🐧 Basic Linux Shell Navigation and System Commands

This documentation explains basic Linux shell commands for navigation, file management, user management, and system monitoring. Each command is described individually, including syntax, function, and examples.

🐧 Introduction: What is Linux?

🔧 What is Linux?

Linux is a free and open-source operating system based on the Unix principle. It consists of several components:

- **Kernel**: The core of the system – controls hardware access and processes.
- **Shell**: The interface between the user and the system (e.g., Bash).
- **GNU Tools**: Fundamental system programs (e.g., `ls`, `cp`, `mkdir`).
- **Filesystem**: Everything is treated as a file – including devices and processes.

Linux has a modular design: package management, network tools, etc., are individually interchangeable and configurable because all packages consist of "text files."

📂 Directory Structure (Examples)

Directory	Function
`/`	Root directory
`/bin`	System commands for all users
`/sbin`	System commands for administrators
`/etc`	Configuration files
`/home`	User home directories
`/var`	Variable data (logs, caches, etc.)
`/usr`	User programs, documentation
`/tmp`	Temporary files
`/boot`	Bootloader & kernel files
`/dev`	Devices (e.g., hard drives, USB)
`/proc`, `/sys`	Virtual kernel information
`/opt`	Optional installed third-party software

📁 1. Filesystem Navigation

`pwd` – *Print Working Directory*

```
pwd
```

Shows the path of the current working directory.

`cd` – *Change Directory*

```
cd [directory]
```

Changes to another directory.

Examples:

- `cd /etc` – Absolute path
- `cd ..` – One directory up
- `cd ../..` – Two directories up
- `cd` – Change to home directory
- `cd -` – Return to the previous directory

`ls` – *List*

```
ls [options] [path]
```

Shows the content of a directory.

Important options:

- `-l` – Long format (permissions, owner, size)
- `-a` – Also show hidden files
- `-h` – Human-readable sizes
- `-R` – Recursively list subfolders

Example:

```
ls -lah /etc
```

📅 2. Working with Files and Directories

`mkdir` – *Create Directory*

```
mkdir [folder name]
```

Creates a new directory.

Use `-p` for nested folders as well:

```
mkdir -p projekt/src/data
```

`rm` – *Delete Files or Folders*

```
rm [file/directory]
```

Deletes files. Use `-r` for directories as well.

Warning:

```
rm -rf /wichtige_daten/
```

deletes everything recursively and without a prompt!

`cp` – *Copy*

```
cp source destination
```

Copies a file or a directory.

Important options:

- `-r` – Recursive (for directories)
- `-i` – Prompt before overwriting

Example:

```
cp -r projekt/ backup/
```

`mv` – *Move or Rename*

```
mv source destination
```

Moves or renames files/directories.

Example:

```
mv urlaub.jpg bilder/
mv alt.txt neu.txt
```

`nano` – *Text Editor*

```
nano datei.txt
```

A simple editor directly in the terminal.

Most important key combinations:

- `CTRL+O` – Save
- `CTRL+X` – Exit
- `CTRL+W` – Search

🔐 User Administration in Linux – Basics (from root's perspective)

🔹 1. The root User

- **root** is the system administrator with full privileges.
- **Root** can do anything: change system files, manage users, stop services, etc.
- In many Linux distributions like Ubuntu, the root user exists, but:
  - The **root** password is disabled.
  - Direct login as **root** is blocked.
  - Instead, a normal user with **sudo** privileges is set up during the first system boot.

> ✅ Note: During installation on Ubuntu, a user is automatically created who receives admin rights via `sudo`. The root account remains disabled and cannot log in directly to the shell.

🔹 2. Manually Creating a User (if only root is present)

If you are logged in directly as `root` (e.g., on a server or a minimal image) and want to create a new user:

```
useradd -m -s /bin/bash benutzername
passwd benutzername
```

- `-m`: Automatically creates a home directory (`/home/benutzername`).
- `-s /bin/bash`: Sets the login shell to Bash.
- `passwd`: Sets the password for the user.

> Note: In Debian-based systems, the `adduser` command is used instead of `useradd` (Proxmox is based on Debian). It's always helpful to check the relevant documentation here 😃

◇ 3. Granting sudo Privileges to the User

To allow the new user to perform administrative tasks, they must be added to the `sudo` group (works on Ubuntu and Debian-based systems):

```
usermod -aG sudo benutzername
```

> 💡 This means: Members of the `sudo` group are allowed to execute administrative commands via `sudo`.

This rule is in the file:

```
/etc/sudoers
```

The following line is typically found there:

```
%sudo   ALL=(ALL:ALL) ALL
```

It allows all users in the `sudo` group to execute any command as any user.

◇ 4. Editing the sudoers File – Only with `visudo`

What is `visudo`?

