FIELD GUIDE
EDITION 1.0

FOSSLA.org

A PRACTICAL LINUX & OPEN-SOURCE FIELD GUIDE

RTFM

Read the *FOSS* Manual

```
$ Start with the symptom. Gather evidence.
$ Change one thing at a time.
```



PART I · BASE CAMP
Field manual, 5 chapters

PART II · THE HIGH TRAIL
Reference library, 190 topics

October 2026



> *“The mountain doesn’t care how confident you are. Neither does the kernel. Read the conditions, check your gear, and turn back while turning back is still easy.”*
>
> FROM THE TRAIL



RTFM: Read the FOSS Manual
A practical Linux and open-source field guide.

Published by FOSSLA.org. Edition 1.0, October 2026.

Commands were written for current releases of Debian, Ubuntu, Fedora, RHEL, Arch and openSUSE, and tested with slow, intermittent links in mind. Software changes; check the version you run against its own documentation.

Invite repair. Found an error, a missing distribution note, or a clearer way to say something? Corrections and translations are welcome at FOSSLA.org.

Commands that need root, change the system, or delete data are marked. Run nothing you don’t understand, on a machine whose data you haven’t backed up.

Contents

PART I BASE CAMP · FIELD MANUAL

How to use this manual 5
01 Terminal essentials 7
02 System inspection 8
03 Packages and software 9
04 Troubleshooting playbook 10
05 Open-source practice 11

PART II THE HIGH TRAIL · REFERENCE LIBRARY

How the library works 13
01 Linux foundations 14
02 Shell and terminal 15
03 Filesystems and storage 16
04 Package management 17
05 Services and boot 18
06 Networking basics 19
07 Network services 20
08 Troubleshooting 21
09 Hardware and devices 22
10 Graphics and desktop 23
11 Security foundations 24
12 Cryptography and privacy 25
13 Containers and virtualization 26
14 Development tools 27
15 Automation and scripting 28
16 Data and text 29
17 Servers and operations 30
18 Open-source practice 31
19 Reference and recovery 32

BACK APPENDICES

A Rescue card 33
B Further reading 34

PART I · BASE CAMP

Field manual

A method before a cheat sheet. Like acclimatising at base camp, these five chapters build the habits you need before the climb: read a system, change it safely, and recover when it fails.

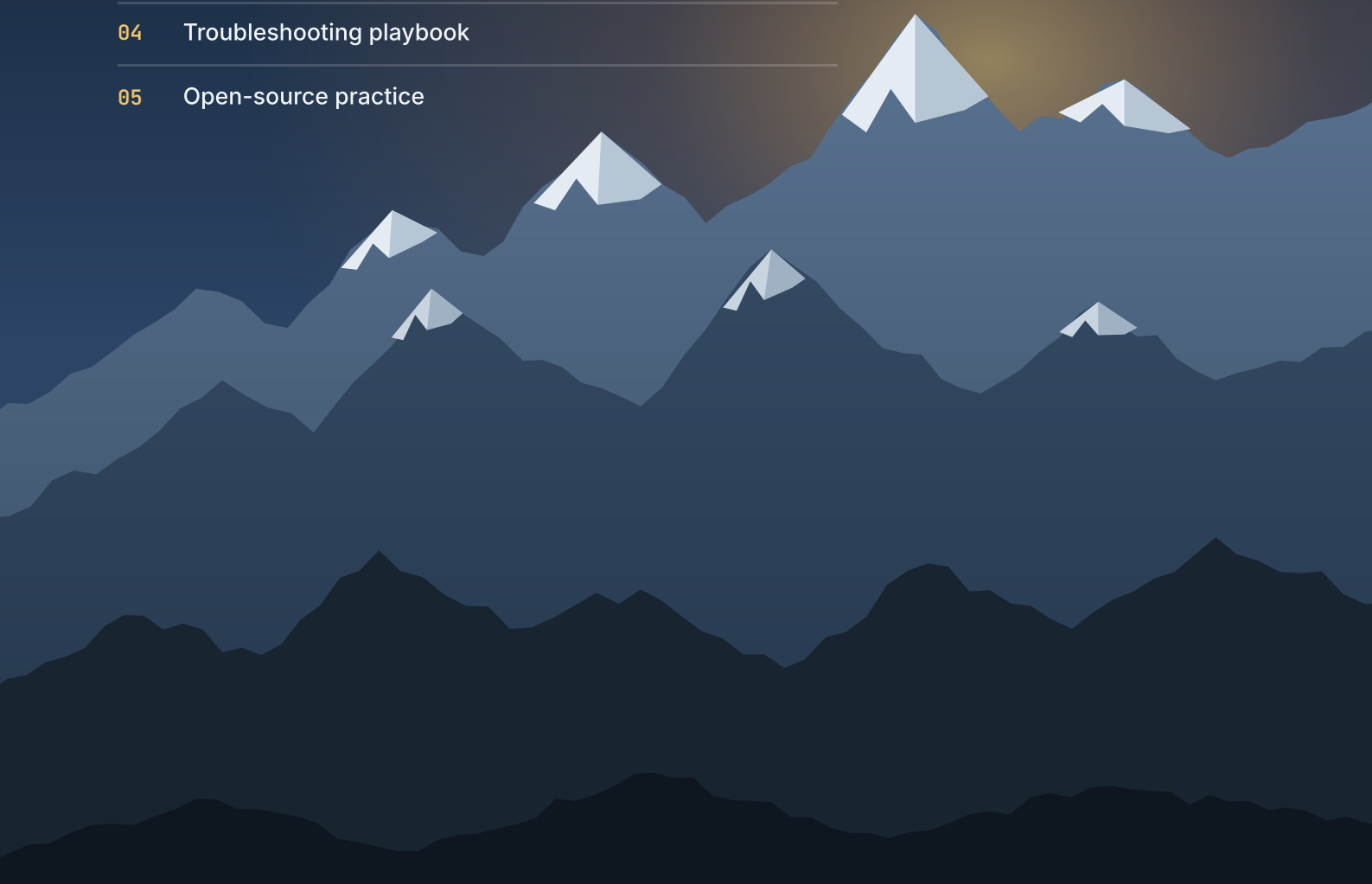
01 Terminal essentials

02 System inspection

03 Packages and software

04 Troubleshooting playbook

05 Open-source practice



How to use this manual

Read it at a desk, keep it open beside a terminal, or reach for it mid-failure. Each page is organised around the moment you are in: *learning* a concept, *doing* a task, or *recovering* from a problem.

In the mountains nobody climbs on a guess. You read the weather, check your gear, keep a line back to camp, and turn around while turning around is still easy. Running a computer well asks for the same discipline.

The method throughout is the same: start with the symptom, gather evidence, change one thing at a time, and never run a command you can't explain. Part I teaches that method with the commands you'll use daily. Part II is a reference library of 190 topics you can open at any page.

Editorial principles

Explain the why A
Every command comes with its purpose and the output you should expect.

Name the risk B
Anything that needs root, changes the system, or deletes data is flagged.

Prefer upstream C
The project's own documentation is the first reference, not a forum post.

Invite repair D
Every page can be corrected, translated or improved by readers.

Notation

- Commands are set in `monospace`. Replace anything in `<angle brackets>` or CAPITALS (`<file>`, `PKG`) with your own value.
- A line starting with `$` runs as your normal user; a line starting with `#` runs as root. Don't type the prompt character itself.
- `sudo` appears only where it is required, and the text says why.

bash

```
$ uname -a # kernel name, version and architecture
$ cat /etc/os-release # which distribution and release you are on
```

The three questions

Ask these before you touch anything. They turn a vague "it's broken" into a search you can finish.

1. **What changed?** A new package, an update, a device, a config file, or something a user did.
2. **What still works?** Find the smallest path that succeeds, then narrow the gap between it and the failure.

3 **What does the evidence say?** Logs and command output beat memory and guesses.

Distribution map

Commands and package names differ by distribution. Check this table before running anything with `sudo`.

FAMILY	PACKAGE TOOLS	CONFIG LOCATION	RELEASE MODEL
Debian / Ubuntu	`apt`, `dpkg`	`/etc/apt/`	Fixed releases; Ubuntu LTS every two years
Fedora / RHEL	`dnf`, `rpm`	`/etc/dnf/`, `/etc/yum.repos.d/`	Fedora about every six months; RHEL long-term
Arch	`pacman`	`/etc/pacman.conf`	Rolling
openSUSE	`zypper`, `rpm`, YaST	`/etc/zypp/`	Leap fixed; Tumbleweed rolling

FROM THE TRAIL

This manual is a starting point. When it disagrees with your distribution's documentation, the distribution wins.

