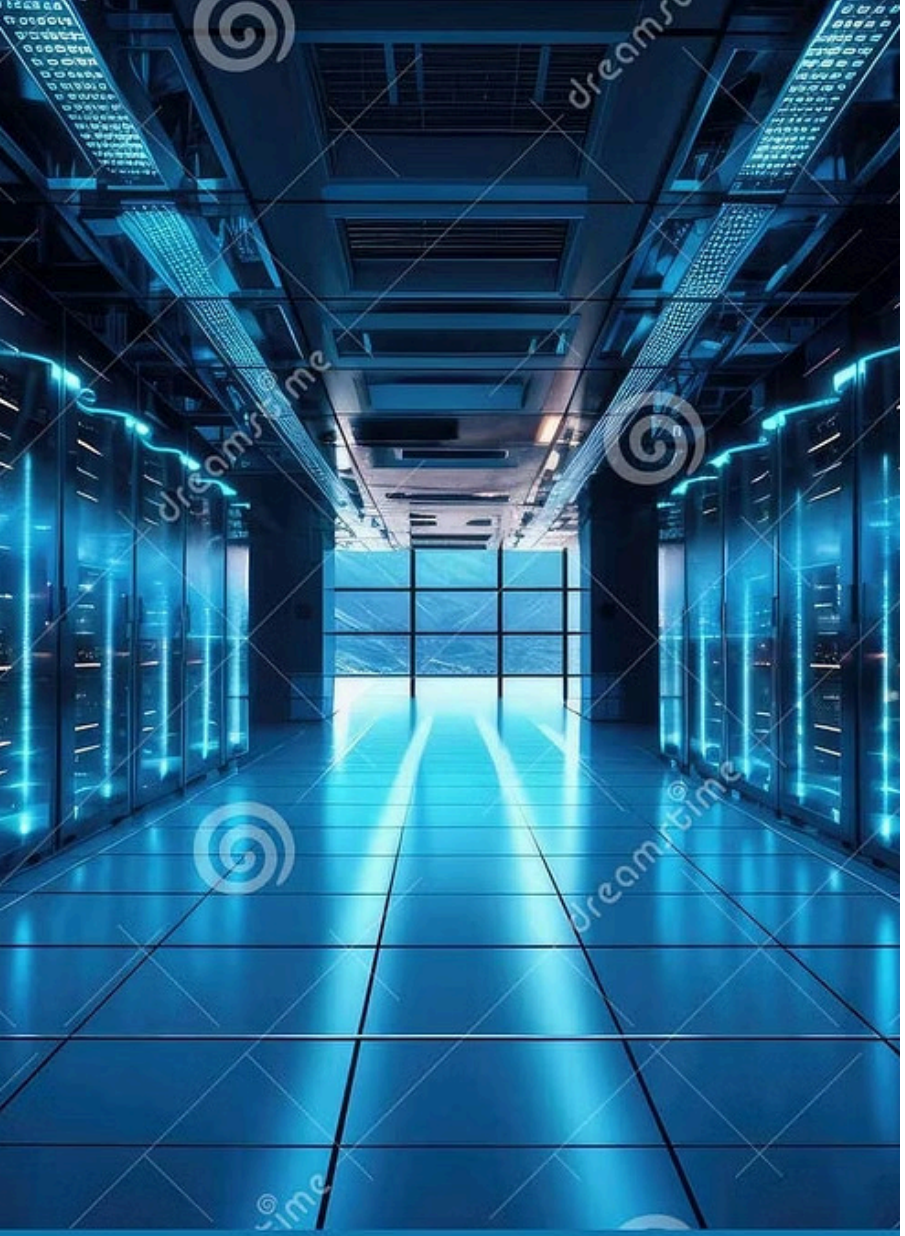


Linux Fundamentals: Architecture, File System, Commands, Permissions & More

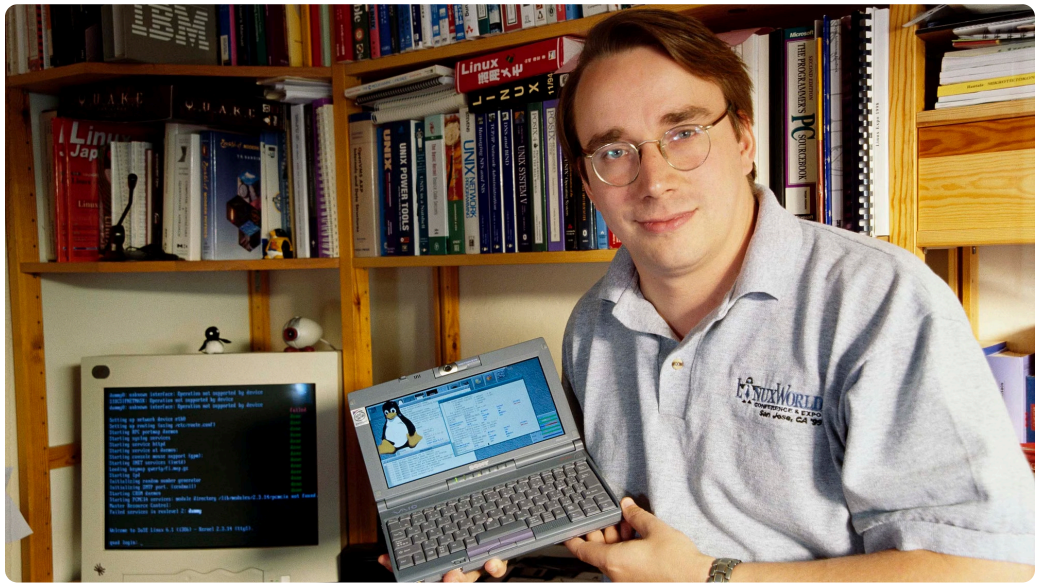
A comprehensive guide to understanding the powerful open-source operating system that powers the modern digital world.

CHAPTER 1

Understanding Linux Architecture

Explore the fundamental building blocks that make Linux one of the most versatile and powerful operating systems available today.

What is Linux?



Linux is an open-source, Unix-like operating system kernel created by Linus Torvalds in 1991. What began as a personal project has evolved into the backbone of modern computing infrastructure.

Today, Linux powers an extraordinary range of devices—from the smartphone in your pocket to the world's most powerful supercomputers and the servers hosting your favourite websites.

