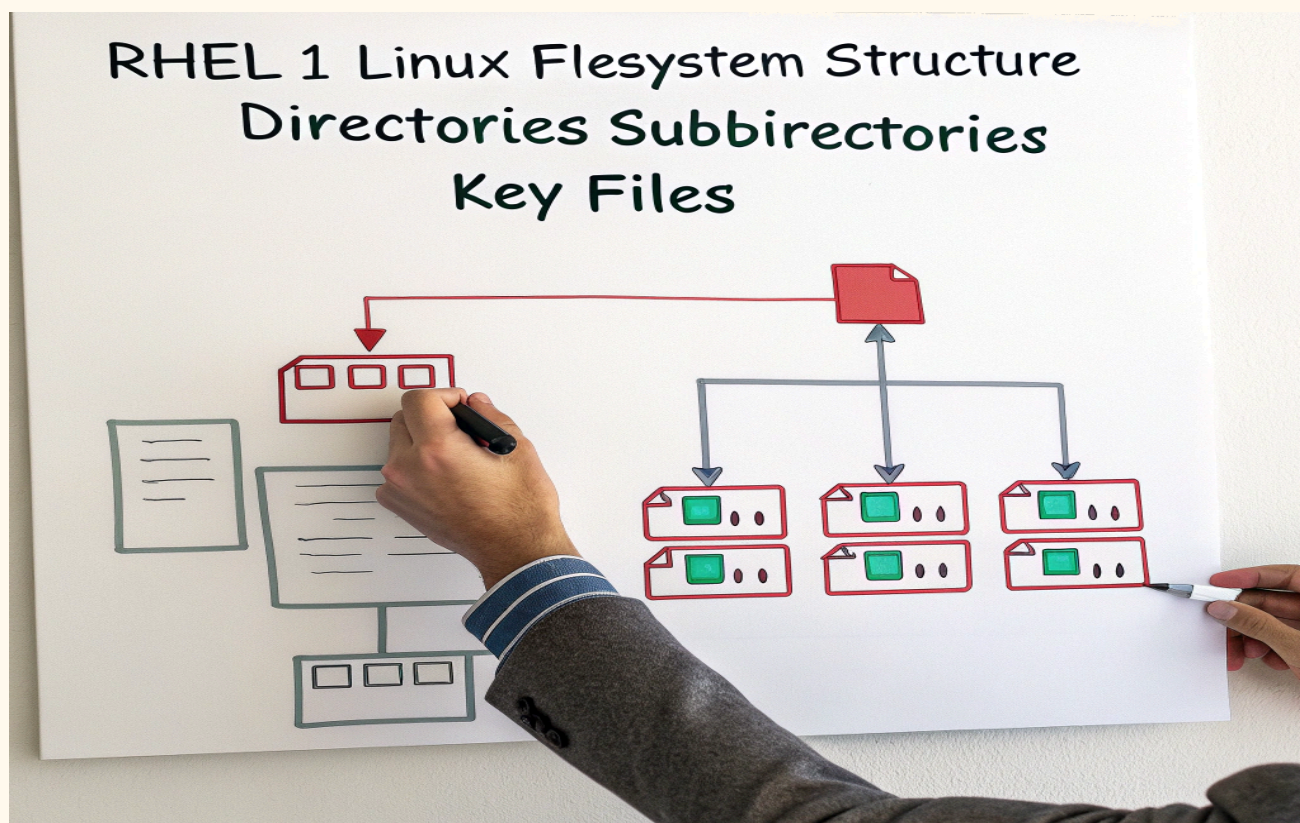




Linux Filesystem Structure

Directories · Subdirectories · Key Files

By Sameer ur Rehman



1. INTRODUCTION

Let's start at the foundation, the Linux filesystem. Everything begins at the root directory `/`, which acts as the single starting point for all files and folders in your system.

From here, Linux builds an elegant, hierarchical structure based on the **Filesystem Hierarchy Standard (FHS)**, a guideline maintained by the **Linux Foundation** that ensures consistency across all major distributions.