01 Terminal essentials

Learn what each command reads, what it changes, and what it returns, before you combine them.

These nineteen commands cover most daily Linux work. The last column is the one to read first: it tells you whether a command is safe to experiment with.

JOB	COMMAND	WHAT IT DOES	CHANGES ANYTHING?
FILES	`pwd`	Print the directory you are in	No
	`ls -lah`	List all files with sizes, owners and permissions	No
	`cd <dir>`	Move to another directory	No
	`cp <from> <to>`	Copy a file (`-r` for directories)	Yes: may overwrite `<to>`
	`mv <from> <to>`	Move or rename	Yes: may overwrite `<to>`
TEXT	`less <file>`	Page through a file (`q` quits, `/` searches)	No
	`head -n 20 <file>`	First 20 lines	No
	`tail -f <log>`	Follow a log as it grows (Ctrl+C stops)	No
	`grep -R "term" .`	Search every file below here	No
	`wc -l <file>`	Count lines	No
PROCESSES	`ps aux`	Snapshot of every running process	No
	`top`	Live CPU and memory view (`htop` if installed)	No
	`pgrep -a <name>`	Find process IDs by name	No
	`kill <pid>`	Ask a process to stop (sends TERM)	Yes
	`systemctl status <unit>`	State and recent log lines of a service	No
IDENTITY	`whoami` / `id`	Your user, and your user and group IDs	No
	`hostnamectl`	Host name, OS and kernel	No
	`uname -a`	Kernel version and architecture	No
	`cat /etc/os-release`	Distribution name and version	No

CAUTION

`rm -rf` is not a learning command. Use `rm -i` while you learn, run `pwd` first, and keep a backup before deleting anything you care about. `cp -i` and `mv -i` ask before overwriting, too.

Read the output

A command that prints nothing has usually succeeded. When it matters, check the exit status straight after: `0` means success, anything else means failure. For help on any command, try `man <command>`, `<command> --help`, or `tldr <command>`.

bash

```
$ cp report.txt /backup/
$ echo $?
0
```

02 System inspection

Measure before you modify. Five minutes of evidence saves an hour of guessing.

When a machine is slow, full, offline or running hot, collect evidence before you change any configuration.

The five-minute snapshot

Run these in order and save the output to a file. Together they tell you *when*, *how loaded*, *how full*, *how connected*, and *what has already failed*.

#	COMMAND	WHAT IT TELLS YOU	LOOK FOR
1	`date`	When you took the snapshot	Match it to log timestamps
2	`uptime`	Running time and 1/5/15-minute load average	Load above your CPU count (`nproc`)
3	`free -h`	RAM and swap in use	"available" near zero; swap filling
4	`df -h`	Space on each filesystem	Any mount at 90% or more
5	`ip addr`	Network interfaces and their addresses	Interface `DOWN`, or no `inet` address
6	`journalctl -p err -b`	Errors logged since this boot	Repeated errors, and the first one

Save it all in one go:

bash

```
$ { date; uptime; free -h; df -h; ip addr; journalctl -p err -b --no-pager; } \
    > snapshot-$(date +%F-%H%M).txt 2>&1
```

Logs are evidence

Don't paste a whole log into a forum. Share four things: the command you ran, the time window, the relevant lines, and the change that came just before.

REDACT FIRST
Remove passwords, API tokens, private IP addresses, hostnames you don't want public, and any personal data.

FROM THE TRAIL
The fastest fix is often a rollback. Undo the newest change, confirm the symptom goes away, then bring changes back one at a time.

03 Packages and software

Install from a source you can name, and know how its updates will reach you.

Use your distribution's repositories or the upstream project's documented release channel. Avoid `curl … | sudo bash` scripts, unsigned binaries, and any instructions that hide what runs as root.

TASK	DEBIAN / UBUNTU	FEDORA / RHEL	ARCH	OPENSUSE	FLATPAK
Refresh lists	`sudo apt update`	`sudo dnf check-update`	Only as part of `sudo pacman -Syu`	`sudo zypper refresh`	`flatpak update --appstream`
Search	`apt search TERM`	`dnf search TERM`	`pacman -Ss TERM`	`zypper search TERM`	`flatpak search TERM`
Show details	`apt show PKG`	`dnf info PKG`	`pacman -Si PKG`	`zypper info PKG`	`flatpak info APP`
Install	`sudo apt install PKG`	`sudo dnf install PKG`	`sudo pacman -S PKG`	`sudo zypper install PKG`	`flatpak install REMOTE APP`
Remove	`sudo apt remove PKG`	`sudo dnf remove PKG`	`sudo pacman -R PKG`	`sudo zypper remove PKG`	`flatpak uninstall APP`
Upgrade all	`sudo apt upgrade`	`sudo dnf upgrade`	`sudo pacman -Syu`	`sudo zypper update` (Tumbleweed: `sudo zypper dup`)	`flatpak update`
Who owns a file	`dpkg -S /path`	`rpm -qf /path`	`pacman -Qo /path`	`rpm -qf /path`	n/a

Why `sudo`? Installing and removing write to system directories that only root can change. Searching and reading package details never need it.

ARCH WARNING

Never run `pacman -Sy PKG` on its own. Installing against a fresh package list without upgrading the rest of the system creates a partial upgrade that can break libraries. Use `pacman -Syu PKG`.

HIGH-ALTITUDE TIP

On a slow, metered or intermittent link, update during the quietest hours, keep one machine as a package cache (`apt-cacher-ng` for APT; a shared `/var/cache/pacman/pkg` for Arch), and let downloads resume: `apt`, `dnf` and `pacman` all retry partial downloads when you run them again.

Check provenance before you install

A package name is not proof of where software came from. Before adding anything outside your main repositories, check:

- [ ] **Repository.** Is it the official one, or a third-party repo, PPA or COPR you'd now trust for every future update?
- [ ] **Maintainer.** Is the publisher the actual upstream project?
- [ ] **Signature.** Is the package or repo GPG-signed, and does the key match the project's published key?
- [ ] **Permissions.** For Flatpak and Snap, what file, device and network access does it ask for?
- [ ] **Update path.** How will security fixes reach you after today?

04 Troubleshooting playbook

Recover without guessing: five steps, one change per pass, undo what didn't help.

Work through the loop below. Change exactly one thing per pass, and undo anything that didn't help before you try the next idea.

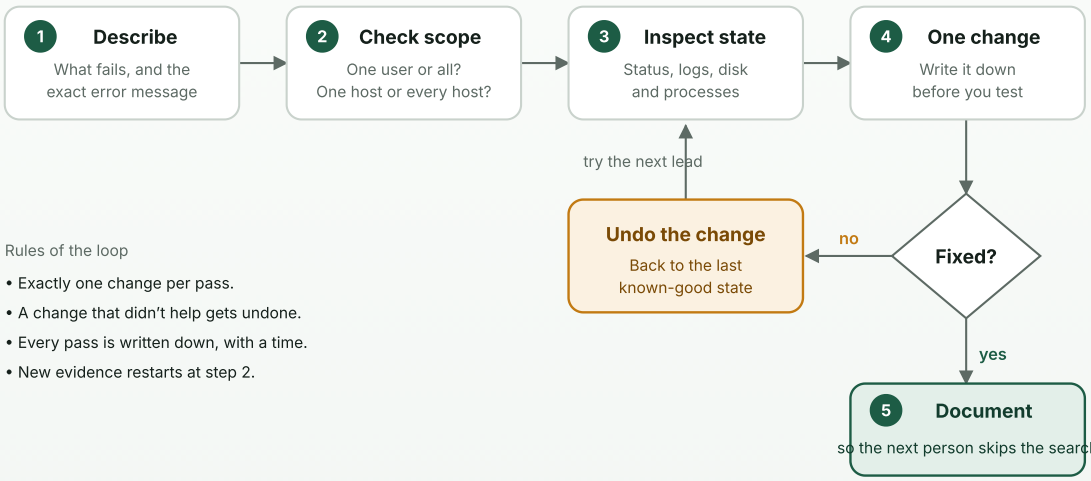


Figure 1 · The recovery loop: change one thing at a time, and loop until the evidence agrees.

