🚀 Day 1: The Absolute Basics

📌 Introduction

Welcome to **Day 1** of your Linux SRE training journey! In this module, we'll combine a **structured, tiered approach** with a **conversational tone**, ensuring that everyone—from true beginners to those with intermediate experience—can understand the essentials of Linux navigation and command-line usage. By the end of this session, you'll see how these basics apply directly to real-world Site Reliability Engineering (SRE) tasks.

Objectives by Tier

Beginner (Tier 1)

1. **Identify** what Linux is and why it's widely used in production.
2. **Recognize** the purpose of a shell and key commands (`pwd`, `ls`, `cd`, `man`).
3. **Demonstrate** simple navigation in the filesystem.

Intermediate (Tier 2)

1. **Apply** additional command options (flags) for more detailed insights.
2. **Locate** and **utilize** help resources (`man`, `--help`, `info`).
3. **Relate** core filesystem structures to daily operational tasks.

SRE-Level (Tier 3)

1. **Diagnose** issues quickly using command-line mastery.
2. **Automate** environment checks for incident response and deployments.
3. **Analyze** filesystem implications for performance, security, and reliability.

Connection to Future Topics: Mastering these fundamentals sets the stage for deeper work in file manipulation, system monitoring, service management, and automation. These advanced skills all build upon the bedrock foundation you'll learn today.

📚 Core Concepts

Below, each concept is broken into four parts: a **Beginner Analogy**, a **Technical Explanation**, an **SRE Application**, and a **System Impact**.

1. Linux

- **Beginner Analogy**: Think of Linux like the engine powering a massive ship—reliable, tested, and the driving force behind most servers around the globe.
- **Technical Explanation**: Linux is an open-source operating system kernel. Different distributions (Ubuntu, CentOS, Debian, etc.) add software around the kernel. It's prized for stability, performance, and extensive community support.

- **SRE Application**: Almost all large-scale infrastructures rely on Linux. As an SRE, you'll handle production servers, containers, and cloud resources all built on Linux's robust foundations.
- **System Impact**: Linux's modular design influences how services run, how you allocate resources, and how you troubleshoot or optimize in high-stakes production.

2. The Shell (Terminal)

- **Beginner Analogy**: A shell is like a conversation partner. You type a request, and it "translates" that into actions the operating system can perform.
- **Technical Explanation**: The shell (like `bash`, `zsh`) interprets your typed commands. It's the direct interface between you and the Linux kernel.
- **SRE Application**: SREs often remotely manage servers via SSH, living in the shell to monitor, troubleshoot, and automate.
- **System Impact**: Shell usage is lightweight, but commands you run can range from harmless queries to full system reconfigurations.

3. Basic Navigation

- **Beginner Analogy**: Navigating a Linux filesystem is like using hallways and rooms in a building. `pwd` shows your location, `ls` shows what's in the room, and `cd` lets you move to a different room.
- **Technical Explanation**: `pwd` (print working directory) reveals your current directory path. `ls` (list) displays items in a directory. `cd` (change directory) moves you between directories.
- **SRE Application**: Quick navigation saves precious time during incidents. You'll jump to `/var/log` to check logs or `/etc` to tweak configurations under tight deadlines.
- **System Impact**: While these commands themselves are low-impact, they guide you to where you can make critical changes (like removing files or editing configs).

4. Getting Help

- **Beginner Analogy**: `man` pages and `--help` text are like instruction manuals. They explain what each button (flag) does.
- **Technical Explanation**: `man <command>` opens the official manual. `<command> --help` gives a quick usage summary. `info <command>` can offer more structured details.
- **SRE Application**: Under pressure, verifying syntax or flags ensures you run the correct command. Mistakes in production can be costly.
- **System Impact**: Checking manuals doesn't change the system, but it prevents errors that might cause big impacts.