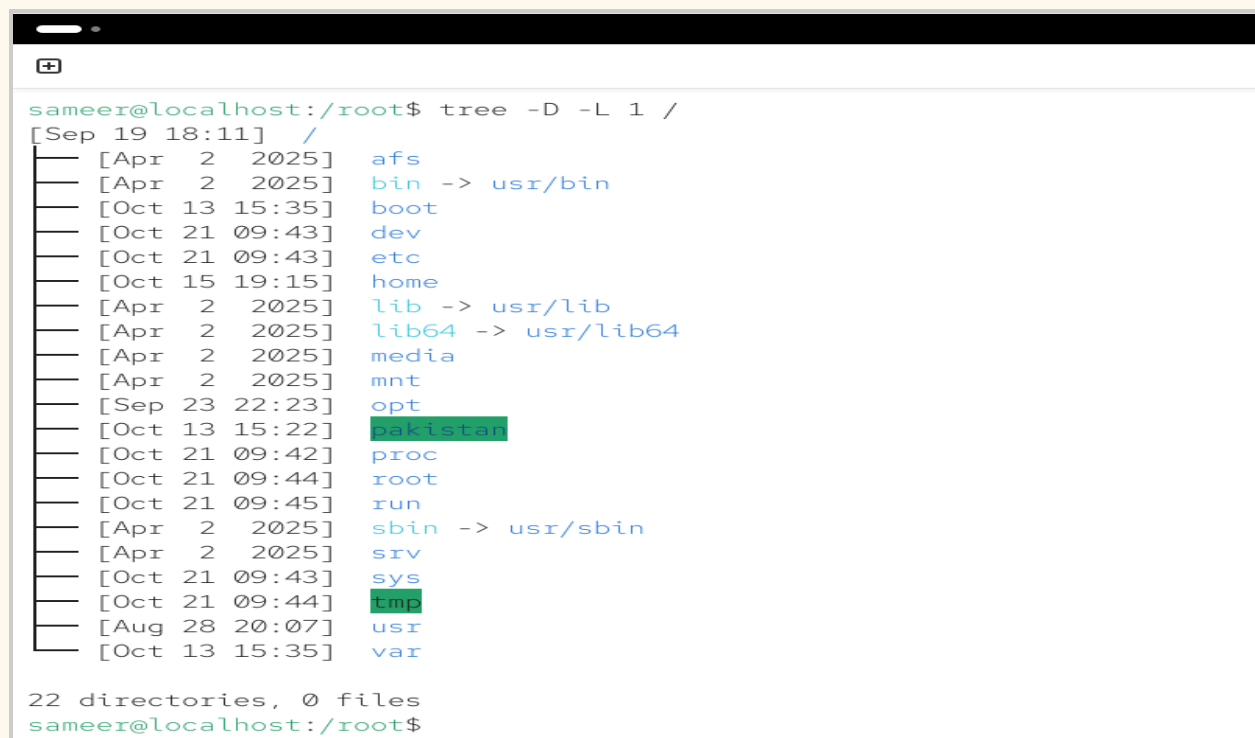
The latest version, **FHS 3.0.3 (June 2015)**, defines how directories should be organized and what each should contain. Understanding this structure makes it easier to navigate, troubleshoot, and configure your system, whether on RHEL, CentOS, or Fedora.

Think of it like a tree: `/` is the root, top-level folders are the branches, and the deeper subdirectories are the leaves. Every file and process ultimately traces back to this single point of origin.

Once you understand this map, you can easily locate configuration files, diagnose issues, and design storage layouts for servers or containers.

To visualize this, open your terminal and run:

```
tree -D -L 1 /
```



```
sameer@localhost:/root$ tree -D -L 1 /
[Sep 19 18:11] /
├── [Apr  2  2025] afs
├── [Apr  2  2025] bin -> usr/bin
├── [Oct 13 15:35] boot
├── [Oct 21 09:43] dev
├── [Oct 21 09:43] etc
├── [Oct 15 19:15] home
├── [Apr  2  2025] lib -> usr/lib
├── [Apr  2  2025] lib64 -> usr/lib64
├── [Apr  2  2025] media
├── [Apr  2  2025] mnt
├── [Sep 23 22:23] opt
├── [Oct 13 15:22] pakistan
├── [Oct 21 09:42] proc
├── [Oct 21 09:44] root
├── [Oct 21 09:45] run
├── [Apr  2  2025] sbin -> usr/sbin
├── [Apr  2  2025] srv
├── [Oct 21 09:43] sys
├── [Oct 21 09:44] tmp
├── [Aug 28 20:07] usr
└── [Oct 13 15:35] var

22 directories, 0 files
sameer@localhost:/root$
```

This lists the first-level directories under `/` — a snapshot of your system’s core layout, as shown in above snapshot.