Linux Kernel: The Core

Hardware Management

Manages CPU, memory, and device resources with exceptional efficiency

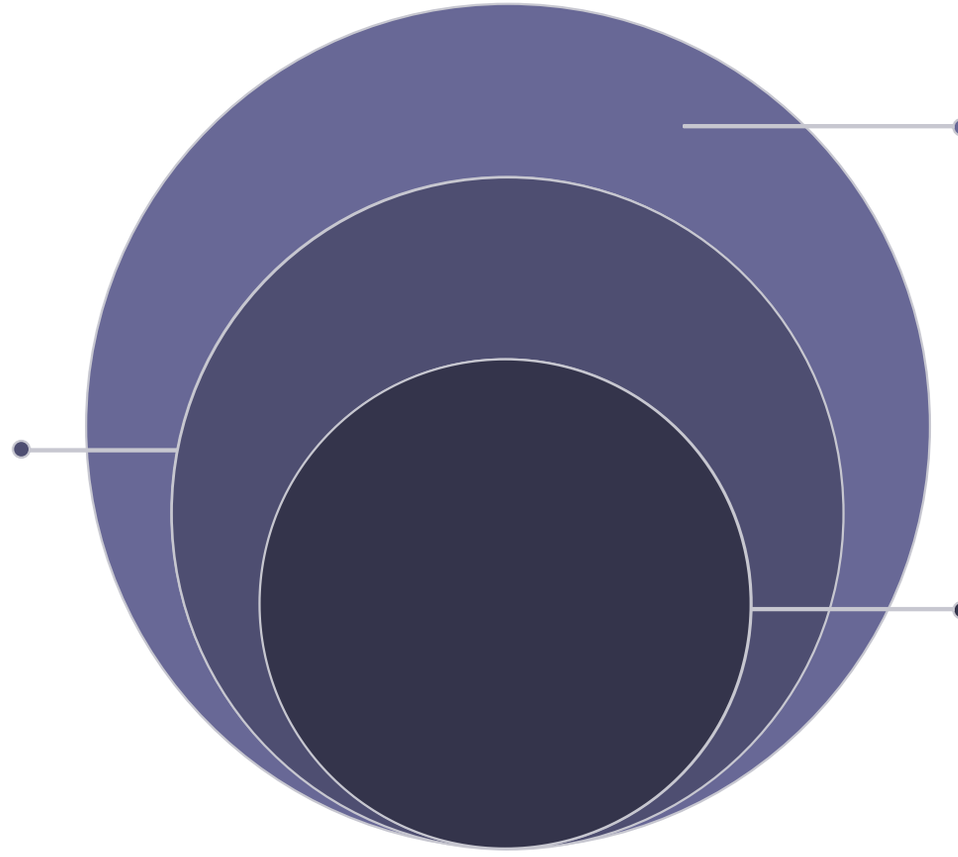
System Calls

Provides interfaces for user programmes to interact with hardware safely

Modular Design

Supports loadable kernel modules for flexible functionality without reboots

User Space vs Kernel Space



System Calls

Controlled bridge for requests between spaces

User Space

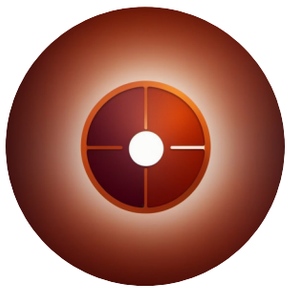
Isolated applications running without hardware access

Kernel Space

Privileged mode with direct hardware access

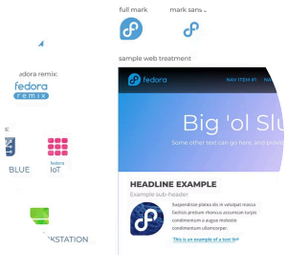
The separation between kernel and user space is fundamental to Linux security and stability. Kernel space operates in privileged mode with direct hardware access, whilst user space runs applications in isolation. Communication between these spaces occurs through carefully controlled system calls.

Linux Distributions: Variants for Every Need



Ubuntu

User-friendly, ideal for beginners and desktop use



Fedora

Cutting-edge features, sponsored by Red Hat

Debian

Rock-solid stability, extensive package repository

CentOS

Enterprise-ready, server-focused distribution

Arch Linux

Minimal, customisable, for advanced users

Each distribution bundles the Linux kernel with carefully selected software, package managers, and tools tailored to specific use cases.

Linux Architecture Layers

Hardware Layer

Physical components: CPU, RAM, storage devices, peripherals

Kernel Layer

Core operating system managing resources and system calls

System Libraries

Standard libraries providing common functions to applications

User Applications

Programs and tools users interact with daily

CHAPTER 2

Linux File System Hierarchy

Understanding the Linux file system structure is essential for navigating and managing your system effectively.

The Linux File System Tree

Everything Starts at Root

Unlike Windows with multiple drive letters (C:, D:), Linux uses a single unified directory tree beginning at the root directory `/`.

This elegant design means every file, directory, and device has a unique path from the root, creating a logical and consistent structure across all Linux systems.

