**Recipe: Linux Fundamentals for DevOps**

This recipe uses the **What – How – Why** format to guide you through the most essential Linux fundamentals every DevOps engineer must master:

* File Systems
* Package Management
* Systemd
* Permissions
* Logs
* Disk & Process Management

**🔸 What (Introduction & Theory – Beginner Friendly)**

Think of Linux as the **engine room of modern IT**. Whether you are deploying code in the cloud, managing servers, or working with containers like Docker and Kubernetes — Linux is always running in the background. To control it confidently, you need to understand its **building blocks**.

Let’s break down the key concepts:

**1. File Systems**

* In Linux, **everything is a file** — text files, devices, even processes.
* The file system is like a **tree**, starting at / (called “root”).
* Important directories:
  + /etc → system configuration files (like settings for users, network, and services).
  + /var → variable data, like logs, caches, and mail.
  + /home → where each user stores their personal files.
  + /bin and /sbin → essential system commands and binaries.
  + /dev → represents devices (like hard drives, USBs) as files.
* **Mounting**: Instead of having drive letters like Windows (C:, D:), Linux “mounts” drives into this tree. For example, plugging in a USB may mount it at /media/usb.

👉 Without understanding the file system, you’ll get lost when editing configs, debugging logs, or mounting disks.

**2. Package Management**

* Linux software is distributed as **packages**.
* A package is like a **bundle** of files needed to install software (binary files, configs, docs).
* Each distribution (Ubuntu, CentOS, Fedora) has its own **package manager**:
  + **Debian/Ubuntu** → apt
  + **RedHat/CentOS/Fedora** → yum or dnf
* Package managers make installing/updating software easy. For example:
  + apt install nginx will fetch and install **nginx** plus any extra files it needs.
* Think of this as the **App Store / Play Store of Linux**, but in command-line form.

**3. Systemd (Services & Startup)**

* When Linux boots, it must start many things: networking, logging, background services like ssh, nginx, docker.
* This job is handled by **systemd**, the “init system” of modern Linux.
* It manages:
  + **Services** (e.g., systemctl start nginx)
  + **Boot targets** (runlevels, e.g., “graphical” mode vs. “server” mode).
  + **Logs** (via journalctl).
* Services are defined by **unit files**, usually ending in .service.

👉 In DevOps, knowing systemd is critical because you’ll be starting/stopping services on servers daily.

**4. Permissions**

* Linux is a **multi-user system** — many people (and processes) can share the same server.
* Every file has:
  + **Owner** (who created it).
  + **Group** (a set of users).
  + **Others** (everyone else).
* Permissions are shown as rwx (read, write, execute). Example:
* -rw-r--r-- file.txt

means: owner can read/write, group can read, others can only read.

* Commands:
  + chmod → change permissions.
  + chown → change ownership.

👉 Without permissions, you might get “Permission denied” errors while running scripts or editing configs.

**5. Logs**

* Logs are **history books of your system**. They record what happened, when, and by whom.
* Stored mainly in /var/log/. Examples:
  + /var/log/syslog → general system messages.
  + /var/log/auth.log → login and authentication attempts.
* Modern systems also use **journald**, accessed with journalctl.
* Reading logs is the #1 way to debug issues in Linux.

**6. Disk & Process Management**

* Servers run multiple applications, each consuming CPU, memory, and storage. You must keep them under control.
* **Disk tools**:
  + df → check free space.
  + du → check folder size.
  + lsblk → list disks and partitions.
* **Process tools**:
  + ps → list processes.
  + top/htop → live resource usage.
  + kill → stop runaway processes.

👉 If you don’t monitor disks/processes, a full disk or hung process can crash your production server.

📌 In short:

* **File system** = where things are.
* **Package manager** = how you install things.
* **Systemd** = who starts/runs things.
* **Permissions** = who can access things.
* **Logs** = what happened before.
* **Disk/process management** = how to keep the system healthy.