`visudo` is a special program for editing the `/etc/sudoers` file, which specifies which users or groups can use `sudo` and with what privileges.

Why use visudo?

It checks the file's syntax before it is saved.

This prevents the `sudo` function from becoming unusable due to typos or incorrect entries.

If you edit `/etc/sudoers` with a normal editor (`nano`, `vim`), errors can block the system so that no more admin commands are possible.

Usage The `/etc/sudoers` file is security-critical. Even a small error can render the system unusable.

Therefore, you should always use:

```
visudo
```

◇ 5. What does "sudoers" mean?

> ☑ "sudoers" is a fixed term for: the `/etc/sudoers` configuration file, which defines who can use `sudo`. the users (and groups) listed in it.

◇ **Example**: New user with `sudo` rights on Ubuntu Server

As root:

```
useradd -m -s /bin/bash alice
passwd alice
usermod -aG sudo alice
```

Now Alice can log in and work with `sudo`:

```
sudo apt update
```

❓ 7. Help and Documentation

`man` – *Manual Pages*

```
man command
```

Shows the official help for a command.

`--help` – *Brief Info*

```
ls --help
```

Shows options and a brief description directly in the terminal.

🌐 8. Network and System Control Commands

scp – Secure Copy Protocol

```
scp source destination
```

Securely copies files between two computers over SSH.

Examples:

```
scp file.txt user@remote:/home/user/
scp -r folder/ user@192.168.1.10:/srv/data/
```

- `-r`: Recursive (for directories)
- Requires SSH access to the target system.

ping – Test Network Connection

```
ping [target]
```

Sends ICMP packets to a target (IP or domain) to check for reachability.

Examples:

```
ping 8.8.8.8          # Ping an IP address (Google DNS)
ping www.google.com   # Ping a domain
```

📡 Why `ping 192.168.137.1` (local router / gateway)?

Tests whether the connection to the local network is working.

If this `ping` fails, the problem lies with your own computer or LAN/WLAN.

🌍 Why `ping 8.8.8.8` (Google DNS)?

Tests whether a connection to the internet exists.

If this `ping` works, the LAN is fine and the internet connection is active – regardless of DNS.

🌐 Why `ping heise.de` (domain)?

Tests whether name resolution (DNS) is working correctly.

If the IP `ping` works but the domain `ping` does not, the problem is with the DNS resolver.

👉 In short:

- Local (Gateway) → Is my network cable/WLAN working?
- External by IP → Do I have internet access?
- External by Domain → Is DNS working?

Exit with `CTRL+C`.

shutdown – Shut down or restart the system

```
shutdown [option] [time] [message]
```

Examples:

```
shutdown now            # Shut down immediately
shutdown -h now         # Shut down and halt
shutdown -r now         # Restart (reboot)
shutdown -P now         # Shut down and power off
shutdown +10 "Maintenance"  # Shut down in 10 minutes with a message
```

Important options:

Option	Meaning
`-h`	Shut down (halt)
`-P`	Power off
`-r`	Reboot
`+min`	Delay in minutes
`now`	Immediately
`cancel`	Cancel a scheduled shutdown

Cancel a scheduled shutdown:

```
shutdown -c
```

reboot – Restart the system

```
reboot
```

Restarts the system immediately. Internally, it is equivalent to:

```
shutdown -r now
```

poweroff – Turn off the system

```
poweroff
```

Completely shuts down the system and turns off the power (if the hardware supports it). Equivalent to:

```
shutdown -P now
```

⏰ 9. Time and Synchronization in Linux

timedatectl – Manage system time and time zones

`timedatectl` is a command-line tool to query and change time and date settings.

Examples:

```
timedatectl status
```

👉 Shows current time, time zone, and whether NTP is enabled.

```
timedatectl set-timezone Europe/Berlin
```

👉 Changes the time zone.

```
timedatectl set-time "2025-08-18 14:30:00"
```

👉 Manually sets the time and date.

timesyncd – Synchronize time via NTP

`systemd-timesyncd` is a simple NTP client included by default in modern Linux systems (e.g., Ubuntu, Debian).

It ensures that the system clock is automatically synchronized with internet time servers (NTP servers).

Check if it's active:

```
timedatectl timesync-status
```

or:

```
systemctl status systemd-timesyncd
```

👉 Relationship:

`timedatectl` = a tool to set time/date/timezone and control NTP.

`timesyncd` = the background service that performs the actual automatic time synchronization.

🜂 10. Further Topics

Once you master the basics, you can expand your knowledge with the following topics:

- Bash scripting
- Package management (`apt`, `dnf`, `pacman`)
- Network analysis (`curl`, `netstat`, `ss`)
- Systemd & services (`systemctl`)
- Log files (`journalctl`, `/var/log`)