Stacked changes hide which one mattered and leave the system in a state nobody recorded. If a change doesn't fix the problem, put it back first.

> **DANGER**
> If the machine holds data you can't lose, stop and back it up before any filesystem repair (`fsck`), bootloader change, partition edit, or bulk deletion.

What a good issue report includes

A report a stranger can act on without asking you a single follow-up question.

FIELD	EXAMPLE
Distribution and version	Ubuntu 24.04.1 LTS
Hardware	School lab PC, Intel Core i3, 8 GB RAM, on a UPS
Exact command	`sudo systemctl restart nginx`
Exact output	`Job for nginx.service failed because the control process exited with error code.`
Expected behavior	nginx restarts and serves the site
Actual behavior	Service stays in `failed` state
Recent changes	Added a new `server` block yesterday
Already tried	`nginx -t` reports a duplicate `listen` directive

05 Open-source practice

Leave the trail better marked than you found it.

Open source is more than a license. It means sharing enough context that someone else can inspect, reproduce, correct and extend your work.

READ

Start with the README, the `man` page, upstream docs, the issue tracker and the release notes, in that order.

VERIFY

Check versions, checksums (`sha256sum`) and signatures (`gpg --verify`). Confirm advice still matches the version you run.

SHARE

Publish reproducible steps, the license, attribution, and known limitations.

CONTRIBUTE

Fix the confusing sentence, add the missing distribution note, or translate the page. Small fixes count.

Contribution template

Copy this into a bug report, pull request or forum post.

issue.md

```
**Title:** short, searchable description
**Environment:** distribution, version, hardware
**Observed:** what happened
**Expected:** what should have happened
**Steps:** the shortest sequence that reproduces it
**Evidence:** relevant output, with secrets removed
**License:** license and attribution for anything you include
```

PART II · THE HIGH.TRAIL

Reference library

190 topics in 19 areas, each one a waypoint you can reach directly. Open it at any page: every chapter stands on its own.

01 Linux foundations
02 Shell and terminal
03 Filesystems and storage
04 Package management
05 Services and boot
06 Networking basics
07 Network services
08 Troubleshooting
09 Hardware and devices
10 Graphics and desktop
11 Security foundations
12 Cryptography and privacy
13 Containers and virtualization
14 Development tools
15 Automation and scripting
16 Data and text
17 Servers and operations
18 Open-source practice
19 Reference and recovery



How the library works

190 topics in 19 areas. Every row gives you a command you can run today, what it tells you, and the mistake to avoid.

LEAD
Why the area matters, in two sentences.

BEFORE YOU CLIMB
The three commands to run before anything else.

TEN TOPICS
Command, what it tells you, what to watch out for.

ACCLIMATISE
A safe exercise, and a trail marker to remember.

Replace anything in CAPITALS (`FILE`, `PKG`, `UNIT`) with your own value. Commands without `sudo` are read-only unless the row says otherwise.

01 Linux foundations — 14
Files · users · processes

02 Shell and terminal — 15
Quoting · pipes · globbing

03 Filesystems and storage — 16
Mounts · LVM · RAID · backups

04 Package management — 17
apt · dnf · pacman · zypper · flatpak · snap

05 Services and boot — 18
systemd · timers · GRUB

06 Networking basics — 19
Link · address · route · DNS

07 Network services — 20
SSH · HTTP · TLS · NTP · firewalls

08 Troubleshooting — 21
Timelines · bisecting · resources

09 Hardware and devices — 22
PCI · USB · drivers · power

10 Graphics and desktop — 23
Wayland · X11 · GPU · audio

11 Security foundations — 24
Updates · least privilege · audit

12 Cryptography and privacy — 25
Hashes · GPG · TLS · encryption

13 Containers and virtualization — 26
Namespaces · cgroups · Podman · KVM

14 Development tools — 27
git · make · compilers · debuggers

15 Automation and scripting — 28
Functions · traps · cron · timers

16 Data and text — 29
cut · awk · sed · jq · compression

17 Servers and operations — 30
Config · monitoring · incidents

18 Open-source practice — 31
Licenses · patches · governance

19 Reference and recovery — 32
man pages · rescue media · postmortems

01 Linux foundations

FILES · USERS · PROCESSES

Everything on Linux is a file, owned by a user, acted on by a process. Learn how those three fit together and almost every other chapter becomes easier to read.

BEFORE YOU CLIMB 1. `ls -lah /` 2. `id` 3. `ps aux --sort=-%cpu | head`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Filesystem layout	`ls -lah /` · `man hier`	Top-level directories and what each one holds	▲ Most distros now merge `/bin` into `/usr/bin` (the "usrmerge")
Paths and navigation	`pwd` · `find . -maxdepth 2 -type d`	Where you are, and the directories below	▲ Run `pwd` before any destructive command
Files and metadata	`stat FILE` · `file FILE`	Owner, mode, timestamps and the real file type	▲ Extensions don't decide a file's type on Linux; contents do
Links and inodes	`ln -s TARGET LINK` · `ls -li`	Creates a symlink; shows inode numbers	▲ A symlink breaks silently if its target moves; find broken ones with `find . -xtype l`
Users and groups	`id` · `getent passwd USER`	Numeric user and group IDs; the account record	▲ New group membership applies only after a fresh login
Permissions	`namei -l PATH` · `getfacl PATH`	Permissions on every directory in a path, plus ACLs	▲ Test as the affected user (`sudo -u USER …`), not as root
Umask	`umask -S`	Default permissions for newly created files	▲ Create a test file and check it with `ls -l`
Processes	`ps aux --sort=-%cpu | head`	The processes using the most CPU	▲ Note the PID before you investigate further
Signals	`kill -TERM PID` · `kill -KILL PID`	Asks a process to stop; forces it to stop	▲ Try TERM first; KILL skips the program's cleanup
Environment variables	`printenv | sort`	Every variable in your current shell	▲ Never paste the environment into a ticket: it often holds secrets

ACCLIMATISE · TRY IT

1. Run `namei -l ~/.bashrc` and read the permissions on each directory from `/` down to the file.
2. Create `touch test.txt`, run `stat test.txt`, then compare the mode with `umask -S`.
3. Open a second terminal, run `sleep 600`, find it with `pgrep -a sleep`, and stop it with `kill -TERM`.

TRAIL MARKER

When something is "denied", check the whole path, not just the last file.

02 Shell and terminal

QUOTING · PIPES · GLOBBING

The shell rewrites what you type before any command sees it: variables expand, wildcards become file lists, quotes disappear. Most scripting bugs begin in that rewrite, so learn to predict it.

BEFORE YOU CLIMB 1. `echo $SHELL` 2. `type COMMAND` 3. `set -x` (trace, then `set +x`)

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Shell startup files	`echo $SHELL` · `ls -a ~`	Your shell; Bash reads `~/.bashrc` for interactive shells and `~/.bash_profile` or `~/.profile` for login shells	▲ Test edits in a new terminal before closing the old one
Quoting and expansion	`echo "$HOME" '$HOME'`	Double quotes expand variables; single quotes don't	▲ Quote every variable (`"$file"`) or names with spaces break
Pipes and redirection	`cmd > out.txt 2>&1` · `cmd | tee log.txt`	Sends output and errors to a file; shows and saves at once	▲ `>` overwrites, `>>` appends; `set -o noclobber` guards against accidents
Command substitution	`today=$(date +%F)`	Stores a command's output in a variable	▲ Prefer `$( )` to backticks: it nests cleanly
Globbing	`echo /etc/*.conf`	The shell expands wildcards before the command runs	▲ Preview a glob with `echo` before using it with `rm`
History and aliases	`history | tail -20` · `alias ll='ls -lah'`	Recent commands; a shortcut for a long one	▲ Put aliases in `~/.bashrc` or they vanish when the shell exits
Job control	`Ctrl+Z` · `bg` · `fg` · `jobs`	Pause, background and resume jobs	▲ Jobs die with the terminal; use `tmux` for long tasks
Text filters	`sort FILE | uniq -c | sort -rn`	Counts repeated lines, most common first	▲ `uniq` only merges adjacent lines, so sort first
Regular expressions	`grep -E '^(ERROR|WARN)' app.log`	Lines starting with ERROR or WARN	▲ `-E` is extended regex; basic regex needs more backslashes
Shell safety	`set -euo pipefail` · `shellcheck script.sh`	Stops a script on errors and unset variables; lints it	▲ Guard deletes: `rm -rf "${DIR:?}/"` aborts if `DIR` is empty

ACCLIMATISE · TRY IT

1. Make a file with a space: `touch 'my file'`. Compare `f='my file'; ls $f` with `ls "$f"`.
2. Run `echo /etc/*.conf | wc -w` to see how many words one glob becomes.
3. Write a three-line script, run `shellcheck` on it, and fix each warning it reports.