5. Filesystem Structure

- **Beginner Analogy**: Linux's filesystem is like a meticulously organized library. Each section (directory) holds a distinct kind of "book" (file).
- **Technical Explanation**: The Filesystem Hierarchy Standard designates where certain types of files should reside. Key directories include `/etc` for configs, `/var/log` for logs, `/home` for user homes, and `/usr/bin` for executables.
- **SRE Application**: Knowing standard directories is crucial for quick incident resolution (e.g., finding logs in `/var/log`).

- **System Impact**: The layout affects performance (log growth on certain partitions), security (permissions in `/etc`), and day-to-day reliability.

💻 Command Breakdown

Below are four core commands (`pwd`, `ls`, `cd`, `man`) plus the overall **filesystem concept**, using a standardized format that highlights usage, syntax, tiered examples, and operational insights.

Command: pwd (Print Working Directory)

Command Overview:
`pwd` shows you your current location in the filesystem. This prevents you from applying changes in the wrong directory—particularly critical in production.

Syntax & Flags:

Flag/Option	Syntax Example	Description	SRE Usage Context
(none)	`pwd`	Prints current (logical) directory path	Basic confirmation of where you are
`-L`	`pwd -L`	Prints the logical directory path, including symlinks	Verifies the symlinked path you're using
`-P`	`pwd -P`	Prints the physical path (no symlinks)	Avoid confusion in scripts by referencing real locations

Tiered Examples:

- ▨ **Beginner Example:**

```
# Example: Just see where you are
$ pwd
/home/student
```

Explanation: You confirm that `/home/student` is your current directory.

- ▨ **Intermediate Example:**

```
# Example: Resolving symlinked path
$ pwd -P
/home/student/documents
```

Explicit context: If `/home/student/documents` is actually a symlink, `pwd -P` shows the true underlying directory.

- 🎯 **SRE-Level Example:**

```
# Example: Checking correct prod directory before a deploy
$ if [[ "$(pwd -P)" == "/var/www/prod" ]]; then
>   echo "Deploying..."
> else
>   echo "Not in production directory! Abort."
> fi
```

Explicit context: This snippet ensures you don't accidentally deploy to staging or a wrong location.

Instructional Notes:

- 🧠 **Beginner Tip:** Use `pwd` any time you're uncertain of your location.
- 🧠 **Beginner Tip:** Pair it with `ls` to see what's around you.
- 🔧 **SRE Insight:** `pwd -P` is particularly useful in scripts to avoid symlink pitfalls.
- 🔧 **SRE Insight:** Always confirm your directory if you have multiple SSH sessions open.
- ⚠️ **Common Pitfall:** Running destructive commands in the wrong directory.
- ⚠️ **Common Pitfall:** Forgetting symlinks can mask the real path.
- 🚨 **Security Note:** Typically low risk, but watch out for printing directory paths in logs that could expose structure.
- 💡 **Performance Impact:** Extremely minimal.

Command: ls (List Directory Contents)

Command Overview:
`ls` displays files and subdirectories. It's your go-to for discovering what's in a directory, checking permissions, and sorting items by size or time.

Syntax & Flags:

Flag/Option	Syntax Example	Description	SRE Usage Context
`-l`	`ls -l`	Long format (permissions, ownership, size, date)	Quickly see who owns files, when they were modified
`-a`	`ls -a`	Show hidden files	Reveal config or hidden "dotfiles"
`-h`	`ls -lh`	Human-readable sizes	Helps identify large files at a glance
`-t`	`ls -lt`	Sort by modification time	Spot recently updated logs or configs
`-r`	`ls -lr`	Reverse order	Invert default listing order

Tiered Examples:

- 🟢 **Beginner Example:**

```
# Example: Basic listing of current directory
$ ls
documents  downloads  readme.txt
```

Explanation: Quickly see what files or directories are there.

- 🟡 **Intermediate Example:**

```
# Example: Checking most recently updated files
$ ls -lt
-rw-r--r-- 1 user user  10240 Mar 29 10:15 error.log
-rw-r--r-- 1 user user  24576 Mar 29 09:59 app.log
...
```

Explicit context: Sorting by time helps pinpoint logs updated just before an issue.

