My Linux Command-Line Journey: A Comprehensive Guide

By Sharma Ishika Anup | TYIT Student (Sem 6)

In my journey through IT, I've realized that the terminal isn't just a black box it's a superpower. This documentation serves as a curated map of the Linux commands I've mastered, ranging from basic navigation to process management and user permissions.

1. The Essentials: Navigation & Files

Before doing anything complex, you have to know where you are and how to move.

- **whoami**: Tells me exactly which user I'm logged in as.
- **pwd**: "Print Working Directory" - my GPS for the terminal.
- **ls**: Lists files. I often use ls -la to see hidden files and permissions.
- **cd**: Navigating between folders.
- **mkdir / rmdir**: Creating and removing directories.
- **touch**: The quickest way to create an empty file.
- **open**: Opens a file with the default system application.
- **cp / mv**: Copying and moving (or renaming) files.

2. Reading & Manipulating Text

Data is useless if you can't view or organize it.

- **cat**: Dumps the whole file into the terminal.
- **less / more**: For reading long files one page at a time.
- **head / tail**: Viewing the start or the end of logs (crucial for debugging!).
- **grep**: The ultimate search tool. It finds patterns within files.
- **wc**: Word count, line count, and byte count.
- **sort / uniq**: Organizing data and removing duplicates.
- **diff**: Comparing two files to see what changed.

3. System & Process Management

Managing how the "brain" of the computer works.

- **ps / top**: Monitoring active processes and CPU usage in real-time.
- **kill / killall**: Stopping unresponsive tasks using PIDs or names.
- **jobs / fg / bg**: Managing background and foreground tasks essential for multitasking.
- **history**: A lifesaver for remembering that long command I typed an hour ago.
- **du**: "Disk Usage" --finding out which folder is eating up my storage.

4. Permissions & User Control

This is where the "Admin" side of Linux comes in. Understanding the **Read (4), Write (2), and Execute (1)** logic was a turning point for me.

- **sudo**: Executing commands with "Super User" privileges.
- **chown**: Changing file ownership.
- **chmod**: Modifying permissions (e.g., chmod 755 script.sh).
- **passwd**: Updating user passwords.
- **useradd / groupadd**: Managing system users and their respective groups.

5. Automation & Efficiency

- **alias**: Creating shortcuts for long commands.
- **tar / gzip / gunzip**: Archiving and compressing data for backups.
- **ln**: Creating symbolic links (shortcuts) to files.
- **xargs**: Passing the output of one command as an argument to another.
- **nano**: My go-to terminal-based text editor for quick config changes.

Shell Scripting (The .sh World)

Beyond individual commands, I've been working with **Shell Scripts (.sh)**. This is where automation happens. By combining these commands into a script, I can:

1. Automate backups.
2. Batch rename files.
3. Set up environment variables automatically.

The "Shebang" (#!/bin/bash): I learned that every script starts with this line to tell the system which interpreter to use.

Why am I sharing this?

As a TYIT student, I believe that documenting your learning is just as important as the learning itself. These commands are the foundation of DevOps, Cloud Computing, and Backend Development.