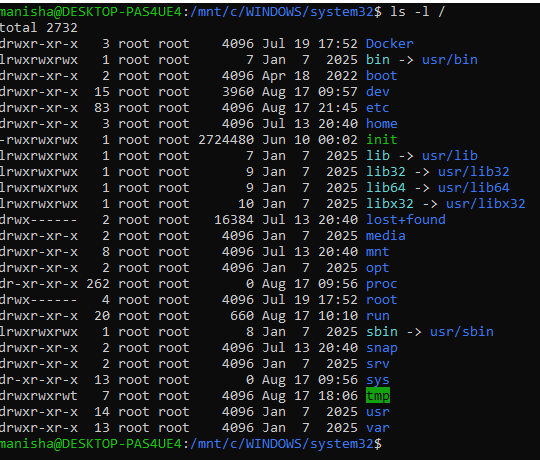
**🔸 How (Step-by-Step with Beginner Explanations)**

**Part 1: File Systems**

**Step 1: View the root directory structure**

**ls -l /**

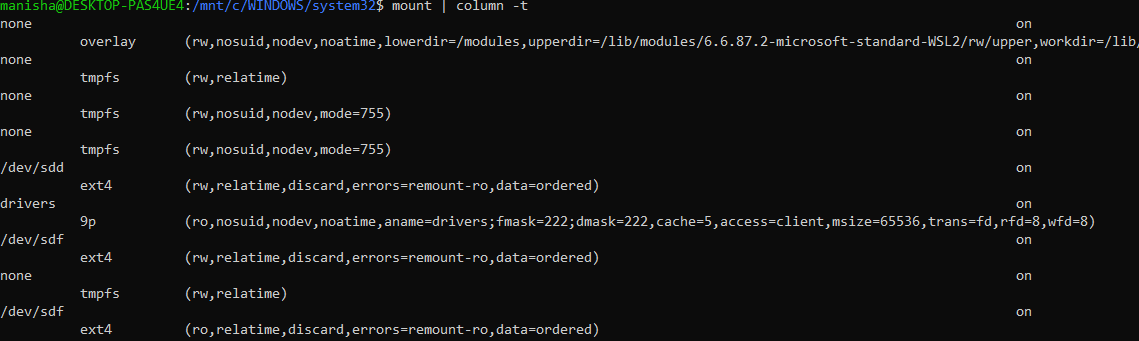
* ls = list files, -l = long format.
* You’ll see folders like /etc, /var, /home.
* Example output:
* drwxr-xr-x 2 root root 4096 etc
* drwxr-xr-x 15 root root 4096 var
* drwxr-xr-x 3 user user 4096 home