2. Types of Files in Linux

In Linux, everything is treated as a file. This unified philosophy simplifies interaction between users and the operating system. Files generally fall into three categories:

- **Regular files** – Text documents, executables, images, and scripts.
 - **Directories** – Special files that contain other files or subdirectories.
 - **Special files** – Device nodes, symbolic links, and sockets that represent hardware or communication interfaces.
-

3. WHO THIS TUTORIAL IS FOR

- **RHEL, CentOS, and Fedora learners** building solid fundamentals
- **System administrators** managing production environments
- **DevOps engineers** automating configurations and deployments

This guide helps each group build a clear mental model of how Linux stores, organizes, and accesses data.

4. WHAT YOU’LL LEARN

You’ll understand:

- The purpose of **first-level directories** under `/`

- The role of **important subdirectories** inside `/usr`, `/etc`, `/var`, and others
- The significance of **key configuration and log files** that keep your system running

Mastering these is essential before diving into advanced topics like partitioning, mounting, or service management.

5. LEVEL 1 — TOP-LEVEL DIRECTORIES

At the top level, Linux defines several standard directories, each with a specific role. Together they form the foundation of the system.

Directory	Purpose	Key Examples
<code>/bin</code>	Essential user commands	<code>ls</code> , <code>cp</code> , <code>mv</code> , <code>bash</code>
<code>/sbin</code>	Core admin tools	<code>fsck</code> , <code>mount</code> , <code>reboot</code>
<code>/usr</code>	User programs & shared data	Applications, libraries, docs
<code>/etc</code>	System configuration	Text configs, startup scripts
<code>/var</code>	Variable or changing data	Logs, caches, databases
<code>/lib</code> , <code>/lib64</code>	Shared libraries	<code>.so</code> files
<code>/home</code>	User directories	Personal data and configs
<code>/root</code>	Root's home	Admin files
<code>/boot</code>	Bootloader & kernel	<code>vmlinuz-*</code> , <code>grub.cfg</code>
<code>/dev</code>	Device files	<code>sda</code> , <code>null</code> , <code>zero</code>

<code>/proc</code>	Virtual process info	Kernel runtime data
<code>/mnt</code>	temporary mount point	<code>/mnt/usb</code>
<code>/sys</code>	Virtual hardware info	Device tree structure
<code>/tmp</code>	Temporary files	Auto-cleared on reboot
<code>/run</code>	Volatile runtime data	PID, lock, socket files

Note: `/etc` → “Edit To Configure” · `/var` → “Variable data”.

To check which directories are actual partitions (mount points), use:

```
lsblk -f
```

If a directory appears in the mount list, it’s backed by a separate partition — otherwise, it’s just a folder under `/`.

These directories define the Linux identity. They separate static binaries from dynamic data, configuration files from user content, ensuring predictability across distributions.

6. LEVEL 2 — IMPORTANT SUBDIRECTORIES

While the top level provides a structural overview, it’s the second-level directories where most day-to-day work happens — configuration, binaries, libraries, and logs.

6.1 Inside `/usr`

This is where user applications, documentation, and system utilities live. Historically, `/usr` stood for “user system resources,” not “user home.”

Subdirectory	Description
<code>/usr/bin</code>	Non-essential user commands (<code>vim</code> , <code>git</code> , <code>python3</code>)
<code>/usr/sbin</code>	Admin tools & daemons (<code>sshd</code> , <code>httpd</code>)
<code>/usr/share</code>	Architecture-independent data (docs, icons, locales)
<code>/usr/lib</code> , <code>/usr/lib64</code>	Libraries for <code>/usr</code> programs

6.2 Inside `/etc`

The `/etc` directory is the configuration hub of your system. Red Hat often describes it as the “system brain.”