Everything is a File

In Linux, an extraordinary principle applies: *everything is treated as a file*. This includes:

- Regular files and directories
- Hardware devices
- Network sockets
- Named pipes

Key Directories Explained

/bin

Essential command binaries needed for system boot and repair (ls, cp, mv)

/etc

System-wide configuration files and scripts

/home

Personal directories for each user storing documents and settings

/var

Variable data including logs, databases, and temporary files

/usr

User applications, libraries, and documentation

/dev

Device files representing hardware components

File Types in Linux

Regular Files

Text, binaries, images—standard data files

Directories

Containers for organizing files

Symbolic Links

Shortcuts pointing to other files

Special Files

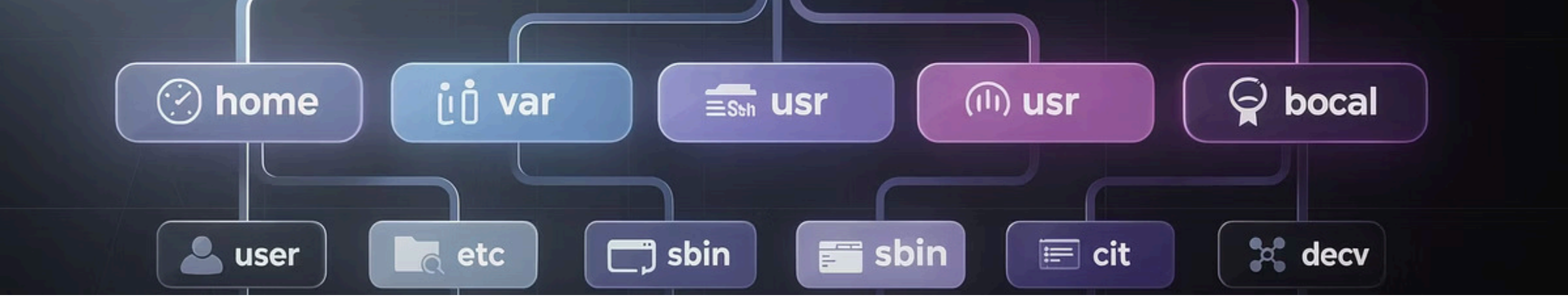
- **Block devices:** Storage devices (hard drives, USB)
- **Character devices:** Serial devices (keyboards, mice)
- **Sockets:** Inter-process communication endpoints
- **Named pipes:** FIFO data channels between processes

Mount Points and Devices

Linux integrates external storage devices seamlessly into the file system tree through mounting. When you connect a USB drive, it doesn't appear as a separate drive letter—instead, it's mounted at a specific location.

Example: A USB drive might be mounted at `/media/usb`, making its contents accessible as if they were part of the main file system.

This unified approach means you access all storage using the same familiar path structure.