- 🔴 **SRE-Level Example:**

```
# Example: Inspect hidden files, large logs in /var/log
$ cd /var/log
$ ls -lha
-rw-r--r-- 1 root root 100M Mar 29 10:15 syslog
-rw-r--r-- 1 root root  24K Mar 29 09:59 auth.log
-rw-r--r-- 1 root root  512 Mar 29 09:15 .secret_log
```

Explicit context: Large log sizes can indicate an incident or disk space issue.

Instructional Notes:

- 🧠 **Beginner Tip:** Combine flags: `ls -lh` gives both details and easier-to-read file sizes.
- 🧠 **Beginner Tip:** Press the Tab key to auto-complete filenames when typing.
- 🔧 **SRE Insight:** Use `ls -lt | head` to quickly see only the top 10 recently modified files.
- 🔧 **SRE Insight:** Combine `ls` with `grep` to filter results, e.g. `ls -l | grep config`.
- ⚠ **Common Pitfall:** Missing hidden files can mask issues (`.env` files, for instance).
- ⚠ **Common Pitfall:** Listing huge directories can flood your terminal. Use `less` or `head`.
- 🚨 **Security Note:** Hidden files sometimes hold secrets or credentials. Use caution when revealing or copying them.

- 💡 **Performance Impact:** Typically small, but can spike if the directory is extremely large.

Command: cd (Change Directory)

Command Overview:
`cd` allows you to move throughout the filesystem. Whether you're jumping into `/var/log` or your home directory, it's foundational for daily tasks and on-call incident work.

Syntax & Flags:

Flag/Option	Syntax Example	Description	SRE Usage Context
(none)	`cd /var/log`	Moves you to the specified directory path	Quickly access critical logs
`-`	`cd -`	Toggles back to the previous directory	Rapidly switch between two directories while debugging
`~`	`cd ~`	Takes you to your home directory	Return to your personal workspace

Tiered Examples:

- ▨ **Beginner Example:**

```
# Example: Navigating into Documents
$ cd Documents
$ pwd
/home/student/Documents
```

Explanation: You're now in the `Documents` folder.

- ▨ **Intermediate Example:**

```
# Example: Toggling between two locations
$ cd /var/log
$ cd -
/home/student
```

Explicit context: Efficiently jumps between logs and your home directory.

- ▨ **SRE-Level Example:**

```
# Example: Script-based navigation
$ cat deploy.sh
#!/bin/bash
```

```
cd /var/www/production || exit 1
git pull origin main
cd static
./build_assets.sh
cd -
systemctl restart app
```

Explicit context: Automatic navigation ensures consistent steps for an SRE deployment.

Instructional Notes:

- 🧠 **Beginner Tip:** Use `cd ..` to go up one level.
- 🧠 **Beginner Tip:** Type `cd` alone (or `cd ~`) to jump back to your home directory.
- 🔧 **SRE Insight:** In scripts, always handle `cd` failures (`cd <path> || exit 1`) to avoid partial runs in the wrong location.
- 🔧 **SRE Insight:** Use environment variables for directories (e.g., `$LOG_DIR`) to make scripts portable.
- ⚠️ **Common Pitfall:** Typos in path names cause confusion and wasted time.
- ⚠️ **Common Pitfall:** Relying on relative paths in automation can lead to misplaced file operations.
- 🚨 **Security Note:** Be mindful of restricted directories—running `cd` as `root` in the wrong place can be risky.
- 💡 **Performance Impact:** Nearly zero for just navigating.

Command: man (Manual Pages)

Command Overview:
`man` opens the manual pages for a given command, offering detailed info on flags, usage, and examples.

Syntax & Flags:

Flag/Option	Syntax Example	Description	SRE Usage Context
(none)	`man ls`	Opens the manual for `ls`	Quick reference for all flags and usage examples
`-k`	`man -k "search"`	Searches man pages for commands matching a keyword	Discover lesser-known commands relevant to a topic
`-f`	`man -f ls`	Identifies which man section a command belongs to	Differentiates commands with multiple man page sections

Tiered Examples:

- 🟢 **Beginner Example:**

```
# Example: Learning about the pwd command
$ man pwd
```

Explanation: Read official documentation on usage and flags.