Each service — from SSH to networking — stores its main configuration files here.

Subdirectory	Description
<code>/etc/ssh/</code>	SSH client/server configs
<code>/etc/systemd/system/</code>	Custom systemd service units
<code>/etc/yum.repos.d/</code>	Repository definitions
<code>/etc/network/</code>	Network settings (distro dependent)

Because Linux follows a “text-based configuration” philosophy, everything here can be viewed or edited with `cat`, `nano`, or `vim`.

6.3 Inside `/var`

The `/var` directory contains files that *change frequently* — logs, caches, mail spools, and databases.

It's often on a separate partition so that log growth doesn't fill the root filesystem.

Subdirectory	Description
<code>/var/log/</code>	System and service logs
<code>/var/lib/</code>	Databases and application data
<code>/var/spool/</code>	Queued jobs (print, mail, cron)
<code>/var/cache/</code>	Cached app data

6.4 Other Notables

Parent	Subdirectory	Description
<code>/boot</code>	<code>/boot/grub2/</code>	GRUB configuration
<code>/lib</code>	<code>/lib/modules/</code>	Kernel modules
<code>/proc</code>	<code>/proc/sys/</code>	Tunable kernel parameters
<code>/sys</code>	<code>/sys/class/</code>	Devices by type (net, block)
<code>/home</code>	<code>/home/<user>/</code>	User dotfiles (<code>.bashrc</code> , <code>.ssh/</code>)

In addition to these, modern distributions also include `/opt`, `/srv`, `/media`, `/mnt`, `/snap`, and `/lost+found`, which handle third-party software, services, removable media, and system recovery fragments.

7. LEVEL 3 — KEY CONFIGURATION & SYSTEM FILES

Once you're familiar with the directory layout, the next logical step is to understand **which files define how Linux actually operates**. These are the configuration and system files that control user access, networking, boot behavior, and services. They're the living blueprint of your operating system.

7.1 `/etc` — Configuration Nucleus

The `/etc` directory is the heart of Linux configuration. Every service, daemon, and core system setting keeps its rules here in plain-text files that can be edited with any text editor. This simplicity is what gives Linux its flexibility.

File	Function
<code>/etc/passwd</code>	User account info
<code>/etc/shadow</code>	Encrypted passwords
<code>/etc/group</code>	Group definitions
<code>/etc/fstab</code>	Filesystem mount points
<code>/etc/hostname</code>	System hostname
<code>/etc/hosts</code>	Local host-IP mapping
<code>/etc/resolv.conf</code>	DNS resolver settings

<code>/etc/sudoers</code>	sudo privileges
<code>/etc/sysctl.conf</code>	Kernel parameter tuning
<code>/etc/ssh/sshd_config</code>	SSH daemon configuration
<code>/etc/httpd/conf/httpd.conf</code>	Apache configuration
<code>/etc/mysql/my.cnf</code>	MySQL settings
<code>/etc/selinux/config</code>	SELinux mode
<code>/etc/default/grub</code>	GRUB defaults

Example:

```
cat /etc/fstab
```

Shows which filesystems auto-mount at boot.

We'll revisit fstab later—it's a crucial file for understanding mounting and storage management.

7.2 `/usr/bin` — User Executables

If `/etc` tells the system what to do, `/usr/bin` provides the tools to do it. This directory contains most of the commands users interact with every day—text editors, interpreters, and utilities.

File	Purpose
<code>vim</code>	Text editor
<code>python3</code>	Python interpreter
<code>git</code>	Version control
<code>curl</code>	HTTP requests
<code>grep</code>	Text search
<code>systemctl</code>	Manage services
<code>journalctl</code>	View logs

Example

`which systemctl`

Displays where the executable resides (commonly `/usr/bin/systemctl`).

In **RHEL 7 and above**, the traditional `service` command was replaced by **systemd**, making `systemctl` the central tool for managing services.

