Below is **Version 3** of the **Day 1** Linux SRE training module, merging beginner and intermediate materials into a single comprehensive guide that follows the **v15** guidelines. All content is carefully structured to build knowledge from basic to SRE-level, with realistic examples, exercises, and scenarios.

Day 1: The Absolute Basics — From Novice to SRE

1. Introduction

Linux is the backbone of modern computing, used in everything from embedded devices to large-scale enterprise and cloud infrastructure. For Site Reliability Engineers (SREs), mastering Linux fundamentals is crucial for maintaining high availability, diagnosing system issues, and automating tasks in production environments.

This module focuses on:

- **What is Linux?**
- **What is a terminal/shell?**
- **Basic navigation (pwd, ls, cd)**
- **Getting help (man)**
- **File system structure**

Objectives by Level

Beginner (Tier 1)

1. Understand what Linux is and why it's used.
2. Learn how to navigate directories using basic commands.
3. Know how to access help within the terminal.

Intermediate (Tier 2)

1. Relate basic shell commands to operational tasks (e.g., checking logs).
2. Use options and flags to refine command outputs.
3. Understand the fundamentals of file system organization for troubleshooting.

SRE-Level (Tier 3)

1. Incorporate navigation and help commands into automated scripts and incident response.
2. Assess security and performance implications of file system and shell usage.
3. Integrate command-line workflows with monitoring and logging for reliability.

Connection to Previous & Future Learning

This first day sets the stage for all further Linux exploration. Tomorrow's module will build upon these fundamentals to cover file manipulation (creating, editing, moving, and deleting files), along with more advanced troubleshooting techniques and security considerations.

---

2. **Core Concepts**

In this section, each concept has four sub-parts:

1. **Beginner Analogy**
2. **Technical Explanation**
3. **SRE Application**
4. **System Impact**

2.1 **What Is Linux?**

Beginner Analogy:
Think of Linux as the "engine" powering many cars (servers). You may not see the engine while driving, but it's what makes everything run smoothly.

Technical Explanation:
Linux is an open-source operating system kernel around which distributions (like Ubuntu, CentOS, Fedora) build user interfaces, package managers, and more. It manages hardware resources, schedules tasks, and provides essential interfaces.

SRE Application:
Most cloud services and on-premise servers rely on Linux for its reliability and security. SREs routinely work in Linux environments to deploy services, handle outages, automate tasks, and maintain uptime.

System Impact:

- **Filesystem Structure**: Linux organizes data into directories and files under a single root `/`.
- **Performance**: Linux offers fine-grained control, from scheduling processes to tuning memory usage.
- **Security**: Linux supports robust permission models and SELinux/AppArmor for restricting processes.

2.2 **What Is a Terminal/Shell?**

Beginner Analogy:
A terminal is like a "conversation window" between you and the computer. You type requests (commands), and the computer responds.

Technical Explanation:
The shell (e.g., `bash`, `zsh`) interprets commands typed into the terminal. It sends those commands to the kernel, then displays the output.

SRE Application:
SREs spend substantial time in the shell—checking logs, configuring services, analyzing processes, and scripting actions for automation.

System Impact:

- **Processes**: Each command runs as a process or set of processes.
- **User Environment**: Shell profiles (`.bashrc`, `.zshrc`) define path variables and aliases.
- **Logging & Auditing**: Commands typed into a shell can be logged or audited for compliance.

2.3 **Basic Navigation (pwd, ls, cd)**

Beginner Analogy:
Imagine exploring a building floor by floor. `pwd` tells you which floor you're on, `ls` shows you what's in the room, and `cd` moves you to a different room.

Technical Explanation:

- `pwd`: Prints the current directory path.
- `ls`: Lists files and directories.
- `cd`: Changes your current working directory.

SRE Application:
During incidents, SREs quickly navigate to logs in `/var/log`, or configuration files in `/etc`, to troubleshoot.

System Impact:

- **Filesystem & Metadata**: These commands are read-only (except that `cd` changes your shell's current location but doesn't modify files).
- **Security**: Standard usage requires permission to enter directories, but it's generally non-destructive.
- **Monitoring**: Usage of `ls` or `cd` doesn't typically show up in system resource monitors unless used in scripts or loops.

2.4 **Getting Help (man)**

Beginner Analogy:
"Man" pages are like user manuals for your car—detailed instructions, features, and warnings for each command.

Technical Explanation:

- `man`: Opens the manual pages for a given command.
- `--help`: A quick usage summary for many commands.
- `info`: A hyperlinked documentation system with more in-depth details.

