Linux — Day 4: Filesystems, Strings, Compression, Transfer & Search (Ultimate DevOps Cheat Sheet)

> Friendly, visual, and practical. Each topic includes **what it is**, **must-know commands**, and a **DevOps scenario** so you can apply it in real systems.

TL;DR (One-screen)

- **Filesystem basics:** know paths, inodes, links, mounts, quotas. Use `lsblk`, `mount`, `df -h`, `du -sh *`.
- **Strings & text:** glue commands with pipes. Core tools: `cat/head/tail/less`, `cut/paste/tr`, `sort/uniq/wc`, `sed/awk`, `xargs`.
- **Compression & archiving:** `tar` + (`z` gzip, `j` bzip2, `J` xz, `--zstd`). Verify with `sha256sum`.
- **Transfer:** prefer `rsync -avh --delete` over `scp` for syncs. Use `ssh` keys. `curl`/`wget` for HTTP.
- **Search:** `find` (by name, size, time, owner, perm), `grep -Rin`, `locate`, `file`, `lsof`.

ASCII Visual — Pipelines:

```
[files] --cat--> [stream] --grep--> [filtered] --awk/sed--> [transformed] --sort|uniq--> [report]
```

---

1) Filesystem fundamentals (what lives where)

1.1 Paths & hierarchy

```
/            # root
/bin /sbin   # essential user/system binaries
/etc         # configuration
/home        # user homes
/var         # variable data (logs, spools)
/tmp         # temporary (often tmpfs, cleared periodically)
/srv         # service data
/mnt /media  # mount points
/proc /sys   # virtual kernel/state
```

Commands

```
pwd; ls -lah
cd /etc; ls -1
cd -          # jump back
```

DevOps scenario: decide where to place app data → `/srv/app`, config in `/etc/app`, logs in `/var/log/app`.

1.2 Inodes, file types & links

- **Inode:** metadata holder (owner, mode, timestamps, blocks). Name ↔ inode mapping lives in directory.
- **Hard link:** another name to same inode (same filesystem). Deleting one name doesn't delete data until last link removed.
- **Symbolic link:** pointer path; can cross filesystems; breaks if target missing.

Commands

```
stat file.txt               # inode & metadata
ln file.txt copy.hard       # hard link
ln -s /var/log/app app-log  # symlink
ls -l                       # note leading char: - f, d dir, l symlink
```

DevOps scenario: share a stable path like `/opt/app/current` → symlink to the latest release directory during blue/green deploys.

1.3 Mounts, disks & space

Commands

```
lsblk -f                    # devices, filesystems, labels, UUIDs
df -h                       # mounted FS usage
du -sh *                    # size of items in current dir
sudo mount /dev/sdb1 /mnt   # mount device
sudo umount /mnt            # unmount (no 'n')
mount | column -t           # see active mounts
```

Tip: Persistent mounts → `/etc/fstab` using UUID.

```
sudo blkid
# Example fstab line (ext4):
UUID=<uuid>  /srv/data  ext4  defaults,nofail,_netdev  0  2
```

DevOps scenario: when a node runs low on space, `du -shx / | sort -h` to find heavy paths, rotate/compress logs, or expand volume.

1.4 Ownership, perms & umask (quick refresher)

```
chown user:group path      # change owner/group
chmod 640 file             # rw- r-- ---
chmod -R g+rwX dir         # recursive, +execute only for dirs/already-exec
umask 0002                 # default new files 664/dirs 775
```

DevOps scenario: team-writable artifact directory with setgid: `chmod 2775 /srv/artifacts`.

---

2) Strings & text processing

2.1 Viewing & slicing

```
cat, tac, nl file          # print, reverse, numbered
head -n 20 file            # first lines
tail -n 50 -f logfile      # last lines, follow
less -S logfile            # page (shift+F follows)
cut -d: -f1,7 /etc/passwd  # select columns by delimiter
paste a.txt b.txt          # merge lines side-by-side
tr '[:lower:]' '[:upper:]' # translate chars
```

DevOps scenario: extract usernames and shells → `cut -d: -f1,7 /etc/passwd | column -t` for audit.