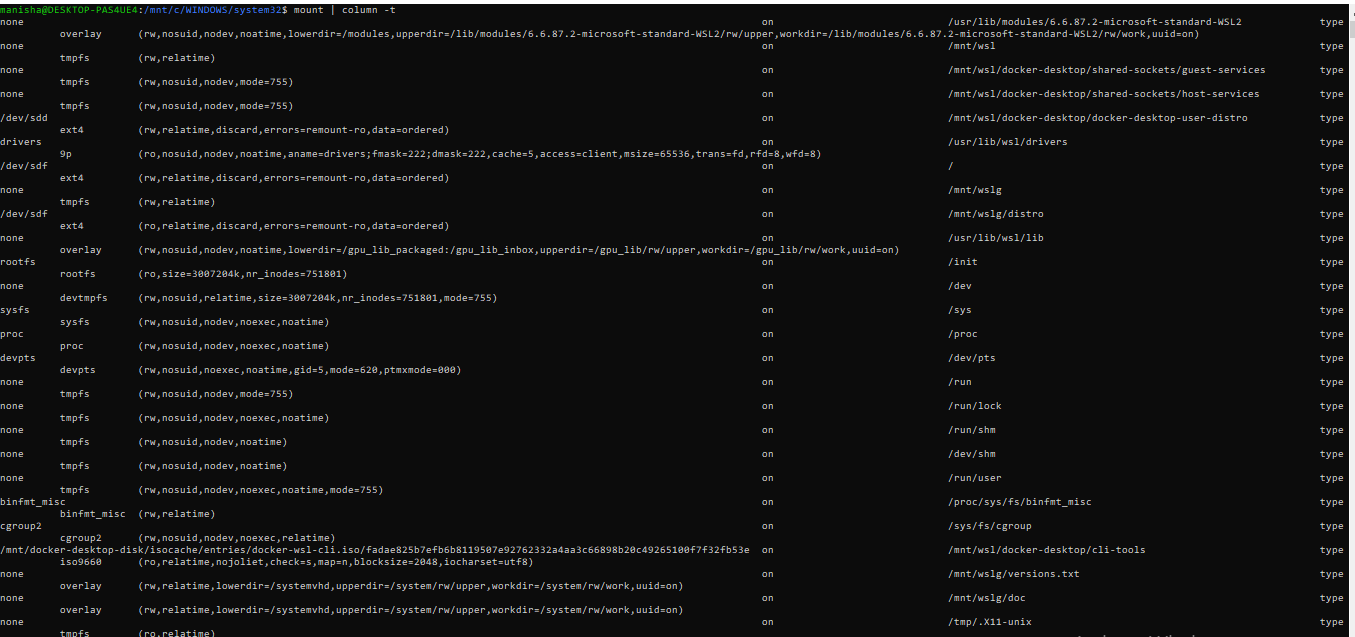


👉 Here d means directory, root is the owner, and numbers show size in bytes.

**Step 2: View mounted file systems**

**mount | column –t**

****

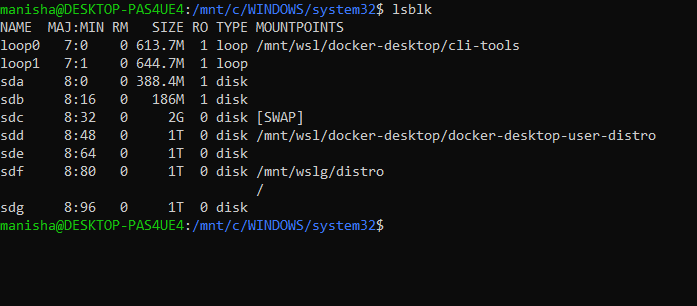
****

* Shows where disks/partitions are “attached” inside the Linux tree.
* Unlike Windows drives (C:, D:), Linux mounts everything under /.

**Step 3: Check disk partitions**

**lsblk**

* Lists block devices (like disks, USBs).
* Example:
* NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
* sda 8:0 0 64G 0 disk
* └─sda1 8:1 0 64G 0 part /

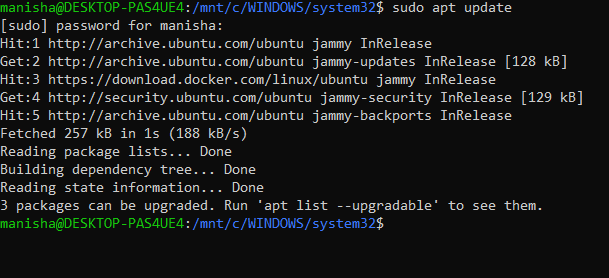


👉 / means this partition is mounted as root.

**Part 2: Package Management**

**Step 4: Update package index (Ubuntu/Debian)**

**sudo apt update**

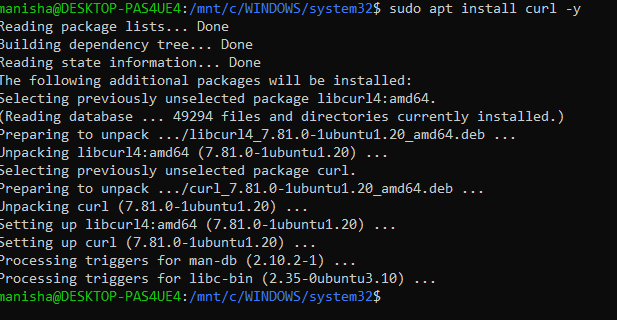
****

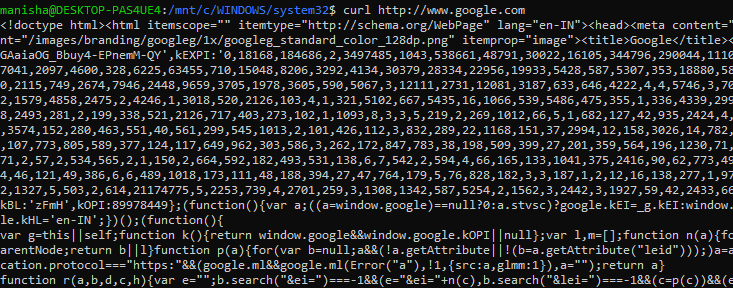
* Refreshes the “package catalog” (like refreshing the App Store).
* Without this, installs may fail.