TRAIL MARKER

If you can't predict what `echo` would print, don't run the real command yet.

03 Filesystems and storage

MOUNTS · LVM · RAID · BACKUPS

Disks fail, fill up and get renamed, and a sudden power cut is the classic way to corrupt one: use a UPS where you can. Inspect storage with read-only commands first, and treat anything that writes to a disk as a change that needs a backup and a way back.

BEFORE YOU CLIMB 1. `lsblk -f` 2. `df -hT` 3. `findmnt`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Mount points	`findmnt`	Tree of what is mounted where, with options	▲ A bad `/etc/fstab` line can stop boot; check with `sudo findmnt --verify`, then `sudo mount -a`
Block devices	`lsblk -f`	Disks, partitions, filesystems and UUIDs	▲ Names like `sdb` can change between boots; use UUIDs in `fstab`
Partition tables	`sudo fdisk -l` · `sudo parted -l`	GPT or MBR layout of each disk	▲ Back up first: editing the wrong disk wipes it
Filesystems	`df -T`	Type of each mounted filesystem (ext4, xfs, btrfs)	▲ Run `fsck` only on unmounted filesystems
Disk usage	`sudo du -xh --max-depth=1 / | sort -h`	Which top-level directories are largest	▲ `-x` stays on one filesystem; `ncdu` is the interactive version
Inode exhaustion	`df -i`	Inodes used on each filesystem	▲ "No space left" with free bytes usually means inodes ran out
LVM concepts	`sudo pvs` · `sudo vgs` · `sudo lvs`	Physical volumes, volume groups, logical volumes	▲ Grow volume and filesystem together: `lvextend -r`
RAID concepts	`cat /proc/mdstat`	Software RAID state and rebuild progress	▲ RAID is not a backup: it mirrors deletions too
Swap	`swapon --show` · `free -h`	Swap devices and how much is used	▲ Constant swapping means you need RAM, not more swap
Backups	`rsync -aAX --delete SRC/ DEST/`	Mirrors a directory, keeping permissions and ACLs	▲ Keep 3 copies on 2 media, 1 off-site, and test restores. Try `--dry-run` first: `--delete` removes files

ACCLIMATISE · TRY IT

1. Run `lsblk -f` and match each UUID with an entry in `/etc/fstab`.
2. Compare `df -h` and `df -i` on your root filesystem: which runs out first?
3. Back up a test folder with `rsync -aAX --dry-run` and read what it would copy before running it for real.

TRAIL MARKER

A backup you have never restored is a hope, not a backup.

04 Package management

APT · DNF · PACMAN · ZYPPER · FLATPAK · SNAP

Day-to-day package work on each family, and how to undo it. For the side-by-side install table, see section 03 of the Field manual.

BEFORE YOU CLIMB

1. `apt list --installed | wc -l` or `rpm -qa | wc -l`
2. `dnf history` or `less /var/log/apt/history.log`
3. `flatpak list --app`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
APT workflow	`sudo apt update && sudo apt upgrade`	Refreshes lists, then installs updates	▲ Read the list of packages to be removed before confirming
DPKG inspection	`dpkg -l PKG` · `dpkg -L PKG`	Installed version; every file the package installed	▲ `dpkg` doesn't resolve dependencies; install `.deb` files with `sudo apt install ./FILE.deb`
DNF workflow	`sudo dnf upgrade --refresh`	Refreshes metadata and updates	▲ `dnf history` lists every past transaction
RPM inspection	`rpm -qi PKG` · `rpm -V PKG`	Package details; files changed since install	▲ `-V` output is terse: a `5` means the checksum differs
Pacman workflow	`sudo pacman -Syu`	Full system upgrade	▲ Read the archlinux.org news before big upgrades
Zypper workflow	`sudo zypper refresh && sudo zypper update`	Refresh and update on Leap	▲ Tumbleweed uses `sudo zypper dup`
Flatpak	`flatpak list --app` · `flatpak info --show-permissions APP`	Installed apps and their sandbox permissions	▲ Not every Flathub app is published by its upstream; look for the verified badge
Snap	`snap list` · `snap connections SNAP`	Installed snaps and what they can access	▲ Snaps auto-update; `sudo snap refresh --hold` pauses that
Building from source	`./configure --prefix=$HOME/.local && make && make install`	Builds and installs without root	▲ Untracked installs in `/usr/local` are hard to remove cleanly
Package rollback	`sudo dnf history undo ID` · `sudo apt install PKG=VERSION`	Undoes a transaction or installs an older version	▲ Snapshot first (Timeshift, Snapper) where available

ACCLIMATISE · TRY IT

1. Pick any file in `/usr/bin` and find which package owns it (`dpkg -S` or `rpm -qf`).
2. List the last five package transactions on your system and what each changed.
3. Check the permissions of one Flatpak app and decide whether each one is justified.

TRAIL MARKER

Every install is also a promise to keep that software updated. Know where the updates come from.

05 SYSTEMD · TIMERS · GRUB
Services and boot

systemd starts, supervises and logs nearly everything on a modern distribution. Knowing how to ask it what happened, and how to boot with less when something breaks, turns a dead machine into a fixable one.

BEFORE YOU CLIMB 1. `systemctl --failed` 2. `systemd-analyze blame | head` 3. `journalctl -b -p err`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
systemd units	`systemctl list-units --type=service`	Loaded services and their state	▲ Don't edit `/usr/lib/systemd/system`; use `sudo systemctl edit UNIT`
Service status	`systemctl status UNIT`	State, main PID and the last log lines	▲ "active (exited)" is normal for one-shot units
Journal queries	`journalctl -u UNIT --since "1 hour ago"`	One service's logs in a time window	▲ Add `-b -1` to read the previous boot (needs a persistent journal)
Targets	`systemctl get-default`	Whether you boot to graphical or multi-user	▲ `rescue.target` gives a root shell with minimal services
Timers	`systemctl list-timers`	Scheduled jobs with last and next run	▲ Timer output goes to the journal, unlike classic cron mail
Socket activation	`systemctl list-sockets`	Sockets that start a service on first connection	▲ The service can look stopped until a client connects
Boot parameters	`cat /proc/cmdline`	Kernel options used for this boot	▲ Edit `/etc/default/grub`, then regenerate the GRUB config
Initramfs	`lsinitrd` (Fedora) · `lsinitramfs /boot/initrd.img-$(uname -r)` (Debian)	What's in the early-boot image	▲ Rebuild after storage changes: `sudo dracut -f` or `sudo update-initramfs -u`
GRUB recovery	`sudo update-grub` (Debian) · `sudo grub2-mkconfig -o /boot/grub2/grub.cfg` (Fedora)	Regenerates the boot menu	▲ Have a live USB ready before touching the bootloader
Shutdown and reboot	`systemctl reboot` · `sudo shutdown -h +10 "Maintenance"`	Reboots now; halts in 10 minutes with a warning	▲ `last -x reboot` shows past reboots and shutdowns

ACCLIMATISE · TRY IT

1. Run `systemd-analyze blame | head` and look up what the slowest unit does.
2. Pick a service and read its last hour of logs with `journalctl -u`.
3. Read your kernel command line in `/proc/cmdline` and look up each option you don't recognise.

TRAIL MARKER

Change a unit with `systemctl edit`; your override survives package updates.

06 Networking basics

LINK · ADDRESS · ROUTE · DNS

Work from the bottom up, like checking each anchor on a rope: link, address, route, then DNS. Most “the internet is down” problems are exactly one of those four, and each has a one-line test.