When you run:

```
systemctl status sshd
```

the shell executes the binary `/usr/bin/systemctl`, which:

1. Communicates with the **systemd** daemon (PID 1) through D-Bus.
2. Reads the service unit file `/usr/lib/systemd/system/sshd.service`.
3. Gathers status, PID, and logs from **journald**.

4. Returns the formatted output to your terminal.

In short, `systemctl` is the front end; `systemd` is the real manager orchestrating everything behind the scenes.

7.3 `/usr/sbin` – Administrative Daemons

Moving deeper into system control, `/usr/sbin` hosts many of the administrative binaries and daemons that keep a server alive—services that start automatically and quietly perform background tasks.

File	Function
<code>sshd</code>	SSH daemon
<code>httpd</code>	Apache web server
<code>firewalld</code>	Firewall daemon
<code>useradd</code>	Create users
<code>cron</code>	Job scheduler

Example

```
systemctl status sshd
```

This verifies that the SSH service is running properly. The `/usr/sbin` directory is where real system orchestration happens—security, networking, and automation all begin here.

7.4 /var — Dynamic Runtime Data

If the system were a diary, /var would be its daily logbook. Everything that changes over time—logs, caches, queued tasks—lives here. It’s dynamic, constantly updated, and critical for troubleshooting.

File	Purpose
/var/log/messages	General system log
/var/log/secure	Authentication logs
/var/lib/mysql/	Database files
/var/spool/cron/root	Root’s cron jobs
/var/cache/dnf/	Package cache

Example

```
journalctl -xe      #Displays detailed logs for recent system activity.
```

When things go wrong, /var/log is your first stop. It captures a running history of your system’s behavior—essential for audits and debugging.

7.5 /boot — Boot Essentials

Every Linux startup begins here. The /boot directory contains the kernel and the files needed to load it. Misplacing anything here can make the system unbootable, so it’s small but sacred.

File	Purpose
<code>/boot/vmlinuz-*</code>	Linux kernel image
<code>/boot/initramfs-*</code>	Initial RAM disk
<code>/boot/grub2/grub.cfg</code>	Bootloader configuration

Regenerate GRUB after kernel changes

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

The GRUB configuration defines how your OS boots and what kernel it uses—an essential skill for recovery and troubleshooting.

Don't get confused by command complexity. Most commands are just short forms of English words, and each has a synopsis that makes them easier to understand. Linux itself also provides built-in help commands like `man` and `--help` for detailed explanations — we'll discuss those later.

7.6 `/dev`, `/proc`, `/sys` — Virtual Interfaces

Unlike other directories, these aren't stored on disk—they're generated in memory at runtime. Together they form Linux's live control center, bridging userspace with the kernel.

Directory	Description
<code>/dev/</code>	Hardware devices (<code>sda</code> , <code>tty</code> , <code>null</code>)
<code>/proc/</code>	Kernel and process info (<code>cpuinfo</code> , <code>meminfo</code>)
<code>/sys/</code>	Hardware tree and system topology

Example

```
cat /proc/cpuinfo
```

Displays CPU details straight from the kernel's live data.

These directories let tools like [top](#), [lsblk](#), and [dmesg](#) function by reading kernel-provided information.

8. CONFUSION OVER /bin AND /sbin

New Linux learners often wonder why there are so many “bin” directories. The reason dates back to Unix design logic, where binaries were separated by purpose and permission. While [/bin](#) and [/sbin](#) might look similar, their roles are distinct.

To better understand where different commands live, the table below compares their location, access level, and purpose.

Knowing which directory a command belongs to helps identify its importance and required privileges.

Directory	Type of Commands	Access	Purpose
/bin	Essential user commands	All users	Boot and repair tools
/usr/bin	User applications	All users	Editors, utilities
/sbin	Essential admin commands	Root	Low-level system maintenance
/usr/sbin	Non-essential daemons	Root	Server and admin tools

This separation keeps the system organized and secure. Regular users can run basic utilities from `/bin`, while system-critical commands in `/sbin` are protected for root only.

In modern distributions like RHEL 9, some overlap exists because `/bin` and `/sbin` are often symbolic links to `/usr/bin` and `/usr/sbin`, simplifying the overall layout without breaking legacy paths.

