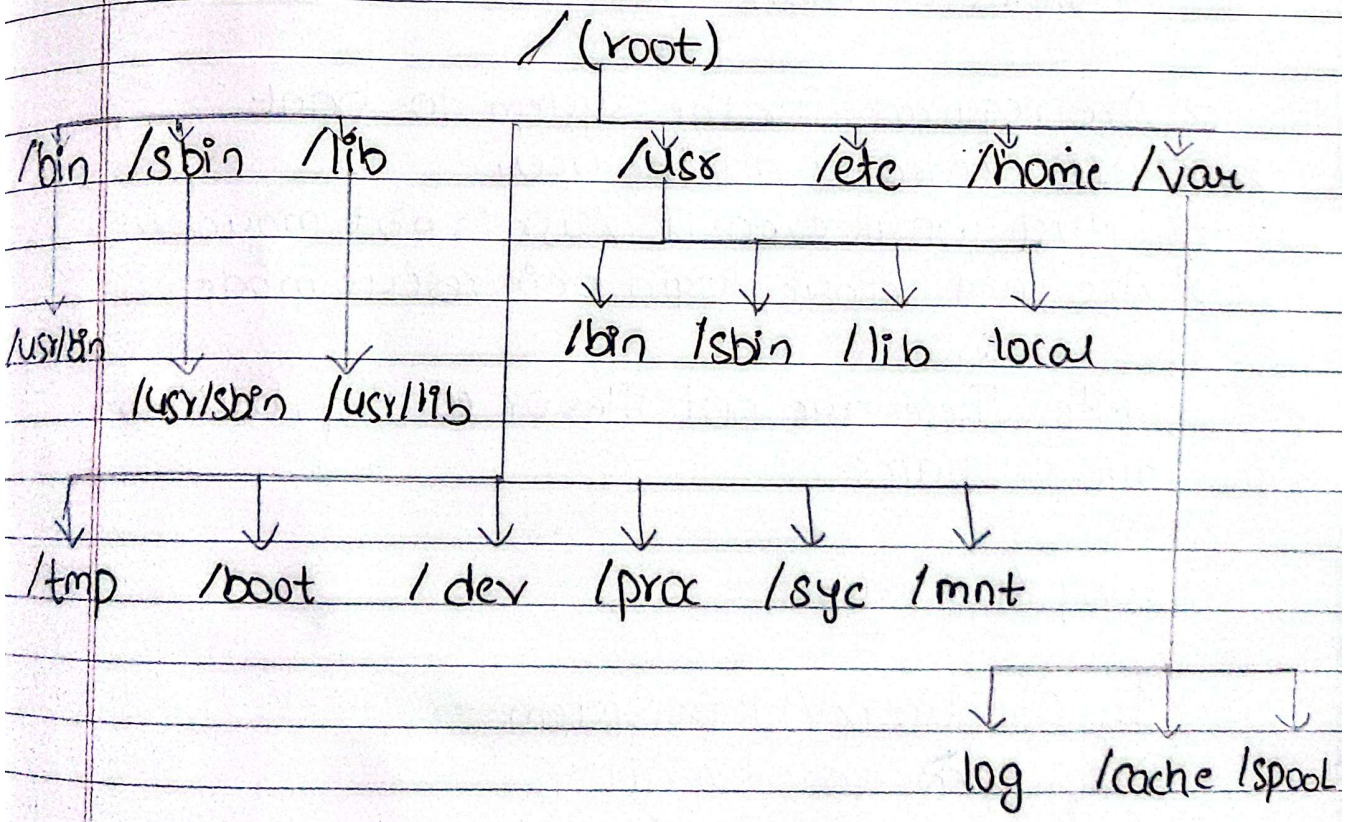
/alish 080/

Phase 1: Linux Foundation

1) Linux File System Structure

- / (root)
- /home
- /etc
- /var
- /bin, /sbin
- /usr
- /tmp
- /boot
- /dev, /proc (basic idea)



① / - Root Directory

⇒ The top-level directory.
⇒ Every file and folder in Linux exist under /

Note: Not same as /root

You can see directories like /etc, /home, /user, etc

② /bin (Essential User Command)

/bin stands for binary.

- It stores executable program (commands) that

→ Are required for the system to boot
→ Are needed by all the user
→ Must work even if /usr is not mounded
→ Are used before login or in rescue mode

Note: These are not files or config, but they are programs.