2.2 Counting, sorting, uniquing

```
wc -l file                 # line count
sort -h sizes.txt          # human-readable sort
sort | uniq -c | sort -nr  # frequency counts
```

DevOps scenario: top IPs hitting Nginx:

```
awk '{print $1}' access.log | sort | uniq -c | sort -nr | head
```

2.3 Grep & friends (search in text)

```
grep -Rin "error|fail" /var/log -E   # recursive, case-insensitive, extended
regex
rg "ERROR" -n src/                   # ripgrep (fast; pkg: ripgrep)
```

Flags you'll use - `-R` recursive, `-n` line numbers, `-i` ignore case, `-E` extended regex, `-v` invert, `-A/-B/-C` context. **DevOps scenario:** fail-fast in CI logs → `grep -Rin "^FAIL|^ERROR" build/`.

2.4 sed (stream edits) & awk (field logic)

```
sed -n '1,20p' file                          # print range
sed -E 's/(password=).*/\1REDACTED/g' cfg  # mask sensitive values
awk -F, '{sum+=$3} END {print sum}' a.csv   # aggregate column 3
awk '{bytes+=$10} END{print bytes}' access.log
```

DevOps scenario: redact secrets in exported configs before sharing to tickets.

2.5 xargs (bridge args from stdin)

```
find . -name "*.tmp" -print0 | xargs -0 rm -f
printf "%s\n" host1 host2 | xargs -I{} ssh {} uptime
```

DevOps scenario: bulk actions safely with `-print0/-0` for spaces/newlines.

---

3) Compression & Archiving

3.1 tar patterns you actually need

```
# Create archive (directory → tar)
tar -cvf app.tar app/
# With compression (gzip, xz, zstd)
tar -czvf app.tar.gz app/      # gzip (fast, common)
```

```
tar -cJvf app.tar.xz app/      # xz (smaller, slower)
tar --zstd -cvf app.tar.zst app/  # zstd (fast & small)

# Extract
tar -xzvf app.tar.gz           # auto creates dir
tar -xJvf app.tar.xz
tar --zstd -xvf app.tar.zst

# List contents without extracting
tar -tf app.tar.gz
```

DevOps scenario: package a release dir, ship to server, extract atomically in `/opt/app/releases/<ts>`.

3.2 Single-file compressors

```
gzip -9 bigfile && gunzip bigfile.gz
bzip2 bigfile && bunzip2 bigfile.bz2
xz -T0 bigfile && unxz bigfile.xz       # -T0 use all cores
zstd -19 bigfile && unzstd bigfile.zst  # needs zstd
```

DevOps scenario: compress rotated logs before uploading to object storage.

3.3 Integrity & splits

```
sha256sum file > file.sha256
sha256sum -c file.sha256            # verify
split -b 200M backup.tar.gz part_   # split for FAT/limited transports
cat part_* > backup.tar.gz          # rejoin
```

DevOps scenario: verify build artifacts integrity between CI and target hosts.

---

4) Transfer files (secure & fast)

4.1 SSH keys (must-have)

```
ssh-keygen -t ed25519 -C "$USER@$(hostname)"
ssh-copy-id -i ~/.ssh/id_ed25519.pub user@server
```

DevOps scenario: passwordless deploys from CI or your laptop.

4.2 rsync > scp for syncs

```
rsync -avh --progress src/ user@host:/srv/app/          # upload dir
rsync -avh --delete user@host:/srv/app/ ./app-backup/   # mirror (careful)
rsync -avz -e 'ssh -p 2222' ./ user@host:/srv/app/      # custom SSH port
rsync --exclude='.git' --exclude='node_modules' -avh ./ user@host:/srv/app/
```

DevOps scenario: zero-downtime deploy: sync to `/opt/app/releases/ts`, update symlink, `systemctl reload`.

4.3 scp / sftp / tar over ssh

```
scp file user@host:/tmp/
sftp user@host               # interactive
# Tar over SSH (no temp files)
tar -czf - app/ | ssh user@host "tar -xzf - -C /srv/app"
```

DevOps scenario: quick one-off transfer when rsync isn't installed.

4.4 HTTP(S) tools

```
curl -LO https://example.com/file.tar.gz    # download w/ original name
curl -I https://example.com                 # headers only
wget -c https://...                         # continue partial download
```

DevOps scenario: pull binaries or APIs in provisioning scripts.

---

5) Search like a pro

5.1 find (by anything)

```
# by name (case-insensitive), type, and depth
find /var/log -maxdepth 2 -type f -iname "*.log"

# by size/time
find / -xdev -type f -size +1G          # > 1GiB, skip other FS
find /srv -mtime -2 -type f              # modified in last 2 days
```

```
# by owner/perm
find /srv -user www-data -group www-data
find /srv -type f -perm -o+w             # world writable files

# exec safely with NUL separators
find /tmp -type f -name "*.tmp" -print0 | xargs -0 rm -f
```

DevOps scenario: find oversized logs or insecure permissions during audits.