**Step 5: Install a package (example: curl)**

**sudo apt install curl –y**

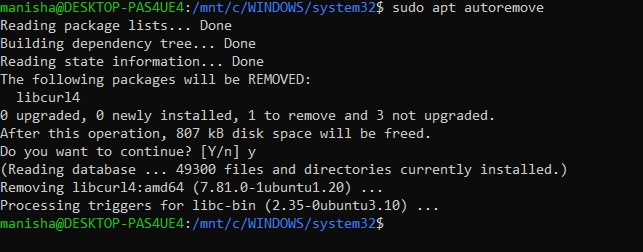
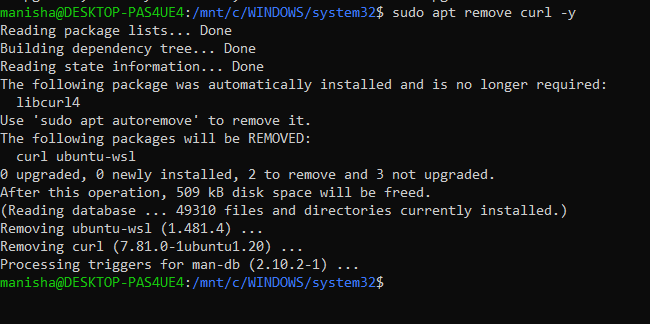
* -y = auto-confirm install.
* Installs curl, a tool to fetch data from the web.
* After install, test with:
* curl <https://example.com>





**Step 6: Remove a package**

**sudo apt remove curl –y**

****

* Uninstalls the software but may keep config files.

**Step 7: (For RHEL/CentOS)**

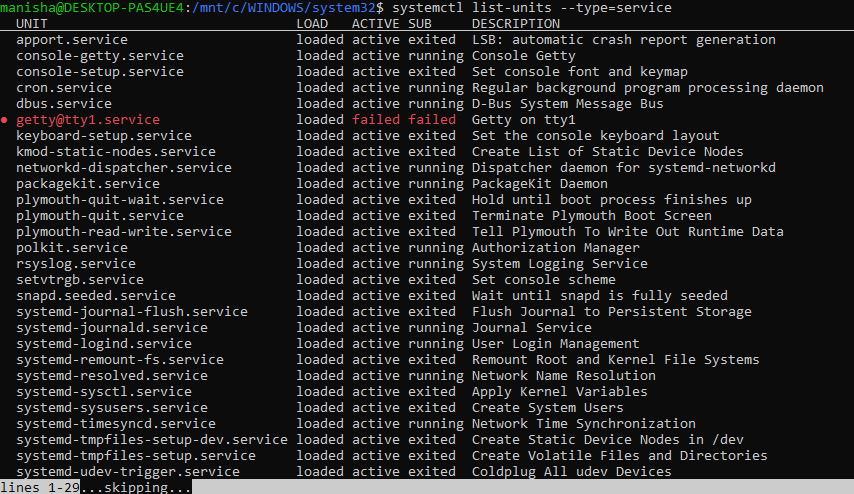
sudo yum install curl -y

* Same as above, but using **yum/dnf** instead of apt.

**Part 3: Systemd**

**Step 8: List running services**

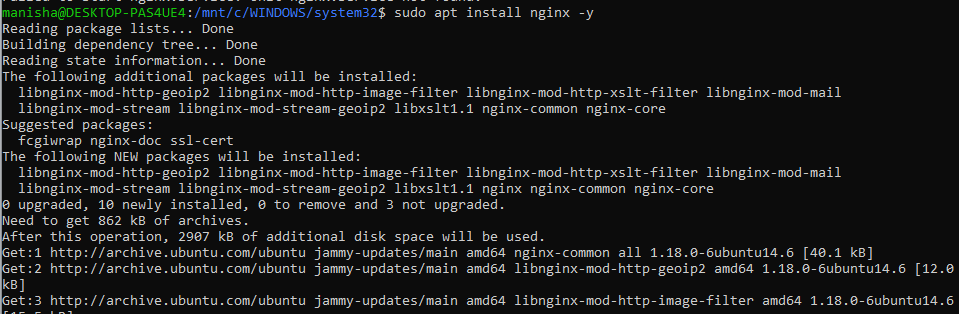
**systemctl list-units --type=service**

****

* Shows all services (running, stopped, failed).