BEFORE YOU CLIMB 1. `ip -br addr` 2. `ip route` 3. `resolvectl query example.com`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Interfaces	`ip -br link` · `ip -br addr`	Each interface, its state and addresses	▲ `ifconfig` is deprecated on most distros
IPv4 addressing	`ip -4 addr`	IPv4 addresses with prefix length	▲ An address in 169.254.x.x means DHCP failed
IPv6 basics	`ip -6 addr`	Global and link-local (`fe80::`) addresses	▲ Blocking all ICMPv6 breaks IPv6
Routes	`ip route` · `ip route get 1.1.1.1`	Routing table; the route a packet will take	▲ No `default` route means no internet
ARP and neighbor state	`ip neigh`	MAC addresses of hosts on your network	▲ `FAILED` entries point to a host that’s unreachable
DNS resolution	`resolvectl query example.com` · `dig example.com`	What a name resolves to and who answered	▲ Test by IP first to separate DNS from connectivity
Hosts file	`getent hosts NAME`	Resolution including `/etc/hosts`	▲ `/etc/hosts` entries override DNS and are easy to forget
Network namespaces	`ip netns list` · `lsns -t net`	Isolated network stacks, such as containers	▲ Each namespace has its own interfaces and routes
NetworkManager	`nmcli device status` · `nmcli connection show`	Devices and saved connections	▲ After editing files by hand run `sudo nmcli connection reload`
Connectivity tests	`ping -c 4 HOST` · `tracepath HOST`	Whether a host answers, and the path to it	▲ Many hosts block ping; try `curl -I` for web servers

ACCLIMATISE · TRY IT

1. Walk the four layers in order: `ip -br link`, `ip -br addr`, `ip route`, `resolvectl query example.com`.
2. Run `ip route get 1.1.1.1` and name the interface and gateway it would use.
3. Compare `getent hosts localhost` with `dig localhost` and explain the difference.

TRAIL MARKER

Ping an IP, then a name. If only the name fails, it’s DNS.

07 Network services

SSH · HTTP · TLS · NTP · FIREWALLS

Remote access, web, time and firewall services, each with a quick check. On a remote machine, the rule is simple: never change the thing that lets you back in without a second way in.

BEFORE YOU CLIMB 1. `ss -tulpn` 2. `systemctl status sshd` 3. `timedatectl`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
SSH client	`ssh -v USER@HOST`	Connects and prints each handshake step	▲ `-v` shows which key or auth method failed
SSH server	`sudo sshd -t` · `systemctl status sshd`	Config syntax check; daemon state	▲ Keep a second session open while editing `sshd_config` (Debian calls the unit `ssh`)
Key authentication	`ssh-keygen -t ed25519` · `ssh-copy-id USER@HOST`	Makes a key pair; installs the public key	▲ `~/.ssh` must be mode 700 and `authorized_keys` 600
Port forwarding	`ssh -L 8080:localhost:80 USER@HOST`	Reaches a remote port through a local one	▲ Bind to localhost unless you mean to share the port
HTTP inspection	`curl -sSIL https://example.com`	Status codes, headers and redirects	▲ Add `-v` to see TLS and request details
TLS certificates	`openssl s_client -connect HOST:443 -servername HOST`	Certificate chain and expiry	▲ A missing intermediate breaks some clients but not browsers
FTP alternatives	`sftp USER@HOST` · `rsync -avP SRC USER@HOST:DEST`	Encrypted file transfer	▲ Plain FTP sends passwords in clear text
NTP and time	`timedatectl` · `chronyc tracking`	Clock, time zone and sync state	▲ A wrong clock breaks TLS, logins and log correlation; check it after any long power cut
mDNS	`avahi-browse -art`	Services advertised on the local network	▲ `.local` names resolve only on the same network segment
Firewall zones	`sudo firewall-cmd --list-all` · `sudo ufw status verbose`	Allowed services and ports	▲ Allow SSH before enabling a firewall on a remote host

ACCLIMATISE · TRY IT

1. Run `ss -tulpn` and name the service behind every listening port.
2. Check a site's certificate expiry: pipe `openssl s_client` into `openssl x509 -noout -dates`.
3. Make an ed25519 key and read its fingerprint with `ssh-keygen -lf`.

TRAIL MARKER

Every listening port is a door. Know who opened it and why.

08 Troubleshooting

TIMELINES · BISECTING · RESOURCES

Recovery moves for common failures. Use them inside the five-step loop in section 04 of the Field manual: observe, narrow, change one thing, verify, record.

BEFORE YOU CLIMB 1. `journalctl -b -p err` 2. `uptime` 3. `df -h`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Build a timeline	`journalctl --since "09:00" --until "10:00"`	Everything logged in a time window	▲ The journal shows local time; note time zones across hosts (`--utc`)
Read an error	`journalctl -xe`	Recent entries with explanations	▲ The first error usually matters; later ones are often fallout
Reproduce a bug	`script -q session.log`	Records a whole terminal session	▲ A bug you can't reproduce can't be confirmed fixed
Bisect a change	`git bisect start` · `git bisect bad` · `git bisect good TAG`	Binary search for the commit that broke it	▲ Works for config too if `/etc` is in git (etckeeper)
Safe rollback	`sudo cp -a FILE FILE.bak.$(date +%F)`	Dated copy before you edit	▲ Keep the backup until the change has proven itself
Single-user recovery	Add `systemd.unit=rescue.target` at the GRUB prompt (`e` to edit)	Boots to a root shell	▲ Some distros ask for the root password; know it in advance
Disk-full recovery	`sudo du -xh / | sort -h | tail -20` · `sudo journalctl --vacuum-size=200M`	Largest items; trims the journal	▲ Deleted files still open keep using space: `sudo lsof +L1`
Memory pressure	`free -h` · `journalctl -k | grep -i oom`	Memory state; processes killed by the OOM killer	▲ Read the "available" column, not "free"
High CPU	`top -o %CPU` · `pidstat 1 5`	Which processes use the CPU	▲ High load with idle CPU usually means I/O wait (`wa` in top)
Frozen desktop	`Ctrl+Alt+F3`, log in, then `loginctl`	A text console to inspect from	▲ Restarting the display manager ends the graphical session

ACCLIMATISE · TRY IT

1. Build a ten-minute timeline around your last boot with `journalctl -b --since`.
2. Find the five largest directories in your home with `du -xh ~ | sort -h | tail -5`.
3. Before your next config edit, make a dated `.bak` copy and note it in a change log.

TRAIL MARKER

The first error in the log is the cause more often than the loudest one.

09 Hardware and devices

PCI · USB · DRIVERS · POWER

Identify each device, the driver bound to it, and its health. A device without a driver is the most common reason hardware "doesn't work", and it is a two-command check.

BEFORE YOU CLIMB 1. `lspci -k` 2. `lsusb` 3. `sudo dmesg --level=err,warn`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
PCI devices	`lspci -k`	Each PCI device and the driver bound to it	▲ No "Kernel driver in use" line means no driver loaded
USB devices	`lsusb` · `lsusb -t`	USB devices and the bus tree	▲ Run `journalctl -kf` while plugging a device in
udev rules	`udevadm info -a /dev/DEVICE`	Attributes a rule can match on	▲ Reload with `sudo udevadm control --reload`, then `sudo udevadm trigger`
Kernel modules	`lsmod` · `modinfo MODULE`	Loaded modules; their options and firmware	▲ Blacklist in `/etc/modprobe.d/` instead of deleting files
Firmware	`fwupdmgr get-updates`	Firmware updates available from LVFS	▲ Keep a laptop on AC power while flashing firmware
CPU information	`lscpu`	Model, cores, threads and virtualization support	▲ Scripts should use `nproc` for the usable core count
Memory information	`sudo dmidecode -t memory`	RAM slots, sizes and speeds	▲ Test suspect RAM with memtest86+
Sensors	`sensors`	Temperatures, fan speeds and voltages	▲ Run `sudo sensors-detect` once after installing lm-sensors
Power management	`powerprofilesctl` · `sudo powertop`	Active power profile; what drains the battery	▲ `powertop --auto-tune` can disable USB mice and keyboards
Laptop diagnostics	`upower -i $(upower -e | grep BAT)`	Battery health, capacity and charge cycles	▲ Compare `energy-full` with `energy-full-design`

ACCLIMATISE · TRY IT

1. Run `lspci -k` and find the driver for your network card and GPU.
2. Start `journalctl -kf`, plug in a USB stick, and read what the kernel reports.
3. Work out your battery health as `energy-full` ÷ `energy-full-design` × 100.

TRAIL MARKER

Missing driver first, broken hardware second.

10 Graphics and desktop

WAYLAND · X11 · GPU · AUDIO

Display servers, GPU drivers, audio and input on the Linux desktop. First find out whether you are on Wayland or X11: many fixes, and most forum answers, only apply to one of them.