Why /bin exist

In old Linux / Unix System
- Disk space was small
- System booted in stages
- /usr was in seperate disk

So Linux needed a place to store a critical command

that must work immediately.

That place was /bin.

- What is stored inside /bin.

Commands needed to:
1) View files
2) Move / copy / delete file
3) Read text
4) Start a shell
5) Repair or fix broken system

Example:
ls, cp, mv, rm, cat, sh, bash, chmod, & chown
chmod → Fix permission
chown → Fix Ownership

- Why /bin is critical

→ Imagine /usr is broken or corrupt
→ GUI is broken

You boot into recovery shell, you would need ls, cat, cp
ls
cat /etc/fstab
cp backup.conf orginal.conf

→ These must work without /usr, so thats why /bin is used

- Modern Linux system (change)

→ Today, Disk are large,
/usr is found easily ~~early~~ at early state

So, Linux developer merged directories

Now,

/bin → /usr/bin
/sbin → /usr/sbin
/lib → /usr/lib

This is called usr-merge

what happens is when you ls -l /bin
you will see

⇒ /bin → usr/bin

So,
- /bin is not a real directory
- It is a symbolic link
- Actual files lives in /usr/bin

Why do we need /bin today then?
→ Thousands of script runs /expect /bin specially olds one
→ System Compatibility

Removing would break software
so today linux system keep them as link not storage

So, what is /usr/bin

→ This is the real folder: In Modern Linux system, almost every executable program are physically stored here. Whether it's python, git or ls, the actual bits and byte I've inside /usr/bin.

When you do ls / you might also see bin.usr-is-merged.

↳ This is a marker file. It acts as a safe to proceed sign for the system. It tells the Operating system "Hey I have already move everything to /usr/bin and set up the short cut. Don't try to move them again."

Name	Type	Purpose
/usr/bin	Directory	The actual physical home for your command
/bin	Symbolic Link	A redirect so "ls" works whether you look in /bin or /usr/bin
/bin.usr-is-merged	Marker file	A note saying Merge to /usr is complete

To see both /usr/bin and /bin are same (leads to same)

/usr/bin/date and /bin/date } both same

③ /sbin

sbin = system binary

It stores administrative commands used to.

- Boot the system
- Configure hardware
- Manages, disk, networking and Kernel
- Repair a Broken system (fsck)

These are mainly for the administator (root)

Who uses /sbin?

- root user.
- system startup scripts
- recovery mode

Normal users usually don't need these commands

SBin command are used to.

- Change system status
- Control hardware
- Modify Kernel behaviour
- Bring the System up.

Example

fdisk

fsck - Checks and repair file system

mkfs - create file system

mount - Attach file system

unmount - Detach filesystem

ip - Configure network

reboot - Restart system

So, Why ~~es~~ /sbin command are critical?

In agine:
- System crashes
- Filesystem is corrupt
- Linux drop into recovery mode

Modern Linux:

/sbin is a symbolic link to /usr/sbin (usr-merge) kept for compatibility.

Measure Question:
Why does sume command requires sudo for some command and some command don't? (being same /sbin command)

∋ There are 2 reasons to decide sudo is required

① Does the command MODIFY the system?

→ If yes → sudo is required
→ If NO → sudo is not required

② Does the Kernel allow non-root access?

→ Kernel enforces permission checks
→ Some operations are root-only by design

For example.

ifconfig
→ Reads network interface information
→ Shows IP address, MAC, status
→ Read-only operation which Kernel allows it

If you do
ifconfig eth0 down
Now, sudo is required because use this command disables a network interface. Affects system networking

→ Modify system state
→ Kernel blocks non-root user

NOTE: IF YOU WANT TO KNOW where command belongs to which just type

whereis ____

For e.g
whereis fdisk

/allshare0/

(4) /lib

/lib contains critical shared library (libraries) needed by:
→ Essential command (/bin, /sbin)
→ The linux Kernel
→ System startup (boot process)

without /lib linux can't even start properly

Example:-

ls

ls doesn't contain everything inside itself.

It depends upon libaries like:

- libc.so (Standard library)
- libselinux.so
- libpthread.so

Those libaries live inside /lib or /usr/lib.

Why does /lib exist seperately

Early boot problem
During boot:
- /usr may not be mounted yet
- But /bin and /sbin must work

so linux need ... libraries available very early
That's why /lib exists

What is stored in stored in /lib?