SRE Application:
When under pressure, SREs use `man` or `--help` to confirm complex command options or flags, reducing risk of mistakes.

System Impact:

- **Read-Only**: Accessing documentation has negligible system impact.
- **Security**: Some man pages detail advanced flags that require elevated permissions—knowing them can be critical to safe changes.

2.5 **File System Structure**

Beginner Analogy:
Think of the Linux file system like a well-organized filing cabinet, starting with a main drawer (`/`) and many

labeled folders inside (`/etc`, `/var`, `/home`), each with a specific purpose.

Technical Explanation:

- **Root (`/`)**: The base of the filesystem.
- `/etc`: Configuration files.
- `/var`: Logs, spool files, and other variable data.
- `/home`: Personal directories for each user.
- `/bin`, `/usr/bin`: Essential command binaries.
- `/tmp`: Temporary storage, cleared frequently.

SRE Application:

- Identifying the correct location of logs (`/var/log`) and configs (`/etc`) is crucial in production.
- Knowing how data is organized ensures quick resolution during incidents.

System Impact:

- **Performance**: Partition layout can affect read/write speeds, especially logs or databases.
- **Security**: Permissions on directories like `/etc` or `/var` are critical to prevent unauthorized changes.
- **Monitoring Visibility**: Tools often monitor free space in critical directories (`/`, `/var/log`).

---

3. **Command Breakdown**

Below are **pwd**, **ls**, **cd**, and **man**, each with:

1. **Purpose and SRE Relevance**
2. **Syntax Table with Examples**
3. **Tiered Examples (Beginner, Intermediate, SRE)**
4. **Instructional Notes** (2 tips, 2 insights, 2 pitfalls)

3.1 **pwd**

Purpose & SRE Relevance

- **Purpose**: Show the current working directory.
- **SRE Relevance**: Prevents confusion about where you're operating—especially critical when working on production systems.

Syntax & Examples

Command	Description	Example Output
`pwd`	Prints the current directory	`/home/sre`
`pwd -P`	Prints the physical path, resolving any symlinks	`/mnt/logs`

Tiered Examples

- **Beginner**

1. `pwd` → Displays something like `/home/beginner`.
2. Use `pwd` after `cd /etc` → Shows `/etc`.

- **Intermediate**
  1. `pwd -P` → Shows the actual directory path if symbolic links are involved.
  2. Running `pwd` inside a script to confirm the execution directory.
- **SRE**
  1. Checking `pwd` in a complex automated deployment script where paths might be symlinked.
  2. Using `pwd` output in environment variables to dynamically track working directories.

Instructional Notes

- **Tips**
  1. Always verify your directory before running a high-impact command (e.g., `rm -rf`).
  2. Combine with `echo` to store paths in variables for scripts.
- **Insights**
  1. In Docker containers, `pwd` helps confirm layered directories.
  2. Some production systems use symlinks heavily; `pwd -P` can avoid confusion.
- **Pitfalls**
  1. Relying on `pwd` in partial or incorrectly set `$PATH` scripts might cause confusion in crontab.
  2. Overlooking symlinked directories can lead to unintended file modifications.

3.2 **ls**

Purpose & SRE Relevance

- **Purpose**: List files and directories.
- **SRE Relevance**: Helps quickly locate logs, config files, or check file properties (permissions, timestamps).

Syntax & Examples

Command	Description	Example Output
`ls`	Lists contents of current directory	`file1.txt script.sh folder/`
`ls -l`	Long listing format (permissions, owner, size, date)	`drwxr-xr-x 2 sre sre 4096 Mar 25 10:00 logs`
`ls -a`	Includes hidden files (.)	`.bashrc .gitconfig notes.txt`
`ls -lh`	Human-readable file sizes	`-rw-r--r-- 1 sre sre 1.2K Jan 10 14:10`
`ls -lrt`	Sort by modification time, reverse order (oldest first)	(Oldest to newest in listing)

Tiered Examples

- **Beginner**
  1. `ls` → Simple listing in your home directory.
  2. `ls -l` → Check file permissions and sizes.

- **Intermediate**
    1. `ls -la /var/log` → Find hidden or config-related logs.
    2. `ls -lh /etc` → List system config files with sizes.
- **SRE**
    1. `ls -ltr /var/log | tail -5` → Quickly see the five oldest or newest logs for diagnosing issues.
    2. `ls -lah /path/to/symlinks` → Identify and differentiate symlinks from actual directories.