5.2 locate & updatedb

```
sudo updatedb         # refresh database (periodic via cron/systemd)
locate nginx.conf     # blazingly fast name search
```

DevOps scenario: jump to unknown config paths on new servers.

5.3 grep family

```
grep -Rin "pattern" path/
fgrep/grep -F        # fixed string (no regex)
egrep/grep -E        # extended regex
```

DevOps scenario: sift through configs and code for feature flags or credentials to rotate.

5.4 file, strings, lsof

```
file binary           # guess file type
strings binary | head # printable strings in binaries
sudo lsof -i :8080    # which process uses port 8080
```

DevOps scenario: troubleshoot listening ports & binary compatibility.

---

6) Practical Playbooks (copy/paste)

A) Space pressure triage

```
df -h | sort -hk5             # sort by Use%
du -x -h -d1 / | sort -h      # biggest top-level dirs
sudo journalctl --vacuum-time=7d
find /var/log -type f -name "*.log" -size +200M -print
```

B) Compress and ship logs

```
sudo tar --zstd -cvf logs_$(date +%F).tar.zst /var/log/myapp
sha256sum logs_*.zst > logs.sha256
rsync -avh logs_*.zst backup@vault:/backups/logs/
```

C) Blue/Green release (rsync + symlink)

```
TS=$(date +%Y%m%d%H%M%S)
rsync -avh --delete ./build/ user@host:/opt/app/releases/$TS/
ssh user@host "ln -sfn /opt/app/releases/$TS /opt/app/current && sudo systemctl
reload myapp"
```

D) Hunt secrets in repo

```
git rev-list --all | xargs -I{} git grep -I --color -n "AWS_SECRET" {}
rg -n --hidden -g '!node_modules' '(api[_-]?key|secret|token)'
```

E) Find recent large files owned by service

```
find /srv -user www-data -type f -mtime -3 -size +200M -ls
```

---

7) Visual mini-maps

Filesystem growth check

```
[lsblk] -> [df -h] -> [du -shx /path/*] -> [log rotation/compress] -> [resize]
```

Text pipeline

```
log -> grep -i error -> awk '{print $1,$7}' -> sort | uniq -c -> head
```

Transfer

```
[build/] --rsync--> [/opt/app/releases/ts] --symlink--> /opt/app/current --
reload--> service
```

---

8) Packages you may want (Ubuntu names)

- `ripgrep` (`rg`): super fast grep → `sudo apt install ripgrep`
- `fd-find` (binary name `fdfind`): user-friendly `find` → `sudo apt install fd-find` and optionally `ln -s $(which fdfind) ~/.local/bin/fd`
- `eza`: modern `ls` replacement → `sudo apt install eza`
- `jq`: JSON processing → `sudo apt install jq`
- `zstd`: modern compressor → `sudo apt install zstd`

---

9) Quick self-test (1 minute)

1. Create a gzipped tar of `/etc` and list its contents without extraction.
2. Find files in `/var` bigger than 1GiB changed in last 24h.
3. Count unique IPs in `access.log` and print top 5.
4. Sync a local `dist/` to `/opt/www/` on a server with SSH port 2222.
5. Show which process is listening on 9000.

<details> <summary>Answers</summary>

1. `sudo tar -czvf etc.tgz /etc && tar -tzf etc.tgz | head`
2. `sudo find /var -type f -size +1G -mtime -1 -print`
3. `awk '{print $1}' access.log | sort | uniq -c | sort -nr | head -5`
4. `rsync -avz -e 'ssh -p 2222' dist/ user@host:/opt/www/`
5. `sudo lsof -i :9000`

</details>

---

10) DevOps mental models (sticky notes)

- **Streams > files:** think in pipes; create reports from logs without temp files.
- **Idempotence:** `rsync` / `tar` commands should be safe to rerun.
- **Least privilege:** compress/export only what you need; avoid world-writable perms.

- **Observability first:** before deleting large files, check if a process holds them (`lsof +L1`).
- **Verify:** hash artifacts in CI and on target before promoting.

Keep this open during labs. Practice the playbooks and swap tools (`grep` ↔ `rg`, `find` ↔ `fd`) until they're muscle memory.