- ▨ **Intermediate Example:**

```
# Example: Searching for disk-related commands
$ man -k disk
...
```

Explicit context: You find tools like `fdisk`, `diskutil`, etc.

- ▧ **SRE-Level Example:**

```
# Example: Deep dive into systemd service configuration
$ man systemd.unit
```

Explicit context: SREs often consult advanced sections on service management for reliability.

Instructional Notes:

- 🧠 **Beginner Tip:** Press `q` to exit. Use arrow keys, Page Up/Down to navigate.
- 🧠 **Beginner Tip:** If you see “No manual entry,” try `<command> --help` or install extra docs.
- 🔧 **SRE Insight:** During outages, rechecking exact syntax can prevent damaging commands.
- 🔧 **SRE Insight:** The “EXAMPLES” section in man pages often addresses real-world usage.
- ⚠ **Common Pitfall:** Skimming too quickly and missing crucial flags or warnings.
- ⚠ **Common Pitfall:** Some older or custom commands may lack man pages entirely.
- 🚨 **Security Note:** Some man pages detail security flags or recommended safe practices.
- 💡 **Performance Impact:** Minimal.

Concept: Filesystem Structure

Command/Concept Overview:
While not a single command, understanding the Linux filesystem layout is essential for SREs. Key directories have distinct purposes.

Syntax & Flags:
(Not applicable as a single command, but important references)

Location	Example Path	Description	SRE Usage Context
`/`	*(root directory)*	Top of the filesystem hierarchy	All absolute paths begin with `/`
`/etc`	`/etc/ssh/sshd_config`	System-wide configuration files	Adjusting service or daemon configs
`/var/log`	`/var/log/syslog`	Logs for the system and services	Incident investigation and auditing
`/home`	`/home/sre`	User home directories	Personal user space for scripts, local testing
`/usr/bin`	`/usr/bin/bash`	Common executables	Where most default user-level commands live

Tiered Examples:

- ▨ **Beginner Example:**

```
# Example: Listing root directory
$ cd /
$ ls
bin  boot  dev  etc  home  lib  media  mnt  ...
```

Explanation: You see all the major top-level directories.

- ▨ **Intermediate Example:**

```
# Example: Checking /etc for config files
$ ls -l /etc
-rw-r--r-- 1 root root   3028 Mar  1 12:12 ssh_config
drwxr-xr-x 2 root root   4096 Feb 28 07:45 cron.d
...
```

Explicit context: SREs frequently tweak configurations under `/etc`.

- ▨ **SRE-Level Example:**

```
# Example: Searching logs recursively in /var
$ grep -Ri "ERROR" /var/log/
...
```

Explicit context: You might hunt for errors across multiple logs when diagnosing production issues.

Instructional Notes:

- 🧠 **Beginner Tip:** `/root` is the home directory of the root user, while `/` is the filesystem root.
- 🧠 **Beginner Tip:** Each directory has a specific function; it's not random.
- 🔧 **SRE Insight:** Quick knowledge of `/etc`, `/var/log`, `/home`, and `/usr/bin` is vital for on-call shifts.
- 🔧 **SRE Insight:** Some distributions add or rename directories, but the general structure remains consistent.
- ⚠ **Common Pitfall:** Accidental edits in `/etc` can break critical services.
- ⚠ **Common Pitfall:** Large logs in `/var/log` can fill disk and cause outages.
- 🚨 **Security Note:** Directories like `/etc` and `/var/log` may contain sensitive data. Protect access.
- 💡 **Performance Impact:** Plan partitions wisely (e.g., separate `/var` partition) for better stability.