Instructional Notes

- **Tips**
    1. Combine options: `ls -lah` is a frequent quick-check for file sizes and hidden files.
    2. Use `--color=auto` (if supported) to visually distinguish file types.
- **Insights**
    1. Checking timestamps (`ls -lt`) is key in triaging logs during incidents.
    2. Large file sizes can hint at log rotation issues or unbounded growth.
- **Pitfalls**
    1. Relying purely on `ls` for file existence—some directories might have restricted permissions, causing partial listings.
    2. Accidental usage of `ls *` in huge directories can freeze or flood your terminal.

3.3 **cd**

Purpose & SRE Relevance

- **Purpose**: Change the current working directory.
- **SRE Relevance**: Lets SREs move between key system directories quickly (e.g., `/etc`, `/var/log`, `/home/sre`).

Syntax & Examples

Command	Description	Example
`cd [path]`	Changes directory to `[path]`	`cd /var/log`
`cd ..`	Moves one directory up	From `/var/log` to `/var`
`cd -`	Switches back to previous working directory	Toggles between last two folders
`cd ~` or `cd`	Moves to your home directory	`/home/sre`

Tiered Examples

- **Beginner**
    1. `cd /etc` → Moves to `/etc`.
    2. `cd ..` → Moves one level up.
- **Intermediate**
    1. `cd /var/log && ls -l` → Navigate to logs and list them in one command.
    2. `cd -` → Toggle back to previous directory (handy for quick comparisons).
- **SRE**

1. Using `cd` within a shell script to parse logs in multiple directories automatically.
2. `cd /usr/share/nginx/html && grep -i error *` → Jump into web root and check for application errors.

Instructional Notes

- **Tips**
  1. Use tab-completion to avoid typos in directory names.
  2. Create aliases or environment variables for frequently visited directories.
- **Insights**
  1. Maintaining a correct working directory is essential in scripts or Ansible playbooks.
  2. Some SRE workflows use `pushd`/`popd` for advanced directory stack management.
- **Pitfalls**
  1. Forgetting to verify your new directory before running commands like `rm` or `mv`.
  2. Using `cd` in a script without error handling can cause subsequent commands to fail silently if the directory doesn't exist.

3.4 **man**

Purpose & SRE Relevance

- **Purpose**: Provides detailed manual pages for commands.
- **SRE Relevance**: Critical for quick reference in high-pressure incident scenarios; ensures accurate command usage.

Syntax & Examples

Command	Description	Example
`man [command]`	Opens the manual page for `[command]`	`man ls`
`man -k [keyword]`	Searches for commands related to a keyword	`man -k timezone`
`[command] --help`	Quick usage summary	`ls --help`

Tiered Examples

- **Beginner**
  1. `man cd` → If installed, shows details (some distros treat `cd` as a shell built-in).
  2. `man ls` → Check listing options.
- **Intermediate**
  1. `ls --help` → Quick reference for flags.
  2. `man grep` → Understand advanced pattern matching options.
- **SRE**
  1. `man systemd` → Explore systemd service configurations in detail.
  2. Using `man -k network` → Searching manual pages to discover relevant network commands during an outage.

Instructional Notes

- **Tips**
    1. Press `/` in `man` pages to search for keywords (e.g., `/size`).
    2. Use `Shift+G` to jump to the end of a `man` page quickly.
- **Insights**
    1. Cross-referencing multiple man pages can save time during incidents (e.g., `man journald` + `man systemd`).
    2. SREs sometimes install additional man pages for custom software to ensure consistency in documentation.
- **Pitfalls**
    1. Some commands (shell built-ins or minimal tools) might lack man pages—always try `--help` as a fallback.
    2. Relying on partial reading of man pages can lead to missed details about destructive flags or default behaviors.

---

4. **System Effects**

Even simple commands can affect your system in subtle ways:

1. **Filesystem & Metadata**
    - **`ls`**, **`pwd`**, **`cd`**, `man` are read or metadata checks. They do not alter disk data, but repeated usage in large directories can increment disk I/O slightly.
2. **System Resources**
    - Minimal CPU or memory usage unless listing extremely large directories (e.g., `ls /proc` with thousands of entries).
3. **Security Implications**
    - Access to directories for `cd` or `ls` requires proper permissions.
    - Running `man` pages doesn't require extra privileges but reveals command capabilities (some could be dangerous if misused).
4. **Monitoring Visibility**
    - Typically not visible in system resource monitors unless heavily scripted.
    - Commands show in shell histories, which can be important for audit trails in production environments.