BEFORE YOU CLIMB 1. `echo $XDG_SESSION_TYPE` 2. `glxinfo -B` 3. `wpctl status`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Display server	`echo $XDG_SESSION_TYPE`	`wayland` or `x11`	▲ Many tools behave differently under each
Wayland basics	`loginctl show-session $XDG_SESSION_ID -p Type`	Confirms the session type from systemd	▲ Screen sharing and remote control need xdg-desktop-portal
X11 basics	`xrandr` · `xdpyinfo | head`	Connected outputs and display details	▲ On Wayland, X11 apps run through XWayland
GPU drivers	`lspci -k | grep -A3 -E 'VGA|3D'` · `glxinfo -B`	GPU, driver and active renderer	▲ Renderer "llvmpipe" means software rendering: the GPU driver isn't working
Multi-monitor	Display settings, or `xrandr --output HDMI-1 --right-of eDP-1` (X11)	Arranges screens	▲ Mixed scaling (HiDPI plus a normal screen) works best on Wayland
Input devices	`sudo libinput list-devices`	Keyboards, touchpads and their settings	▲ Touchpad gestures depend on the desktop, not the driver
Desktop services	`systemctl --user list-units --failed`	Failed per-user services	▲ Read their logs with `journalctl --user -u UNIT`
Audio stack	`wpctl status` · `pactl info`	PipeWire devices and the default output	▲ Most current distros run PipeWire with a PulseAudio compatibility layer
Bluetooth audio	`bluetoothctl`, then `devices` and `info MAC`	Pairing and connection state	▲ Headsets drop to low-quality audio while the mic is in use
Remote desktop	GNOME Remote Desktop · KRDC · `xrdp`	Remote access to a graphical session	▲ Tunnel over SSH or a VPN; never expose RDP or VNC to the internet

ACCLIMATISE · TRY IT

1. Check your session type, then confirm it a second way with `loginctl`.
2. Run `glxinfo -B` and check the renderer is your real GPU, not llvmpipe.
3. Use `wpctl status` to find your default output, then switch it with `wpctl set-default ID`.

TRAIL MARKER

Wayland or X11? Answer that before you search for any fix.

11 Security foundations

UPDATES · LEAST PRIVILEGE · AUDIT

Most real-world compromises are stopped by unglamorous basics: prompt updates, least privilege, careful handling of secrets, and a record of who did what.

BEFORE YOU CLIMB 1. `sudo -l` 2. `last -n 10` 3. `ss -tulpn`

TOPIC	COMMAND OR PRACTICE	WHAT IT TELLS YOU	WATCH OUT
Threat modeling	Ask: what am I protecting, from whom, and how bad is losing it?	Which controls are worth their cost	▲ Write it down; revisit when the machine's role changes
Updates	`apt list --upgradable` · `dnf updateinfo list --security`	Pending updates and security fixes	▲ Kernel and libc fixes need a reboot (or service restarts) to apply; `needrestart` lists them
Least privilege	`sudo -l`	What your account may run as root	▲ Give each service its own unprivileged user
Sudo policy	`sudo visudo -f /etc/sudoers.d/NAME`	Edits sudo rules with a syntax check	▲ A broken sudoers file can lock out all admin access
File capabilities	`getcap -r / 2>/dev/null`	Binaries holding special privileges	▲ Prefer a single capability over setuid root
Secure deletion limits	`shred -u FILE`	Overwrites a file, then deletes it	▲ Doesn't work on SSDs or copy-on-write filesystems; use disk encryption
Secrets hygiene	`git log -p | grep -iE 'password|token'` · `gitleaks detect`	Secrets committed to git history	▲ A leaked secret must be rotated, not just deleted
Audit trails	`sudo ausearch -m USER_LOGIN -ts today`	Today's audited logins	▲ auditd rules must exist before the event happens
Login history	`last -n 20` · `sudo lastb -n 20`	Recent successful and failed logins	▲ Many failed SSH logins: turn off password authentication
Security checklist	Updates on · SSH keys only · firewall on · disk encrypted · backups tested	A baseline for any machine	▲ Review it every quarter

ACCLIMATISE · TRY IT

1. Write a three-line threat model for your laptop: what, from whom, how bad.
2. Run `sudo -l` and remove any rule you no longer need.
3. Score your machine against the five-point checklist in the last row.

TRAIL MARKER

Rotate, don't just delete. A secret that was public once is public forever.

12 Cryptography and privacy

HASHES · GPG · TLS · ENCRYPTION

Verify what you download, protect what you store, and know what you leak. Each tool here answers a narrow question; the skill is knowing which question it doesn't answer.

BEFORE YOU CLIMB

1. `sha256sum FILE`
2. `gpg --verify FILE.sig FILE`
3. `lsblk -f` (look for crypto_LUKS)

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Hashing	`sha256sum FILE` · `sha256sum -c SHA256SUMS`	Computes or checks file checksums	▲ A hash from the same server proves integrity, not who made the file
GPG keys	`gpg --verify FILE.sig FILE`	Whether a signature is valid, and whose key made it	▲ Confirm the key's fingerprint through a second channel
SSH fingerprints	`ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub`	A server's host key fingerprint	▲ A "host key changed" warning can mean interception: check before accepting
TLS trust	`trust list | head` (Fedora) · `sudo update-ca-certificates` (Debian)	The system's trusted certificate authorities	▲ Add internal CAs to the store; never turn verification off
Disk encryption	`lsblk -f` · `sudo cryptsetup luksDump DEVICE`	Whether a disk is LUKS-encrypted, and its settings	▲ Keep the recovery passphrase offline
Password managers	KeePassXC · Bitwarden · `pass`	A unique password for every account	▲ Protect the vault with a long passphrase and 2FA
Metadata	`exiftool FILE` · `mat2 FILE`	Shows and strips hidden file metadata	▲ Phone photos often carry GPS coordinates
Tor concepts	Tor Browser	Hides your IP address from the sites you visit	▲ Logging into your own accounts still identifies you
DNS privacy	`resolvectl status`	Whether DNS-over-TLS is in use	▲ Your ISP still sees the IP addresses you connect to
Privacy tradeoffs	Decide per your threat model	Convenience versus exposure	▲ Sync and telemetry trade privacy for features: choose deliberately

ACCLIMATISE · TRY IT

1. Download a distro's `SHA256SUMS` and its signature; verify the signature first, then the ISO.
2. Run `exiftool` on a phone photo and look for GPS fields; strip them with `mat2`.
3. Check whether your disk is encrypted with `lsblk -f`.

TRAIL MARKER

Checksum = not corrupted. Signature = made by who you think.

13 Containers and virtualization

NAMESPACES · CGROUPS · PODMAN · KVM

Isolation built from kernel namespaces and cgroups, from lightweight containers up to full virtual machines. Examples use Podman; most commands work with `docker` unchanged.

BEFORE YOU CLIMB 1. `systemd-detect-virt` 2. `podman ps -a` 3. `systemd-cgtop`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Containers vs VMs	`systemd-detect-virt`	Whether you're inside a VM or a container	▲ Containers share the host kernel; VMs run their own
Namespaces	`lsns`	Namespaces in use and the processes in each	▲ Container isolation is only as strong as the kernel
cgroups	`systemd-cgls` · `systemd-cgtop`	Resource groups and their live usage	▲ Current distros default to cgroup v2
Rootless containers	`podman run --rm -it alpine sh`	Runs a container with no root daemon	▲ Your user needs ranges in `/etc/subuid` and `/etc/subgid`
Images	`podman images` · `podman image inspect IMAGE`	Local images, their layers and metadata	▲ Pin a version tag or digest; `latest` changes under you
Volumes	`podman volume ls` · `-v ./data:/data:Z`	Persistent storage	▲ `:Z` sets a private SELinux label on Fedora and RHEL; use `:z` for shared volumes
Container networking	`podman port CONTAINER` · `podman network ls`	Published ports and networks	▲ Docker's published ports can bypass host firewall rules
Compose concepts	`podman compose up -d` · `docker compose up -d`	Starts a multi-container app from `compose.yaml`	▲ Keep secrets in env files outside version control
Virtual machines	`virsh list --all` · virt-manager · GNOME Boxes	KVM guests and their state	▲ Check `lscpu` for VT-x or AMD-V support first
Resource limits	`podman run --memory 512m --cpus 1 IMAGE`	Caps a container's memory and CPU	▲ Without limits one container can starve the host

ACCLIMATISE · TRY IT

1. Run `podman run --rm -it alpine sh`, then `cat /etc/os-release` inside and outside it.
2. Inside the container run `uname -r` and compare with the host: same kernel.
3. Start a container with `--memory 64m` and watch it in `systemd-cgtop`.