(1) Shared libraries

Files like:-

/lib/x86_64-linux-gnu/libc-so.6
/lib/x86-64-linux-gnu/libm-so.6
/lib/x86_64-linux-gnu libdl-so-2

These are required by:

- bash
- ls
- mount
- fsck

(2) Kernel modules

/lib/modules/$(uname -r)/

Kernel modules are:

- Drivers
- Filesystem support
- Network driver

Example:

- ext4.ko
- e1000e.ko
- nf_conntrack.ko

lets see libraries used by command

Type :- ldd /bin/ls

Output will be:

libselinux.so.1 => /usr/lib/x86_64-linux-gnu/libselinux.so.1

This proves /bin depends on /lib.

What happens if /lib is broken or missing

⇒ Then,
Commands fail: No such file or directory.

Example Error:

/bin/bash: error while loading shared libraries:
libc.so.6: cannot open shared object file

This is why /lib is critical

/lib vs /lib64

/lib	32-bit or essential file
/lib64	64-bit lib varies

On 64-bit system
/lib64 often exist

Modern Linux uses usr-merge.

/lib is a symbolic link to /usr/lib

/lib → /usr/lib
/lib64 → /usr/lib64

If you type

ls -l /lib

You will see

/lib. → usr/lib

Why normal users rarely touch /lib

- Because
- Editing library can break System
- Libraries are managed by package manager
- Manual changes = danger

Only system and package manager should modify /lib.

(9) /usr — User System Resources

/usr – unix system Resources

It contains:
- Program
- Lib raries
- Documentation
- Shared system Re. sources

Used by all users, not personal file

When /usr is used?

Stage.

After System is up

Daily work

That's why:
/bin, /sbin /lib exist for early boot
/usr/* exist for normal operation

Structure inside /usr
- Run
  ls /usr

You will see directories like
bin sbin lib local man include share

① /usr/bin

It contains
- Majority of linux command

Example
- gcc
- python
- vim
- nano
- curl
- git

In Modern linux /bin → /usr/bin

② /usr/sbin - admin control

It contains
- Ad...

Example
- useradd
- groupadd
- fsck
- fdisk
- mount
- unmount

In Modern linux /sbin → /usr/sbin

(8) /usr/lib

Contains:
- libraries used in program in /usr/bin and /usr/sbin

Example

ldd /bin/python3

You will see library used by python3 from /usr/lib

---

(9) /usr/share — Architecture-independent data

Contains
- Documentation
- Icons
- Time zone
- man page
- local files

(No binaries, only data)

What does Architecture – Independent mean?

- Programs in /usr/bin are "architecture dependent" meaning a version for an intel chip won't work for ARM chip

- Data in /usr/share is universal. A text file, icon, or a timezone look exactly same for all computers regardless of the professor

What lives in there?

→ If you run ls /usr/share/ you will see bunch of folders. Here are the most common.

man	Manual page
icons	Graphics
themes	Visual styles
dict	dictionaries
zoneinfo	Timzone

Why is it so big?

→ It is one of the largest directories on linux.

For example.

Whenever you install a game, the code runs on /bin or usr/bin/ (small), but the texture, sounds, and music go to /usr/share.

Note:
ls /usr/share/zoneinfo/Asia
You will see bunch of city of Asia.

/alish080/

(5) /usr/include — Header file

Used for
- C/C++ development

contains
- stdio.h
- Unistd.h
- stdlib.h

Used by
- gcc
- Compilers

(6) /usr/local - locally installed software

Used for
- Software installed manually
- Not managed by package manager

Structure

/usr/local

If you run ls /usr/local you will see bunch of folders

bin sbin man share src

Why do we have bin and other folders inside /usr/local ?
Lets understand it →

Linux keeps its "Factory-installed" gear seperate from manual-installed gear:

- /usr/bin: The "Factory-Settings", These are the app that came with your Linux Distribution (like ls, apt)
- /usr/local/bin: The "Personal Additions". This is where you compile yourself or install manually; so, they don't get mixed up with the system core file

For example:

You have installed nmap which is (third party scripts) source code and compiled

So, it lives in /usr/local/bin

{ You might see it in /usr/bin if you have installed by doing "sudo apt install nmap" }

Here I see Burpsuite community and packettracer but not nmap.

Why?