---

5. **Hands-On Exercises**

Each tier has **exactly 3** exercises. Complete them on a real or virtual Linux environment.

5.1 **Beginner Exercises (Tier 1)**

1. **Find Your Home**
    - Open a terminal and run `pwd` to confirm you're in `~` (home directory).

- Then run `ls` to see what's there.

2. **Navigate and List Hidden Files**
   - `cd /etc`
   - `ls -a` to see hidden files and directories within `/etc`.
   - Return to your home directory with `cd ~`.
3. **Explore the Manual**
   - Run `man ls`.
   - Scroll using arrow keys or `Space`.
   - Press `q` to quit.
   - Try `ls --help` to compare the shorter summary.

5.2 Intermediate Exercises (Tier 2)

1. **Examining Log Details**
   - `cd /var/log`
   - Use `ls -l` and `ls -lt` to find the largest or most recently updated logs.
   - Identify logs with suspiciously large sizes.
2. **Config Exploration**
   - `cd /etc`
   - `ls -lh` to check file sizes and permissions.
   - Compare `ls -lah` vs. `ls -lh` outputs to see hidden system config files.
3. **Symbolic Link Check**
   - Move to a directory containing symlinks (e.g., `/etc/alternatives` in some distros).
   - Run `pwd -P` and notice the "physical" path.
   - Investigate differences with a plain `pwd`.

5.3 SRE-Level Exercises (Tier 3)

1. **Directory Stack Management**
   - Use `pushd /var/log && ls` to list logs, then `pushd /etc && ls`.
   - Use `popd` repeatedly to return to previous directories.
   - Note how this can speed up navigating multiple crucial directories during an incident.
2. **Man Page Search During Outage Simulation**
   - Pretend you've forgotten the flags for `netstat` or `ss` command.
   - Use `man -k network` or `man -k socket` to locate relevant commands.
   - Verify how you might discover advanced network troubleshooting tools quickly.
3. **Quick Navigation Script**

- Write a short bash script that moves to `/var/log`, prints the directory, lists files, then returns to `/home/sre`.
- Incorporate `pwd`, `ls -lh`, and a variable to store the old directory.
- Confirm the script works in an environment with symlinks.

---

6. **Quiz Questions**

Below are **3–4 questions per tier**. **Do not** provide the answers inline.

6.1 **Beginner Quiz (Tier 1)**

1. Which command prints your current working directory?
   a) `dir`
   b) `pwd`
   c) `loc`

2. What option would you use with `ls` to see hidden files?
   a) `-h`
   b) `-a`
   c) `-l`

3. Which directory is at the very top of the Linux filesystem hierarchy?
   a) `/root`
   b) `/usr`
   c) `/`

6.2 **Intermediate Quiz (Tier 2)**

1. You suspect a configuration file is hidden in `/etc`. Which command helps you confirm its presence?
   a) `ls -lh`
   b) `ls -la`
   c) `pwd`

2. If you want to see the largest files first in a directory, which combination of `ls` options might help?
   a) `ls -alh`
   b) `ls -lt`
   c) `ls -alS`

3. You used `cd` to move to `/var/log` but discovered you have no permission to list its contents. What likely happened?
   a) The directory doesn't exist
   b) The directory is empty
   c) Your user lacks the required permissions

6.3 **SRE-Level Quiz (Tier 3)**

1. Which command or combination would most efficiently locate the newest rotated log file in `/var/log`?
   a) `ls -al`

b) `ls -ltr`
c) `ls -lS | head -1`

2. How do man pages help during an incident with an unfamiliar tool?
   a) Provide short usage syntax only
   b) Supply detailed options, usage, and examples
   c) They're not helpful; prefer search engines

3. When dealing with symbolic links, which command ensures you see the actual path on disk, not the linked path?
   a) `pwd -L`
   b) `pwd -P`
   c) `pwd --link`

4. If you frequently switch between `/var/log` and `/etc`, which advanced navigation commands might speed up your workflow?
   a) `pushd` and `popd`
   b) `cd -a` and `cd -l`
   c) `pwd` and `ls -a`

---

7. **Troubleshooting**

Here are 3 common scenarios with symptoms, potential causes, diagnostics, resolution steps, and prevention.

7.1 **Scenario 1: Confusion Over Directory Location**

- **Symptom**: You think you're in `/var/log`, but your commands show files unrelated to logs.
- **Cause**: Accidentally typed `cd /var/` or remained in your home directory.
- **Diagnostics**:
  1. Run `pwd` to check your actual path.
  2. `ls` to confirm contents.
- **Resolution**:
  1. `cd /var/log` explicitly.
  2. Re-check `pwd`.
- **Prevention**: Always confirm your directory before destructive or important commands.