9. CATEGORY REFERENCE TABLE

Each directory in Linux serves a well-defined purpose. Once you see how they group together, navigating the system becomes far easier. The table below organizes them by type so you can quickly recall which are editable, temporary, or virtual.

Category	Example Paths	Editable	Volatile	Purpose
Configuration	<code>/etc/</code>	✓	✗	System settings
Commands	<code>/bin, /usr/bin</code>	✗	✗	Executable binaries
Admin Tools	<code>/sbin, /usr/sbin</code>	✗	✗	System maintenance
Libraries	<code>/lib, /usr/lib</code>	✗	✗	Shared code for apps
Variable Data	<code>/var/</code>	✓	✓	Logs, caches, databases
Temporary	<code>/tmp, /run</code>	✓	✓	Short-term files
Virtual	<code>/proc, /sys</code>	✗	⚡	Kernel/runtime info

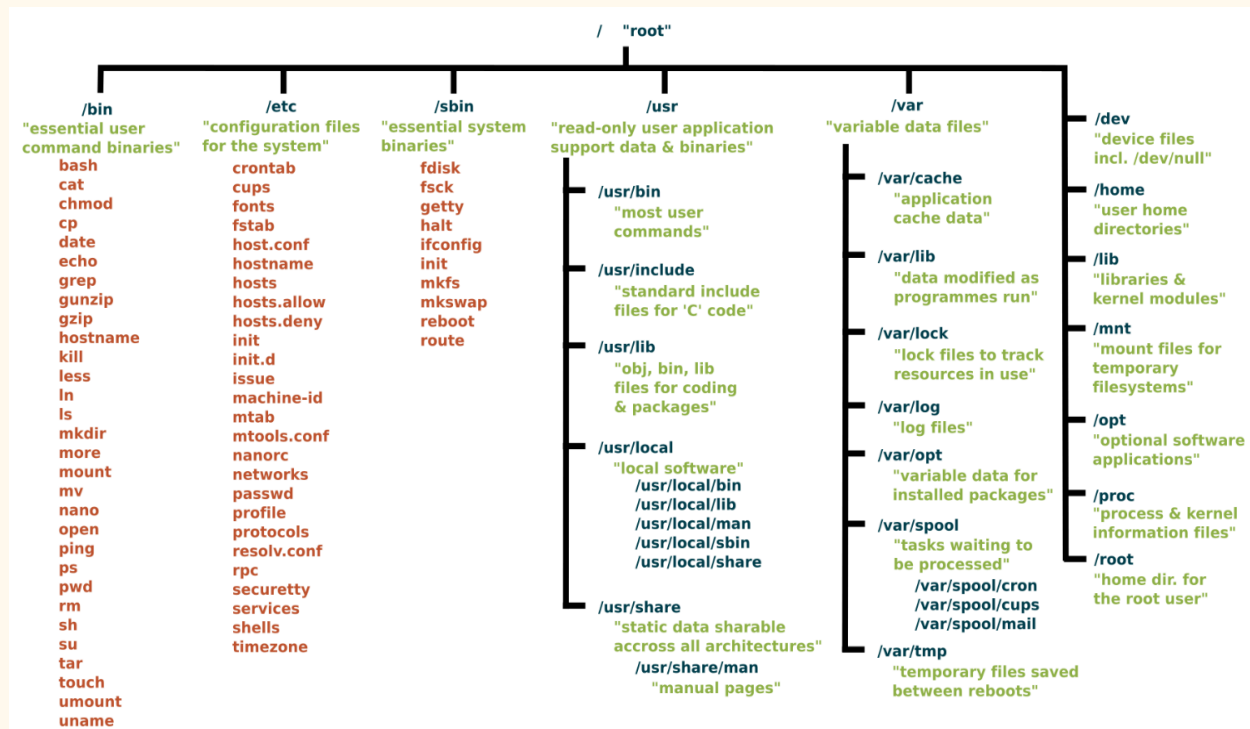
These categories make Linux feel predictable: configs live in `/etc`, code in `/usr`, changeable data in `/var`, and short-lived runtime files in `/tmp` or `/run`. Once this structure clicks, every directory you open will make immediate sense.