⇒ Neither Burpsuite ~~or~~ nor Cisco Packet Tracer are part of the standard "Ubuntu/Debian Factory" (official repositories)

I downloaded Burp Suite Installer from Port Swigger website
And
I Installed : Cisco Packet tracer by downloading .deb file from their website

So, we know when we install "third-party" software that doesn't come from official apt store, the installers are programmed to put the executables in /usr/local/bin.

(6) /etc (System Configuration Directory

→ /etc contains system-wide configuration files

These files contain
→ Control system behaviour
→ Control service
→ Control user, networking, security
→ Are usually plain text

No binaries, no program - only (plaintext) configuration

Why /etc exist

Linux follows this rule:
→ Program lives in /bin and /usr/bin
→ Configuration lives in /etc

It allows:
→ easy customization
→ easy backup
→ Easy troubleshooting

Who uses /etc?

→ System services (sshd, cron, systemd, networking)
→ Package manager
→ Kernel at boot
→ System admin

→ NOTE:
Never delete files from /etc blindly

- Structure inside /etc

Run
[ls /etc]
You will see many files and folders.
such as:
passwd, hostname, pam.conf from a-z
every folder and file.

① User and authentication configuration

- /etc/passwd

Stores:
- Username
- User ID
- Home directories

Run:
If you run /etc/Passwd
→ it won't contains folder
you will see something like

[username : x : UID : GID : comment : home_directory : logic_shell]

Real example:
you will defineetly see this
root : x : 0 : 0 : root : /root : /bin/bash
↓ ↓ ↑UID ↑GID THD → login shell
username pass word stored in /etc/shadow

You will also see like

game:x:5:60:game:/usr/games:/usr/sbin/nologin

and specific for mail, daemon

→ These are for system users
→ Created for services and security isolation

~~too~~ /usr/sbin/nologin

This means
→ User cannot log in
→ Account exist only to run a service

- /etc/shadow

Stores:
→ Encrypted password
→ Password policies

Root-only access

Important
while /etc/passwd lists all the user, it's actually public knowledge - anyone on the system can read it.
etc/shadow/ is where all the real secrete live.

Run:

sudo cat /etc/shadow

you will see lines seperated by colons(:)

mysql : ! : 20488 : : : : : :

Lines represent 9 specific field.

1. Username
2. Encryted Password
3. Last change
4. Minimum Age - How many days must pass before you change
5. Maximum Age - How many days password is valid
6. Warning Period
7. Inactivity Period → Days after expiary before a
8. Expiration date
9. Reserved → left blank for future

- /etc/group

→ Store group information

---

2. System and OS configuration

- etc/OS-release

Run:

cat /etc/os-release

You will see specification of OS of your OS running in machine.

- /etc/fstab

⇒ fstab stands for (File System Table) as your computer's "Mounting Roadmap".

Every time your Linux System boot up, it looks at the file to figure out which "hardware" (disk Partations, USBs, network drives), should be attached to which "software folders" (like /, /home, or /boot).

What is the main job?

⇒ Without this file, Linux wouldn't know where to find your file. It tells the system: "Take Partition A and treat it as root directory, take Partation B and treat it as the home folder".

③ Network configuration

- /etc/hostname

→ System hostname

Run

Cat /etc/hostname

You will see your hostname of machine is

- /etc/hosts

Run
cat /etc/hosts
You can see your local name resolution

---

(4) Service configuration

Each service usually has its own folder

- SSH
SSH is a cryptographical network protocol used to securely operate network services, log into remote systems and execute commands over an unsecured network

SSH is used to :
- log into remote machines
- Manages servers
- Transfer file securely

SSH has two side :
1. Server side (daemon)
2. Client side

Both are configured here

(1) SSH server side configuration

- /etc/ssh/sshd_config or /etc/ssh/ssh_config.d

This controls how your system accepts SSH connections.

(2) SSH Client configuration

- etc/ssh/ssh-config

Controls.

- How this system connects to other SSH servers

(3) SSH host key

If you do ~~cat~~ ls /etc/ssh you will see

- ssh_host_rsa_key
- ssh_host_ed25519_key

These

→ Identify the server
→ Prove server identity to client
→ prevents Man in the Middle.

II Public key & Private key

You will see some files which ends in .pub

- .pub files. These are like padlock. You can give them to anyone.

- Files without any extension. These are your Private key. They are like a Physical key. If anyone steals it they can impersonate as you.