🛠 System Effects

1. **Filesystem & Metadata**: These commands often query or traverse the filesystem, reading metadata (e.g., permissions, sizes) but not necessarily writing.
2. **System Resources**: Generally low overhead, though listing huge directories or grepping across logs can spike CPU/disk usage.
3. **Security Implications**: Accessing sensitive directories (like `/etc`) or hidden files might expose credentials, so permission checks matter.
4. **Monitoring Visibility**: Most user commands aren't logged by default unless enhanced auditing is enabled. However, changes you make to logs or configs are easily traceable.

🎯 Hands-On Exercises

▨ Beginner Exercises (Tier 1)

1. **Find Your Bearings**
   - **Task**: Open a terminal, type `pwd`, then `ls`. See what files/folders exist.
   - **Goal**: Familiarize yourself with your starting location and visible directories.
2. **Move Around**
   - **Task**: Make a `practice` folder in your home directory (`mkdir practice`), then `cd practice`.
   - **Goal**: Understand directory creation and navigation.
3. **Check Documentation**
   - **Task**: Run `man ls`, skim the man page, then exit by pressing `q`.
   - **Goal**: Explore official documentation on a commonly used command.

▨ Intermediate Exercises (Tier 2)

1. **Detailed Listing**
   - **Task**: Go to `/var/log`, run `ls -lt` to see which files updated most recently. Note the top 1–2 changed logs.
   - **Goal**: Connect file modifications to potential system activity.
2. **Hidden Files**

  - **Task**: In your home directory, run `ls -a` to reveal hidden dotfiles. Open one (like `.bashrc`) with `less`.
  - **Goal**: See how your environment is configured.
3. **Filesystem Exploration**
   - **Task**: Explore `/etc` with `ls -l /etc` to see config file permissions. Identify one file and note its owner, group, and permissions.
   - **Goal**: Understand how config files are protected or shared.

SRE-Level Exercises (Tier 3)

1. **Pre-Deployment Safety**
   - **Task**: Write a script that checks if `pwd -P` matches `/var/www/prod` before pulling code from Git. If not, abort.
   - **Goal**: Prevent accidental deployments in the wrong environment.
2. **Log Triage**
   - **Task**: In `/var/log`, find the largest log files (`ls -lh | sort -hk5`) and search for the string `ERROR` with `grep -Ri "ERROR" .`.
   - **Goal**: Simulate an incident response approach focusing on disk usage and error hunting.
3. **Multi-Directory Audit**
   - **Task**: Create a script that navigates to `/etc`, checks for recent changes, then moves to `/var/log` and reviews the `syslog` or equivalent log. Summarize suspicious lines.
   - **Goal**: Perform a mini-audit by combining navigation, listing, and grep.

Quiz Questions

Beginner (Tier 1)

1. Which command shows your current directory?

   - a) `ls`
   - b) `pwd`
   - c) `cd`
   - d) `man`

2. How do you list all files, including hidden ones?

   - a) `ls -h`
   - b) `ls -l`
   - c) `ls -a`
   - d) `ls -t`

3. In Linux, which directory is the root of the entire filesystem?

   - a) `/home`
   - b) `/root`
   - c) `/`
   - d) `/etc`

Intermediate (Tier 2)

4. To sort files by **modification time** (newest first), which command is correct?
   - a) `ls -lm`
   - b) `ls -lt`
   - c) `ls -lh`
   - d) `ls -la`
5. Which command quickly shows the manual pages for `ls`?
   - a) `ls man`
   - b) `help ls`
   - c) `man ls`
   - d) `ls --man`
6. Which directory typically holds system configuration files?
   - a) `/usr`
   - b) `/etc`
   - c) `/var/log`
   - d) `/bin`

▨ SRE-Level (Tier 3)

7. During an incident, which location do you **most** likely check first to view recent error logs?
   - a) `/home/user`
   - b) `/var/log`
   - c) `/etc`
   - d) `/tmp`
8. What's a recommended step before performing a **production** deploy?
   - a) Run `ls -a` in the logs directory
   - b) Confirm your current directory with `pwd -P`
   - c) Only check the top-level directory with `cd /`
   - d) Do nothing; auto-deploy always works
9. Which hidden files might contain environment-specific data?
   - a) Files with `.env` or `.bashrc` in the name
   - b) Only files ending with `.conf`
   - c) Hidden files never store config data
   - d) `.home` always stores environment data

🚧 Troubleshooting Scenarios

1. **Scenario**: "Permission Denied" when entering `/var/log/app`
   - **Symptom**: `cd /var/log/app` → “Permission denied.”
   - **Possible Cause**: The directory’s ownership or permissions block your user.