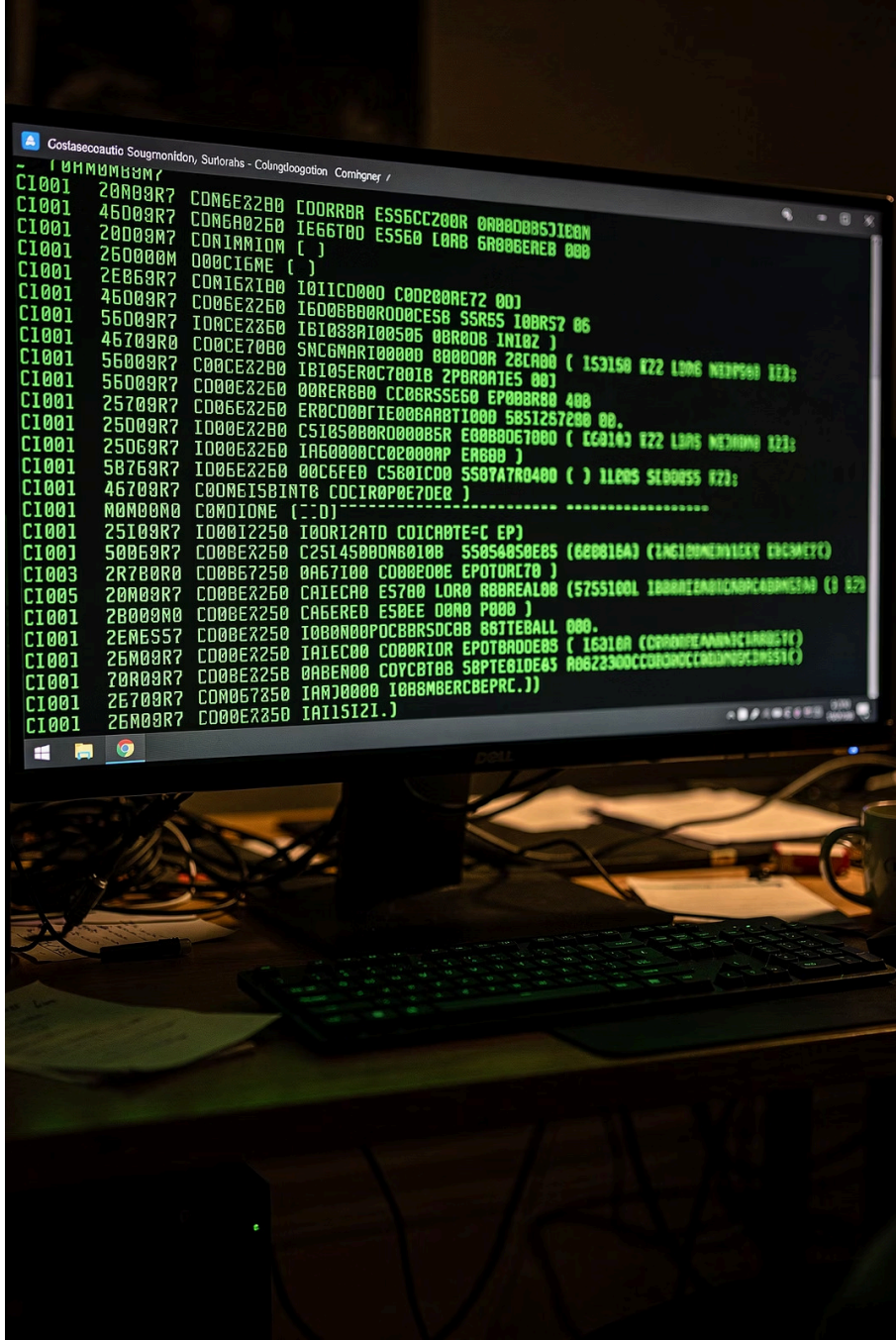
Linux Directory Tree Visual

The hierarchical structure begins at root (/) and branches into specialized directories, each serving a specific purpose in system organization and functionality.

CHAPTER 3

Basic Linux Commands

Master the essential commands that form the foundation of Linux system interaction and administration.



Navigating the File System

pwd

Print working directory—shows your current location in the file system

ls

List directory contents with various options for detailed information

cd

Change directory to navigate between folders efficiently

File and Directory Management

Creating

- `touch filename` creates an empty file
- `mkdir dirname` creates a new directory
- `mkdir -p path/to/dir` creates nested directories

Manipulating

- `rm file` removes files
- `rm -r dir` removes directories recursively
- `cp source dest` copies files
- `mv source dest` moves or renames files

Viewing and Editing Files

cat

Display entire file contents quickly

less

View files page by page with navigation

head

Show first 10 lines of a file

tail

Display last 10 lines, useful for logs

nano

Simple, user-friendly terminal text editor

vim

Powerful modal editor with steep learning curve

Searching and Finding Files

find Command

Locate files by name, type, size, modification time, and more across the entire file system.