- Corn
Used to
→ Schedules task

Run
ls /etc/coon
You will see something like

- /etc/crontab

This is the "Master schedule" file. If you open it, you will see a specific 7 column-format that looks like a secrete code.

1. Minute (0-59)
2. Hour (0-23)
3. Day of month
4. Month
5. Day of week
6. User
7. Command

Example:
Run
cat /etc/corntab
You will see something like

17 * *** root cd/ && run-parts --report /etc/cron.hourly

let's understand it:

① Timing (17 * ***)

→ 17 here refers to minute

→ * refers to day/Month/Hour (* mean "every")

Meaning: This command runs at the 17th minute of every single hour (1:17, 2:17, 12:17)

② User
root

③ The Command (cd / && run-parts ...)

- cd/: "Change director y to Root". It moves the process to the bigest top of folder tree before starting. This makes sure that script won't get lost if they ... expect to start from the root

- &&: This means only do the next part if first part worked.

- run-parts: This is special utility, Instead of running one file, it run all executable files in ... inside (folder)

- --report: This is a Talkative flag. It tells the system don't print anything to logs unless script actually has an error or something important to say.

/etc/ ... cron.hourly :- This is target Folder

Putting all together

Every hour at 17 minutes past the hour, switch to root directory and run every script found inside /etc/cron.hourly/. If any error or something to ... report, send a note to system logs.

- Systemd

systemd is the Manager, It is where you as the manager has to behave.

Structure of /etc/systemd

Run

ls /etc/systemd

You will see folder (subdirectories)

- /etc/systemd/system/ : This is where you create your own service files or "overrides".
- /etc/systemd/user/ : Configures for services that only run when a specific user logs in
- /etc/systemd/journald.conf : Blackbox recorder for your computer.

For example:-

Suppose you have a service called apache2 (a webserver). The Factory setting say it should start after network is up. But you want to wait until yor External Hard Drive is also mounted.

Instead of editing factory file just create a drop-in folder

/etc/systemd/system/apache2.service.d/override.conf

Inside file just add:

```
[Unit]
After=mnt-mydata.mount
```

→ Override config

When the computer boots, systemd looks into /etc/systemd/system/ and sees override.conf. It then perform merge

(5) Security and pamission

- /etc/sudoers

Controls
  - Who can see sudo

/etc/sudoers files is the most powerful configuration file in your system.

/etc/sudoers: The list of who has the "The power"

Run
sudo cat /etc/sudoers

You will see

root ALL=(ALL) ALL

This is the "GODMODE" line. It ensures that root user can do anything, from anywhere as any one.

Next

%sudo ALL=(ALL:ALL) ALL

% symbol means "Group"

Instead of listing every admins name linux says: "If users is member of sudo group, give them full power".

At bottom you will see

@includedir /etc/sudoers.d

It tells the system "After you read the main file go back to look inside the /etc/sudoers.d folders and apply any extra rule.

- PAM config /etc/pam.d

PAM stands for Pluggable Authentication Module. This is the reason you can log into your linux machine in many different way (password, Fingerprint, SSH key).

Why is PAM needed ?
In old linux, if you wanted to change how people logged in, you have to write the code for login program, the ssh program, and sudo program

With PAM, these program don't handel the authentication themselves. They just ask PAM

Inside PAM :

If you Run ls /etc/pam.d/ you will see bunch of files

- common-auth: The main rule for password
- sudo: The rule specially for when you type sudo
- sshd: The rule for remote login
- passwd: The rule for when you try to change your password

---

let's explore common-auth

Run:

cat /etc/pam.d/common-auth.

You will see lines with four parts.

Example

auth	required	pam_unix.so	nullok
↑	↑	↑	↑
Management group	control flag	Module path	argument

You will also see p lines likes

auth [success=2 default=ignore] pam_unix.so nullok.

→ pam_unix.so: This is the standard module that check your password against usr/shadow/

→ success=2: If you typed your password correctly

Great! skip the next 2 line of code and go the ner success part

→ default = ignore. If it fails, ignore it go the next module to see if that one works

You will also see lines like

auth [success=1 default=ignore] pam_sss.so
use_first_pass

auth requisite pam_deny.so

→ pam_sss.so: This is for the Network logins. If you aren't local user, this module tries to find you on network

→ pam_deny.so: This is the safteynet if both pam_unix and pam_sss fails, the code hits requisite means Game over. Kicks out the user.

→ pam_permit.so: If you avoid made it past pam_deny line, this modules allow you to proceed.