TRAIL MARKER

A container is a process with a different view, not a separate machine.

14 Development tools

GIT · MAKE · COMPILERS · DEBUGGERS

Version control, builds and debugging. These are the tools you will use to read, fix and contribute to open-source code, so learn the safe exits as well as the commands.

BEFORE YOU CLIMB 1. `git status` 2. `git log --oneline -10` 3. `strace -f -e trace=openat CMD`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Git basics	`git status` · `git log --oneline -10`	Working tree state; recent history	▲ Commit small, with messages that say why
Branches	`git switch -c NAME` · `git branch -a`	Creates a branch; lists all branches	▲ Delete merged branches to keep the list readable
Rebases	`git rebase main`	Replays your commits on top of main	▲ Never rebase commits others have already pulled
Merge conflicts	`git status` · `git mergetool`	Which files conflict	▲ Back out safely with `git merge --abort` or `git rebase --abort`
Git hooks	`ls .git/hooks` · pre-commit framework	Checks that run before commit or push	▲ Hooks aren't cloned; share them through a tool or script
Make	`make -n`	What would run, without running it	▲ Recipe lines must start with a tab, not spaces
CMake	`cmake -S . -B build && cmake --build build`	An out-of-source build	▲ Delete `build/` to clear stale cached settings
Compilers	`gcc -Wall -Wextra -g FILE.c -o app`	Compiles with warnings and debug info	▲ Treat warnings as bugs
Debuggers	`gdb ./app` · `strace -f ./app`	Step through code; trace system calls	▲ `strace` quickly reveals missing files and permission errors
Package environments	`python3 -m venv .venv && . .venv/bin/activate`	Per-project dependencies	▲ Never `sudo pip install` into the system Python; distros now block it (PEP 668)

ACCLIMATISE · TRY IT

1. Clone any small project and read `git log --oneline -20` for how it explains changes.
2. Run `strace -f -e trace=openat ls` and see every file `ls` opens.
3. Start a rebase on a scratch branch, then practise backing out with `--abort`.

TRAIL MARKER

Know the `--abort` before you start the merge.

15 Automation and scripting

FUNCTIONS · TRAPS · CRON · TIMERS

Turn repeated manual steps into scripts that fail loudly and safely. A good script checks its inputs, cleans up after itself, and gives the same result when run twice.

BEFORE YOU CLIMB 1. `set -euo pipefail` 2. `shellcheck script.sh` 3. `systemctl list-timers --all`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
Bash functions	`backup() { cp -a "$1" "$1.bak"; }`	Reusable shell logic	▲ Declare variables with `local`, or they leak into the caller
Exit codes	`cmd; echo "exit: $?"`	0 means success; non-zero names a failure	▲ Return meaningful codes from your own scripts
Traps	`trap 'rm -f "$tmpfile"' EXIT`	Runs cleanup on exit, even after an error	▲ Without a trap, temp files survive a crashed script
Argument parsing	`while getopts "f:v" opt; do …; done`	Reads flags and their values in a loop	▲ Quote `"$@"` when passing arguments through
Config files	`source config.sh` · read a `.env` with `set -a; source .env; set +a`	Loads settings into a script	▲ Validate required variables: `: "${API_URL:?not set}"`
Cron	`crontab -e` · `crontab -l`	Edits or lists your scheduled jobs	▲ Cron runs with a minimal `PATH`; use full paths in commands
systemd timers	`systemctl list-timers --all`	Scheduled units, including inactive ones	▲ Prefer timers over cron: they log to the journal and can depend on other units
Idempotence	Check before you change: `[ -f FILE ] || touch FILE`	Running a script twice gives the same result	▲ A non-idempotent script is unsafe to retry after a partial failure
Logging scripts	`exec > >(tee -a script.log) 2>&1`	Sends all script output to a log file too	▲ Add timestamps by piping through `ts` (moreutils)
Test scripts	`bats test_script.bats` · `shellcheck script.sh`	Runs shell tests; lints common mistakes	▲ Test the failure paths, not only the happy path

ACCLIMATISE · TRY IT

1. Wrap a task you do weekly in a script with `set -euo pipefail` and a cleanup `trap`.
2. Run the script twice in a row. Does the second run change anything? It shouldn't.
3. Schedule it with a user timer and read its output with `journalctl --user -u`.

TRAIL MARKER

Safe to run twice is the bar for any script you schedule.

16 Data and text

CUT · AWK · SED · JQ · COMPRESSION

The classic Unix text tools are still the fastest way to slice logs and data on the command line. Each does one thing; pipes join them into answers.

BEFORE YOU CLIMB 1. `less FILE` 2. `awk '{print $1}' FILE | sort | uniq -c` 3. `jq . FILE.json`

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
cat and less	`cat FILE` · `less FILE`	Dumps a file; pages through it	▲ Use `less` for anything longer than a screen; `less +F` follows like `tail -f`
cut and paste	`cut -d, -f1,3 FILE.csv`	Pulls specific columns from delimited text	▲ `cut` splits on the literal delimiter; quoted commas in CSV break it
sort and uniq	`sort FILE | uniq -c | sort -rn`	Counts of each unique line, most frequent first	▲ `uniq` only merges adjacent lines, so sort first
awk patterns	`awk -F, '{print $2}' FILE.csv`	Prints one field from each line	▲ `$0` is the whole line, `$NF` is the last field
sed edits	`sed -i.bak 's/old/new/g' FILE`	Edits a file in place, keeping a `.bak` copy	▲ Preview with `sed 's/old/new/g' FILE` (no `-i`) before writing
jq JSON	`curl -s URL | jq '.items[].name'`	Extracts fields from JSON	▲ `jq -r` strips the quotes for plain-text output
CSV handling	`csvcut -c name,email FILE.csv` · `qsv` · `xan`	Tools built for real CSV, quoting included	▲ Plain `awk -F,` breaks on quoted fields with commas inside
Logs to reports	`awk '{print $1}' access.log | sort | uniq -c | sort -rn | head`	Top IPs or values from a log file	▲ Match the log's actual field separator and order
Encoding	`file -i FILE` · `iconv -f latin1 -t utf-8 FILE`	Detects encoding; converts between encodings	▲ Mixed encodings in one file cause silent corruption, not errors
Compression	`tar -czf out.tar.gz DIR` · `zstd -19 FILE`	Archives and compresses; zstd is faster for backups	▲ `tar -x` without `-C DEST` extracts into the current directory; list first with `tar -tf`

ACCLIMATISE · TRY IT

1. Find the ten most common words in a text file with one pipeline.
2. Fetch any public JSON API with `curl -s` and pull one field out with `jq -r`.
3. List an archive with `tar -tf` before extracting it into a fresh directory with `-C`.

TRAIL MARKER

Preview, then write: run `sed` without `-i` and `tar` with `-t` first.

17 Servers and operations

CONFIG · MONITORING · INCIDENTS

Running Linux as infrastructure: naming, configuration, monitoring and incident response. The aim is a server you could rebuild from notes and code, not one you are afraid to reboot. Expeditions keep a logbook for the same reason.