**Step 9: Start and enable a service (example: nginx)**

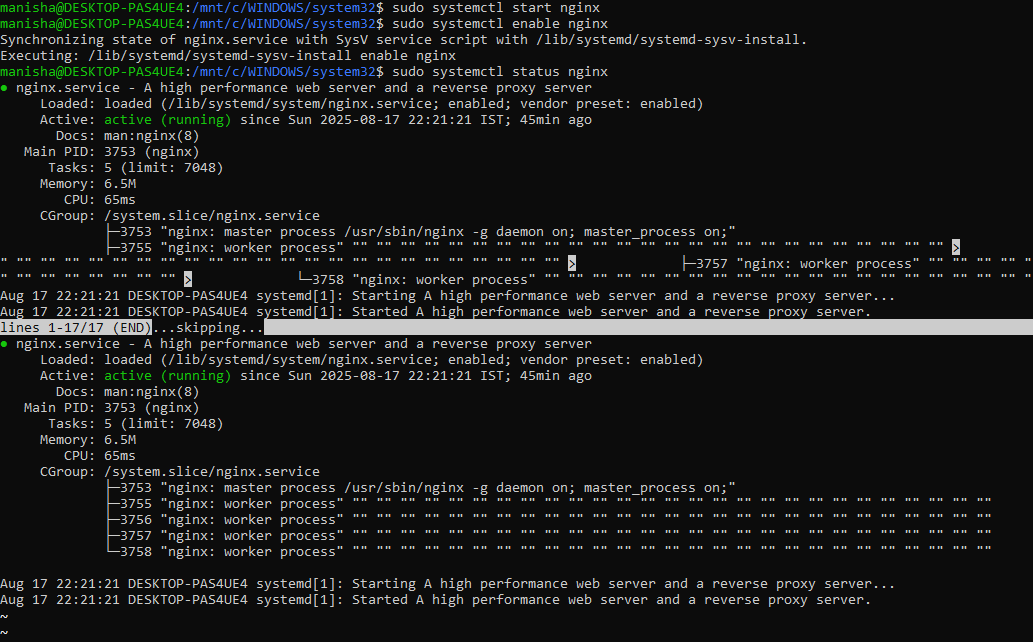
**Install ngnix if not already installed:**



**sudo systemctl start nginx**

**sudo systemctl enable nginx**

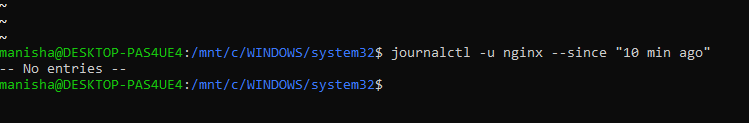
* start = run now.
* enable = run automatically at boot.
* Verify with:
* **systemctl status nginx**



👉 You’ll see logs and whether it’s active (running).

**Step 10: View service logs**

**journalctl -u nginx --since "10 min ago"**

****

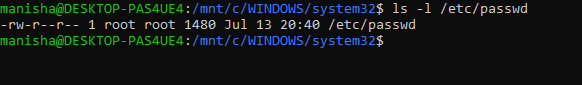
* journalctl is like a **searchable event history**.
* Use --since to filter by time.

**Part 4: Permissions**

**Step 11: Check file permissions**

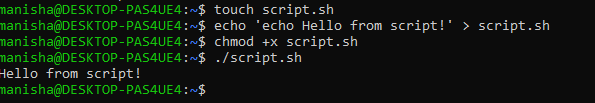
**ls -l /etc/passwd**

* Output looks like:
* -rw-r--r-- 1 root root 2776 Aug 17 passwd
* Means: owner can read/write, others can only read.



**Step 12: Change file permissions (make script executable)**

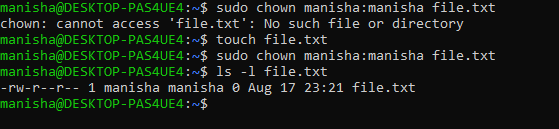
**chmod +x script.sh**

****

* +x = add execute permission.
* Now you can run it with ./script.sh.

**Step 13: Change file ownership**

**sudo chown user:group file.txt**

****

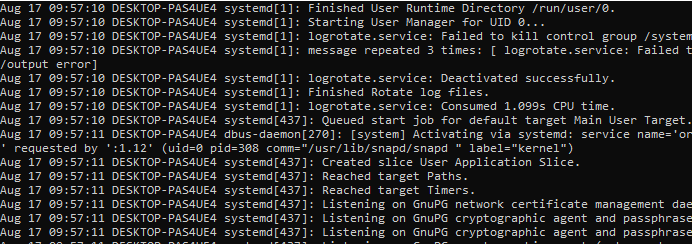
* Example:
* sudo chown ubuntu:ubuntu file.txt
* Changes both owner and group to ubuntu.