Meaning of Terms

- **Editable:**
Means you (or root) can **modify the contents** — like config files in `/etc` or data in `/var`.
 - **Volatile:**
Means the data is **temporary** and may **disappear after reboot** — like `/tmp` or `/run`.
 - **Virtual:**
Means the directory **doesn't exist on disk** — it's created dynamically by the **kernel** to show system or process information (e.g., `/proc`, `/sys`).
 - ⚡ **Volatile (in `/proc` or `/sys`):**
The ⚡ symbol here indicates the data is **runtime-only** — it updates live from memory and **vanishes when the system shuts down or restarts**.
-

10. HIERARCHY OVERVIEW

Let's step back and visualize the big picture. The Linux filesystem is like an inverted tree rooted at `/`. Each branch serves a dedicated purpose, connecting configuration, binaries, logs, and user data into one coherent structure.



/

|— bin/ → core user commands

|— sbin/ → admin commands

|— etc/ → configs (fstab, sshd_config)

|— usr/

| |— bin/, sbin/, share/, lib/

|— var/

| |— log/, lib/, spool/, cache/

|— boot/ → kernel & GRUB

|— dev/, proc/, sys/ → system info

|— home/, root/ → user data

|— tmp/, run/ → temporary data

|— ...

Each branch connects logically — system binaries in `/usr`, configs in `/etc`, runtime data in `/var`.

This predictability is why you can switch between distributions and still feel at home.

11. MEMORY HOOKS

To memorize this structure quickly, keep a few mnemonic cues in mind:

- `/etc` → *Edit To Configure*
- `/var` → *Variable Data*
- `/usr` → *User programs & shared resources*
- `/lib` → *Libraries for commands*
- `/proc` → *Processes*
- `/sys` → *System hardware*
- `/boot` → *Boot files*
- `/home` → *Personal data*

These associations make Linux administration intuitive and efficient—you'll know instinctively where to look, what's editable, and which parts to leave untouched.

12. ACCESS-BASED DIRECTORY SUMMARY

Each directory in Linux has specific **access levels**, determining who can view or modify its contents.

Understanding these access scopes helps you avoid permission errors and safely manage system files.

Directory	Purpose	Example Contents	Access Level
<code>/bin</code>	Essential user commands	<code>ls, mv, bash</code>	All users — accessible for daily operations
<code>/sbin</code>	Administrative commands	<code>fsck, reboot</code>	Root only — system-level maintenance tools
<code>/usr/bin</code>	Installed applications and utilities	<code>vim, git</code>	All users — general software tools
<code>/usr/sbin</code>	System daemons and management services	<code>sshd, httpd</code>	Root only — service management and network daemons
<code>/etc</code>	System configuration files	<code>fstab, sshd_config</code>	Root only — critical settings require admin rights
<code>/var</code>	Logs and variable runtime data	<code>/var/log/, /var/lib/</code>	System — services write here, users read via logs
<code>/lib, /lib64</code>	Shared libraries for binaries	<code>libc.so.6</code>	System — used automatically by programs
<code>/boot</code>	Kernel, GRUB, and boot data	<code>vmlinuz-*, grub.cfg</code>	System/Root only — sensitive for booting process

Key Insight

- **All users:** Safe to use, like `/bin` and `/usr/bin`.
- **Root only:** Configuration or maintenance, e.g., `/etc`, `/sbin`.
- **System-level:** Handled automatically by the OS — avoid manual changes.

Once you grasp who controls what, Linux permissions and directory roles start to make complete sense.

12. SUMMARY

We've now explored the Linux filesystem three levels deep — from directories to subdirectories to the exact files that make your system function.

Understanding this hierarchy is your foundation for every upcoming RHEL concept — mounting drives, configuring services, or managing users.

Every system administrator eventually realizes that filesystem knowledge is *not just structure*, but *power*. It's what allows you to recover from boot errors, optimize servers, and automate configuration confidently.

Next, in **SJ-RHEL 2**, we'll explore more concepts step by step to help you build a logical understanding.

Eventually, all the dots will connect — you'll realize Linux isn't complex; it's a well-engineered system built for clarity and control.