- **Diagnostic**: Run `ls -ld /var/log/app` to see who owns it and the permission bits.
- **Resolution**: Adjust permissions with `sudo chown` or `chmod`, if appropriate.
- **Prevention**: Use standardized, documented permissions for application directories.

2. **Scenario**: Symlink Confusion

   - **Symptom**: Deploy script references `/var/www/live`, but the real path is `/var/www/production`.
   - **Possible Cause**: `/var/www/live` is a broken or outdated symlink.
   - **Diagnostic**: `pwd -P` after `cd /var/www/live` to see actual location.
   - **Resolution**: Recreate or fix the symlink: `ln -s /var/www/production /var/www/live`.
   - **Prevention**: Consistent, documented symlink usage and routine checks.

3. **Scenario**: Logs Not Updating

   - **Symptom**: `ls -lt /var/log` shows no recent changes, but the app is presumably running.
   - **Possible Cause**: Logging might be disabled, or logs redirected elsewhere.
   - **Diagnostic**: Check config files in `/etc`, verify the logging driver or location.
   - **Resolution**: Re-enable logging or redirect to the expected path.
   - **Prevention**: Keep a version-controlled record of logging configs; test changes.

❓ FAQ

▨ Beginner FAQs (Tier 1)

1. **Q**: How do I go "up" one directory?

   - **A**: Use `cd ..`. This moves you to the parent directory of your current location.
   - **Real-world application**: Often used when navigating from a project folder back to a more general location.

2. **Q**: Why is Linux often used for servers instead of a GUI-based OS?

   - **A**: Linux is stable, resource-efficient, and secure. Graphical interfaces add overhead that many servers don't need.
   - **Real-world application**: Cloud providers like AWS, GCP, and Azure largely default to Linux for virtualization.

3. **Q**: Is Linux case-sensitive?

   - **A**: Yes, `File.txt` and `file.txt` are two different names. The same goes for commands.
   - **Real-world application**: Inconsistent capitalization can lead to "file not found" errors.

▨ Intermediate FAQs (Tier 2)

1. **Q**: How can I make `ls` show hidden files by default?

   - **A**: Create an alias, e.g., `alias ls='ls -a --color=auto'` in `~/.bashrc` or `~/.zshrc`.
   - **Real-world application**: Speeds up daily checks if you frequently need hidden files.

2. **Q**: What does `man -k "keyword"` do?

  - **A**: It searches all man pages for that keyword, letting you discover commands or tools you didn't know existed.
  - **Real-world application**: If you suspect there's a command that handles "disk," but can't recall the exact name.

3. **Q**: Why do some commands lack a `man` page?

  - **A**: Not all software includes a man page by default, especially minimal or custom tools.
  - **Real-world application**: For in-house scripts, you rely on built-in help or internal documentation.

SRE-Level FAQs (Tier 3)

1. **Q**: How can I quickly swap between two directories while debugging?

  - **A**: Use `cd -`. It toggles between your current directory and the last one.
  - **Real-world application**: Handy when flipping between `/var/log` and `/etc` during triage.

2. **Q**: If I have multiple shells open, how do I keep track of them?

  - **A**: Use unique shell prompts (e.g., include directory info or environment name), or use tabs with descriptive labels.
  - **Real-world application**: Minimizes confusion, especially when you have multiple servers open.

3. **Q**: How do SREs typically address disk space issues caused by huge logs?

  - **A**: Rotate logs (e.g., using `logrotate`), compress old data, or ship logs to external storage.
  - **Real-world application**: This ensures logs don't fill up local disks, avoiding production downtime.