Examples:

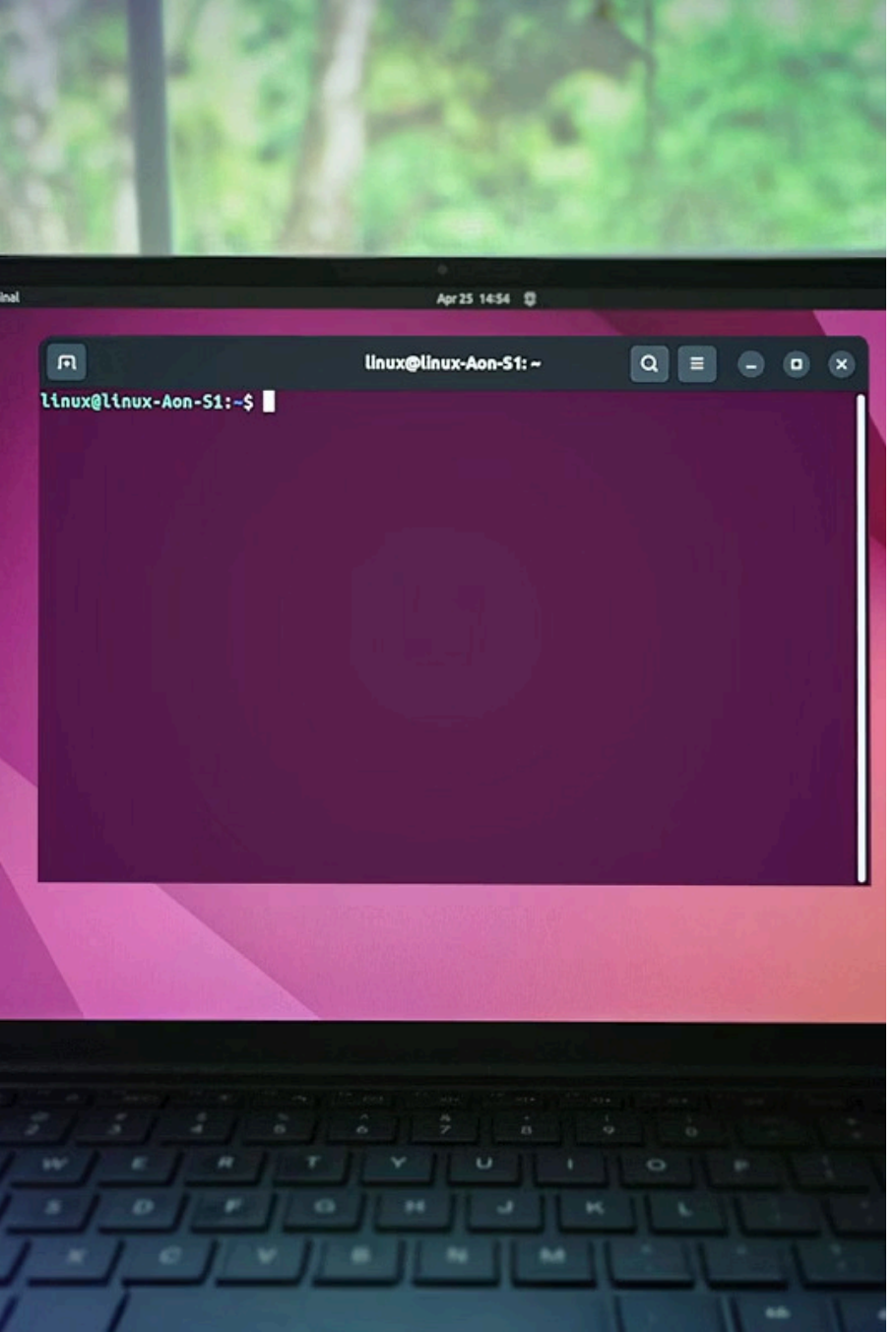
```
find /home -name "*.txt"
find . -type f -size +10M
find /var -mtime -7
```

grep Command

Search text patterns inside files—incredibly powerful for finding specific content.