**Part 5: Logs**

**Step 14: View system logs**

**less /var/log/syslog**

****

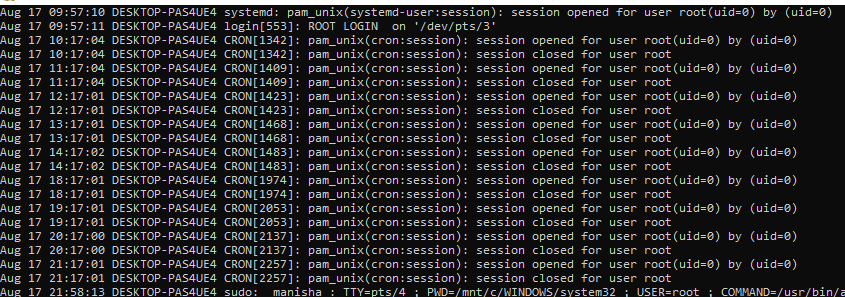
****

* Opens logs in a pager (scroll with space/arrow keys).

**Step 15: View authentication logs**

**less /var/log/auth.log**

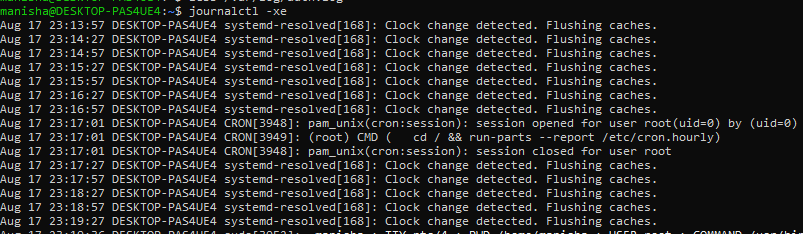
****

****

* Shows successful and failed logins.
* Useful for debugging “why can’t I SSH into this server?”.

**Step 16: View logs with journalctl**

**journalctl –xe**

****

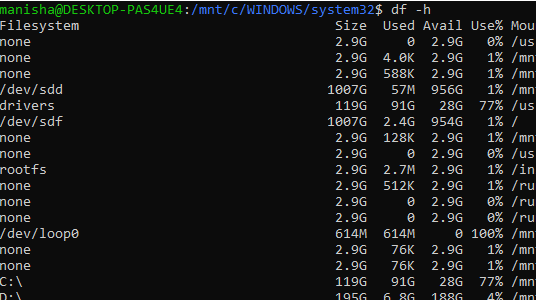
* Shows live error logs with details.
* The -e jumps to the end (latest logs).

**Part 6: Disk & Process Management**

**Step 17: Check disk space**

**df -h**

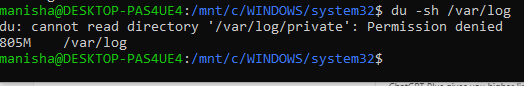
* -h = human-readable (MB/GB).
* Example output:
* Filesystem Size Used Avail Use% Mounted on
* /dev/sda1 64G 23G 39G 38% /



👉 “Use%” is critical. If it hits 100%, your server crashes.

**Step 18: Check directory usage**

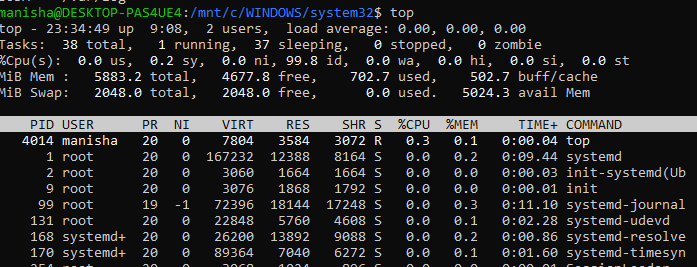
**du -sh /var/log**

****

* du = disk usage, -s = summary, -h = human readable.
* Helps find which folder eats space.