7.2 **Scenario 2: "Permission Denied" When Using ls**

- **Symptom**: `ls -l /var/log/secure` fails with "Permission Denied."
- **Cause**: Regular user lacks read permissions on that log, or the log is restricted to `root`.
- **Diagnostics**:
  1. Check file permissions: `ls -l /var/log/secure`.
  2. Attempt `sudo ls -l /var/log/secure` if you have sudo privileges.
- **Resolution**:
  1. Either switch to an account with correct permissions or adjust file permissions carefully (if policy allows).
- **Prevention**: Ensure correct user privileges or log access policies are established.

7.3 **Scenario 3: Symlink Problems in Scripts**

- **Symptom**: Your script references a directory, but the actions happen in another location.
- **Cause**: The directory is a symlink pointing somewhere else.
- **Diagnostics**:
    1. Run `pwd -P` to see the physical path.
    2. Check `ls -l` for `->` indicating symlinks.
- **Resolution**:
    1. Update the script to use the real path or confirm the symlink's correctness.
- **Prevention**: Always verify symlink targets if your script depends on absolute paths.

---

8. **FAQ**

Below are exactly 3 FAQs per tier.

8.1 **Beginner FAQ (Tier 1)**

1. **Q**: Why do I see `~` instead of `/home/username` in my prompt?
   A: `~` is a shortcut for your home directory. `pwd` shows the full path.

2. **Q**: What's the difference between `ls` and `dir`?
   A: `ls` is the standard tool for listing in most distros; `dir` is a similar command but less commonly used. They can behave slightly differently depending on aliases.

3. **Q**: Can I use the mouse in a terminal?
   A: In most basic terminals, you use keyboard navigation. Some terminals support limited mouse interactions (e.g., selecting text), but commands are typed.

8.2 **Intermediate FAQ (Tier 2)**

1. **Q**: Why does `cd` not have a dedicated man page on some systems?
   A: `cd` is often a shell built-in, so its documentation might be found in `man bash` or via `help cd` in `bash`.

2. **Q**: Is there a difference between `ls /home/sre` and `ls /home/sre/`?
   A: Usually no difference in output. However, certain edge cases (like special file types) can behave differently. Typically, they're the same.

3. **Q**: I typed a command but got "command not found." Are my man pages wrong?
   A: Possibly the command is not installed or not in your `$PATH`. Check if the package is installed or consult your environment variables.

8.3 **SRE-Level FAQ (Tier 3)**

1. **Q**: How do I store the output of `pwd` or `ls` in a variable for automation?
   A: Use command substitution:

```
CURRENT_DIR=$(pwd)
FILE_LIST=$(ls -1)
```

2. **Q**: Can these basic commands affect performance on large-scale systems?
   A: Repeatedly running `ls` or scanning huge directories can hog CPU/IO. Tools like `find` or `du` with proper filters might be more efficient.

3. **Q**: How do I handle special characters and spaces in directory names as an SRE?
   A: Use quotes or escape characters: `cd "My Folder"`. For automation, consider renaming such directories to avoid confusion in scripts.

---

9. **SRE Scenario**

Below is a single **detailed incident scenario** with **5–7 command steps** and explanations.

Situation: An application on your production server is failing to start. The alert says "Unable to read config file." You suspect either the config path is wrong or the file is missing.

1. **Confirm Your Location**

   ```
   pwd
   ```

   Reasoning: Ensure you are in the right directory before investigating (`/home/sre` or `/var/app`).

2. **Navigate to Configuration Directory**

   ```
   cd /etc/myapp/
   ```

   Reasoning: Most application config files reside under `/etc/myapp` by convention.

3. **List Config Files**

   ```
   ls -l
   ```

   Reasoning: Check if the main config file (e.g., `myapp.conf`) is missing or has incorrect permissions.

4. **Check Hidden Files**

   ```
   ls -la
   ```

   Reasoning: Confirm there isn't a hidden `.myapp.conf` or other hidden override files.

5. **Use man to Verify Command Options**

```
man myapp
```

Reasoning: Some applications install man pages that describe the config file location. If `myapp` is built-in or doesn't have a man page, you'd check `/etc/myapp/` documentation.

6. **Validate Symlinked Config Path**

```
pwd -P
ls -l /etc/myapp.conf
```

Reasoning: See if the config file is a symlink pointing to a non-existent location.