BEFORE YOU CLIMB 1. `hostnamectl` 2. `vmstat 1 5` 3. `systemctl --failed`

TOPIC	COMMAND OR PRACTICE	WHAT IT TELLS YOU	WATCH OUT
Host naming	`hostnamectl set-hostname NAME`	Sets a clear, consistent hostname	▲ Use a naming scheme (role-env-number) from day one
Configuration management	Ansible · Puppet · NixOS	Config as code, applied the same way every time	▲ Hand-edited "snowflake" servers are hard to reproduce or replace
Service health	`curl -sf http://localhost/healthz`	Whether a health endpoint responds	▲ A process running isn't the same as a service working
Capacity planning	`vmstat 1 5` · historical metrics	Trend, not just a snapshot	▲ Plan from peak load, not average load
Monitoring	Prometheus node_exporter · `netdata`	Metrics collected and exposed over time	▲ Monitor the user-facing symptom, not just CPU and memory
Alert design	Alert on symptoms (latency, errors) more than causes	Fewer, more meaningful alerts	▲ A noisy alert gets muted, then misses the real incident
Log rotation	`cat /etc/logrotate.d/*` · `sudo logrotate -d FILE`	Rotation rules; a dry run of what would happen	▲ A missing `postrotate` signal can leave a service writing to a deleted file
Incident notes	Timestamp every entry as you go	A record built during the incident, not after	▲ Notes reconstructed from memory lose the exact order
Change windows	Schedule risky changes for low-traffic periods, announced in advance	When changes are allowed	▲ An undocumented emergency change is still an incident waiting to happen
Runbooks	A versioned doc per recurring task	Step-by-step for a known problem, kept current	▲ A runbook nobody updates after an incident goes stale fast

ACCLIMATISE · TRY IT

1. Write a one-page runbook for restarting one service you run, including how to check it worked.
2. Run `vmstat 1 5` at a quiet time and a busy time; compare the `r` and `wa` columns.
3. Dry-run your log rotation with `sudo logrotate -d /etc/logrotate.conf`.

TRAIL MARKER

If it only lives in your head, it isn't operated; it's remembered.

LICENSES · PATCHES · GOVERNANCE

18 Open-source practice

Participating in a project well means reading before asking, and contributing in a way maintainers can act on quickly. Small, clear, well-described changes are the ones that get merged.

BEFORE YOU CLIMB

1. Read `README` and `CONTRIBUTING.md`
2. Search the issue tracker
3. `git log --oneline -20`

TOPIC	WHERE TO LOOK	WHAT IT TELLS YOU	WATCH OUT
Read a README	The project's repository root	Purpose, install steps, how to get help	▲ A stale README still outranks a guess; check the last-updated date
Choose a license	choosealicense.com	Permissive (MIT, Apache-2.0) vs copyleft (GPL) tradeoffs	▲ Check your dependencies' licenses are compatible with yours
Contribute a patch	`git switch -c fix-typo` · open a pull request	A focused, reviewable change	▲ One change per pull request; unrelated fixes slow review
Write an issue	The project's issue tracker	Steps to reproduce, expected vs actual, environment	▲ Search existing issues first; duplicates cost maintainers time
Review a pull request	Read the diff, then run it locally	Whether the change does what it claims, without regressions	▲ Comment on the code, not the contributor
Release notes	`git log TAG1..TAG2 --oneline`	What changed between two versions	▲ Call out breaking changes clearly, at the top
Semantic versioning	`MAJOR.MINOR.PATCH`	Breaking vs additive vs fix-only changes	▲ A breaking change in a MINOR release breaks user trust
SBOM concepts	`syft dir:. -o spdx-json`	A software bill of materials: every dependency, listed	▲ Needed to know your exposure when a dependency has a CVE
Project governance	The project's `GOVERNANCE.md` or wiki	Who decides what, and how	▲ No governance doc often means one maintainer decides alone
Community conduct	The project's `CODE_OF_CONDUCT.md`	Expected behavior and how to report violations	▲ Follow it even in disagreements; it protects the whole project

ACCLIMATISE · TRY IT

1. Find a project you use, read its `CONTRIBUTING.md`, and note one thing it asks that surprised you.
2. Look for an issue labelled "good first issue" and read the whole thread before commenting.
3. Fix one confusing sentence in any project's docs and open a pull request.

TRAIL MARKER

The smallest useful contribution is still a contribution.

19 Reference and recovery

MAN PAGES · RESCUE MEDIA · POSTMORTEMS

Where to find authoritative documentation, and the last-resort moves when a system won't boot. Prepare these before you need them: a rescue USB made during an outage is made in a hurry.

BEFORE YOU CLIMB 1. `man -k KEYWORD` 2. `tldr COMMAND` 3. Make a live USB today

TOPIC	COMMAND	WHAT IT TELLS YOU	WATCH OUT
man pages	`man COMMAND` · `man -k keyword`	The command's own manual; a keyword search across all manuals	▲ Section numbers matter: `man 5 passwd` is the file format, `man 1 passwd` the command
info pages	`info COMMAND`	GNU's more detailed manual format, for tools like `bash` and `coreutils`	▲ Navigate with arrow keys and Enter; `q` quits
apropos	`apropos keyword`	Same search as `man -k`, without opening a page	▲ Needs the `mandb` index to be built (`sudo mandb`)
Online documentation	The project's own docs site	The current, maintained reference	▲ Prefer it over a forum post that may be years out of date
Minimal reproducer	Strip a failing case down to the smallest input that still fails	The actual cause, not incidental complexity	▲ A minimal reproducer is often most of the fix
Recovery media	A live USB (Ventoy, or `dd` an ISO to a drive)	A working system to boot from when the installed one won't	▲ Verify the ISO's checksum first; triple-check `dd` 's `of=` target
Rescue shell	Add `systemd.unit=rescue.target` at the GRUB prompt	A root shell with minimal services running	▲ Know the root password before you need it
Restore a backup	`rsync -aAX --delete BACKUP/ /` (from rescue media) · `tar -xzf backup.tar.gz -C /`	Brings files back from a backup	▲ Restore to a test location first when you can
Postmortem	Timeline · impact · root cause · what fixed it · what prevents it recurring	A record the next person can learn from	▲ Blameless: focus on the system and process, not who made the change
Personal checklist	What changed? · What still works? · What does the evidence say? · Smallest working path · One change at a time · Document the result	The book's method, one line each	▲ Pin this list somewhere you'll see it mid-incident

ACCLIMATISE · TRY IT

1. Look up `man 5 fstab` and `man 8 mount` and note what each section number means.
2. Write a verified live USB and boot from it once, so you know your firmware's boot key.
3. Write a five-line postmortem for the last thing that broke on your machine.

TRAIL MARKER

Pack the rescue kit at base camp, not on the ridge.

Rescue card

Mountain teams carry a rescue card so nobody has to think clearly while cold and scared. This is yours for systems. Print it, and pin it where you'll see it when something breaks.

1 Stop

- Breathe. Don't type yet.
- Start notes with the time: `date`
- Is data at risk? Back up first.

2 Ask

- What changed?
- What still works?
- What does the evidence say?

3 Snapshot

- `uptime` · `free -h` · `df -h`
- `ip -br addr` · `ip route`
- `journalctl -p err -b`

4 Change one thing

- Dated backup: `cp -a F F.bak.$(date +%F)`
- Write down the change, then test.
- Didn't help? Undo it.

5 Verify

- Test the symptom the user saw.
- Check `systemctl --failed`.
- Watch logs: `journalctl -f`

6 Record

- Timeline, impact, cause, fix.
- What prevents a repeat?
- Blameless, and shared.

Never, mid-incident

- ✕ `rm -rf` on a path you haven't checked with `pwd` and `ls`
- ✕ `fsck` on a mounted filesystem
- ✕ `chmod -R 777` to "fix" permissions
- ✕ `curl … | sudo bash` from a forum answer
- ✕ Enable a firewall on a remote host before allowing SSH
- ✕ Paste unredacted logs or `printenv` into public tickets

Further reading

Authoritative, maintained sources. Prefer these to search results.

SOURCE	BEST FOR
`man` and `info` pages	The exact behaviour of the version installed on *your* machine
Arch Wiki	Deep, distribution-neutral explanations of almost every subsystem
Debian Administrator's Handbook	APT, Debian conventions and system administration end to end
Fedora and RHEL documentation	DNF, SELinux, firewalld and systemd on Red Hat-family systems
openSUSE documentation	Zypper, YaST, Btrfs snapshots and Snapper
systemd.io and `man systemd.unit`	Units, timers, the journal and boot
The Linux Command Line (William Shotts)	A free, complete introduction to the shell
Bash Reference Manual (GNU)	Quoting, expansion and scripting, precisely
Pro Git (Chacon & Straub)	Git from first commit to internals; free online
choosealicense.com and SPDX	Choosing and naming open-source licenses
Open Source Guides (opensource.guide)	Contributing to and maintaining a project



Start with the symptom.
Gather evidence.
Change one thing at a time.

RTFM is a field guide for people who run Linux in the real world: on a school lab PC, a home server on a patchy link, or a fleet in a data centre. Part I teaches a calm, repeatable method for learning, doing and recovering. Part II is a reference library of 190 commands, each with what it tells you and the mistake to avoid.

FOSSLA.ORG · EDITION 1.0 · OCTOBER 2026