**Step 19: Monitor processes**

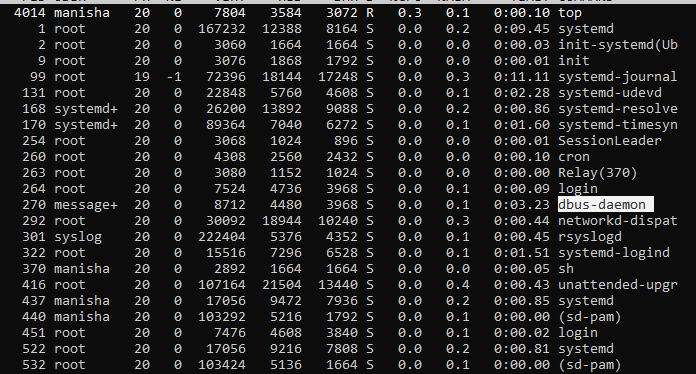
**Top**

****

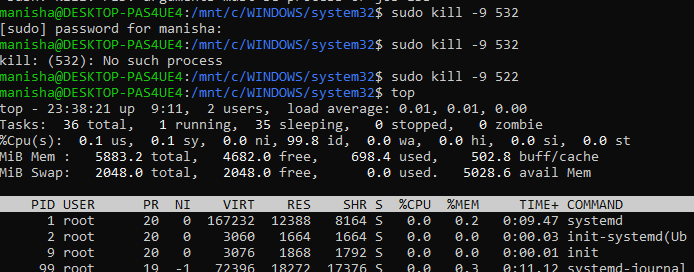
* Shows live CPU, memory usage by processes.
* Keys: q to quit, k to kill a process.
* Advanced alternative: htop (colorful, easier).

**Step 20: Kill a process (replace PID)**

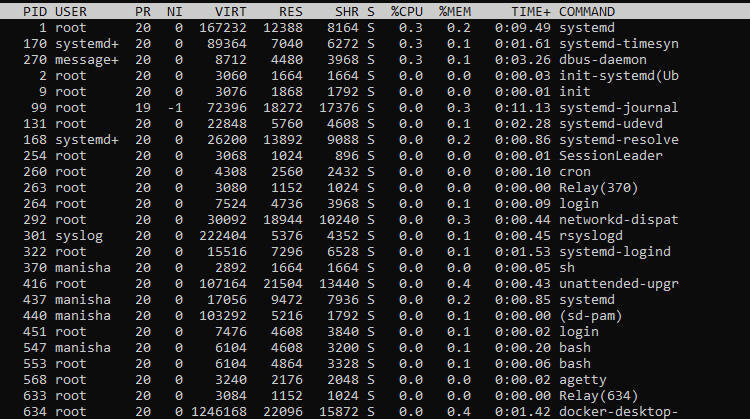
**Before:**

****

**kill -9 PID**

****

**After:**

****

* Example:
* ps aux | grep nginx

kill -9 1234

## 🔸 Why (Practical Thinking & Reasoning)

Now that you know the “what” and “how,” let’s answer **why these topics matter in real-world DevOps work**:

1. **Why File Systems?**

✔️ Because when something breaks in production (e.g., Docker won’t start, or a config is missing), you need to know where to look. If you don’t know /etc holds config files or /var/log holds logs, you’ll waste hours.

1. **Why Package Management?**

✔️ DevOps engineers constantly install tools like Git, Docker, Nginx, Ansible. Doing it manually is error-prone. Package managers (apt, yum) make it safe, fast, and reliable. Without them, automation is impossible.

1. **Why Systemd?**

✔️ Every time you deploy code to a server, it runs as a **service** (nginx, docker, kubelet, jenkins). If you don’t know how to start/stop or check status, you can’t manage servers. Systemd is the **control center** of Linux services.

1. **Why Permissions?**

✔️ Security. If you give the wrong permissions, you might accidentally allow everyone to edit sensitive files (like /etc/passwd). Permissions also fix “Permission denied” errors that new DevOps learners constantly face.

1. **Why Logs?**

✔️ When something fails, logs tell the story. For example: “Why did my deployment fail?” → check /var/log/syslog. “Why can’t I SSH?” → check /var/log/auth.log. Without logs, you’re blind in troubleshooting.

1. **Why Disk & Process Management?**

✔️ Imagine your server runs out of disk space during a deployment → pipeline fails. Or a runaway process eats 100% CPU → app crashes. DevOps engineers prevent this by monitoring disk usage and controlling processes.

👉 In short: These are the **six survival skills** for Linux. Without them, you cannot install software, run services, secure files, debug issues, or keep servers alive.