7. **Check For Permissions**

```
sudo ls -l /etc/myapp/
```

Reasoning: You might need elevated privileges to see or edit certain config files. If permissions are off, fix them so the application can read the file.

By following these steps, you isolate whether the config is missing, misnamed, or permission-blocked—classic issues SREs encounter daily.

---

10. **Key Takeaways**

1. **Command Summary (at least 5)**
   - **pwd**: Know where you are.
   - **ls**: See what's around you.
   - **cd**: Move to relevant directories swiftly.
   - **man**: Access detailed documentation on the fly.
   - **--help**: Quick usage reference.
2. **Operational Insights (3)**
   - Simple navigation can significantly reduce incident response time.
   - Documentation usage (`man`, `--help`) prevents mistakes in critical moments.
   - Proper filesystem understanding is the foundation for advanced debugging.
3. **Best Practices (3)**
   - Always confirm your directory location (`pwd`) before running destructive commands.
   - Combine `ls` flags (e.g., `ls -lah`) for efficient multi-attribute checks.
   - Use caution with symlinks—verify actual paths (`pwd -P`) in production scripts.

4. **Preview of Next Topic**
   Tomorrow, we'll explore **file manipulation**: creating, editing, renaming, moving, and deleting files. We'll also delve deeper into system logs, user permissions, and advanced troubleshooting steps for real SRE workflows.

---

11. **Further Learning Resources**

Here are **2–3** resources for each level, each with a direct link, a short description, and how it applies to that tier.

11.1 **Beginner**

1. **Linux Journey (Command Line Basics)**
   - **Link**: https://linuxjourney.com/lesson/the-shell
   - **What It Teaches**: Fundamentals of the shell, navigation, and basic commands.
   - **How It Applies**: Perfect for newcomers building core familiarity with Linux commands.

2. **The Linux Command Line (William Shotts)**
   - **Link**: http://linuxcommand.org/tlcl.php
   - **What It Teaches**: Comprehensive overview of shell usage, basic scripting, and navigation.
   - **How It Applies**: Helps beginners move beyond just memorizing commands to truly understanding them.

3. **(Optional) Another Beginner-Friendly Guide**
   - **Link**: https://ubuntu.com/tutorials/command-line-for-beginners
   - **What It Teaches**: Step-by-step instructions for absolute new users.
   - **How It Applies**: Reinforces basic usage in a friendly environment.

11.2 **Intermediate**

1. **Linux Filesystem Hierarchy Standard Documentation**
   - **Link**: https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html
   - **What It Teaches**: Detailed explanation of standard directory layout.
   - **How It Connects**: Vital for deeper operational knowledge and troubleshooting.

2. **Linux System Administration Basics (edX Course)**
   - **Link**: https://www.edx.org/course/linux-system-administration
   - **What It Teaches**: Core administrative tasks including user management, logs, package management.
   - **How It Connects**: Bridges from simple navigation to real operational tasks.

3. **DigitalOcean Community Tutorials**
   - **Link**: https://www.digitalocean.com/community
   - **What It Teaches**: Varied topics on server management, best practices, and troubleshooting.
   - **How It Connects**: Great reference for day-to-day operational issues beyond the basics.

11.3 **SRE-Level**

1. **Google's SRE Book – Chapter on Incident Response**
    - **Link**: https://sre.google/sre-book/incident-response/
    - **What It Teaches**: Formal approach to incident management, which heavily relies on command-line agility and systematic debugging.
    - **How It Elevates**: Shows how fundamental commands integrate into large-scale reliability strategies.
2. **Advanced Bash Scripting Guide**
    - **Link**: https://tldp.org/LDP/abs/html/
    - **What It Teaches**: Complex scripting patterns, automation, error handling.
    - **How It Elevates**: SREs frequently automate navigation, file checks, logs analysis via scripts.
3. **Linux Performance Tuning (Brendan Gregg Blog)**
    - **Link**: http://www.brendangregg.com/
    - **What It Teaches**: Deep insights into performance analysis using Linux tools.
    - **How It Elevates**: SREs must maintain reliability under stress; advanced Linux performance knowledge is key.

End of Day 1

You have now built a strong foundation in Linux basics, from navigation to using manual pages. Understanding these core concepts is crucial for anyone looking to excel in SRE, where speed, accuracy, and context matter.

Up Next: Day 2 will focus on **file manipulation**—creating, modifying, moving, deleting files—and we'll continue exploring real troubleshooting methods and reliability engineering principles.