SRE Scenario

Incident: A production web server becomes slow and partially unresponsive. You receive an alert indicating high disk usage.

Steps (5–7) with Reasoning

1. **SSH & Confirm Directory**

```
ssh sre@web-app-prod
pwd
```

Reasoning: Confirm you're on the correct server and see if you start in `/home/sre` or elsewhere.

2. **Navigate to Logs**

```
cd /var/log
```

Reasoning: Application or system logs often reveal the cause of slowdowns.

3. **Inspect File Sizes**

```
ls -lh | sort -hk5
```

Reasoning: Identify which log files are largest, sorting by their size in human-readable format.

4. **Look for Recent Updates**

```
ls -lt
```

Reasoning: Spot logs that have grown rapidly or updated last, possibly indicating new errors.

5. **Search for Errors**

```
grep -Ri "ERROR" .
```

Reasoning: Quickly gather any lines that mention the word “ERROR,” focusing on suspicious activity.

6. **Take Action**

```
mv large_app.log large_app.log.bak_$(date +%F)
echo "" > large_app.log
```

Reasoning: Archiving or clearing the log frees up disk space immediately. Preserves data for analysis.

7. **Verify Resolution**

```
df -h
systemctl status web-app
```

Reasoning: Ensure disk usage is down and the service recovers.

Connection to SRE Principles: Quick navigational skills plus the right checks transform a prolonged outage into a short-lived incident.

🧠 Key Takeaways

1. **Command Summary (5 total)**

   - **pwd**: Checks where you are so you don’t act in the wrong place.

- **ls**: Lists files, reveals hidden items, sorts by size or time.
- **cd**: Moves you around quickly, vital for multi-step incident checks.
- **man**: Shows official documentation, flags, and usage examples.
- **Filesystem structure**: Understanding directories like `/etc`, `/var/log`, and `/home` is fundamental.

2. **Operational Insights (3 total)**

   - Always confirm your directory (`pwd`) in high-stakes environments.
   - Sorting logs by modification time helps track immediate changes during incidents.
   - Knowing standard directories accelerates on-call response.

3. **Best Practices (3 total)**

   - Use `pwd -P` or environment checks in scripts to avoid confusion.
   - Keep logs manageable (rotate, compress) to prevent disk space crises.
   - Familiarize yourself with man pages (`man -k <keyword>`) to discover new or obscure commands.

4. **Preview of Next Topic**

   - **Day 2**: We dive into file creation, manipulation (`touch`, `mv`, `rm`, `cat`, `tail`), permissions (`chmod`, `chown`), and deeper troubleshooting. You'll learn to manage critical config files and logs with confidence.

📚 Further Learning Resources

▨ Beginner (Tier 1)

1. **Linux Journey (Shell Basics)**
   https://linuxjourney.com/lesson/the-shell
   Friendly, interactive tutorials covering shell fundamentals.

2. **The Linux Command Line (William Shotts)**
   http://linuxcommand.org/tlcl.php
   A free, comprehensive ebook for beginners.

▨ Intermediate (Tier 2)

1. **Filesystem Hierarchy Standard**
   https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html
   Official reference on directory structures.

2. **ExplainShell**
   https://explainshell.com/
   Visual breakdown of commands and flags from man pages.

▨ SRE-Level (Tier 3)

1. **Google SRE Book (Emergency Handling Chapter)**
   https://sre.google/sre-book/
   In-depth reliability engineering practices and incident response.

2. **Advanced Bash-Scripting Guide**
   http://tldp.org/LDP/abs/html/
   Comprehensive resource for automating tasks.

3. **Linux Performance Optimization**
   https://www.brendangregg.com/linuxperf.html
   Techniques and tools for analyzing performance at scale.

Congratulations

You've just tackled the foundational commands and filesystem basics every SRE should know. With these core skills, you're well on your way to handling real-world incidents and building up the reliability and efficiency of any Linux-based environment. In **Day 2**, we'll explore manipulating files, analyzing content, and enforcing permissions—the next step toward mastering SRE workflows.