Examples:

```
grep "error" logfile.txt
grep -r "TODO" /project
grep -i "warning" *.log
```



Commands in Action

The terminal is where Linux truly shines, offering precise control and automation capabilities that graphical interfaces simply cannot match.

CHAPTER 4

Linux File Permissions and Ownership

Understanding permissions is crucial for system security and proper multi-user operation.

Why Permissions Matter

Critical for Security

Permissions protect your system from unauthorized access and accidental damage.

In a multi-user environment, proper permissions ensure that:

- Users can only access their own files and authorized shared resources
- System files remain protected from accidental modification
- Sensitive data stays confidential and secure
- Programmes run with appropriate privilege levels

Misconfigured permissions are a common source of security vulnerabilities and system instability.

Permission Types: Read, Write, Execute

Read (r)

View file contents or list directory contents

Write (w)

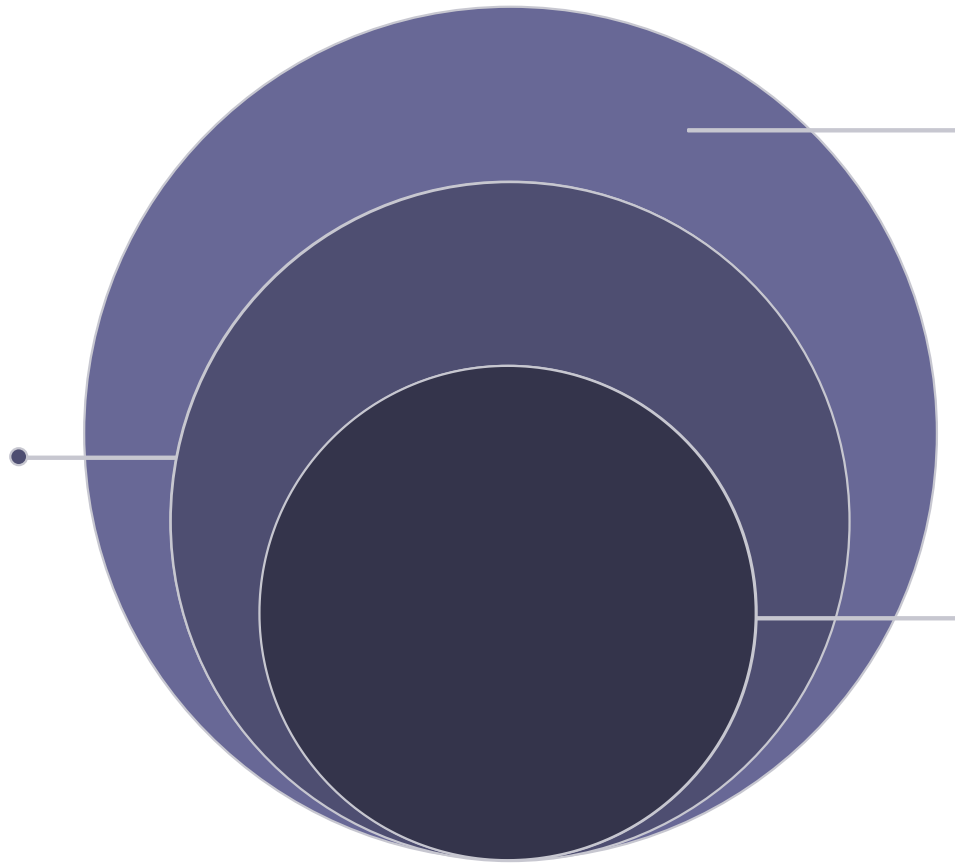
Modify file contents or create/delete files in a directory

Execute (x)

Run a file as a programme or enter a directory

These three permission types combine to control exactly what actions can be performed on files and directories.

Permission Groups: User, Group, Others



Others

Limited access for all others

Group (Team)

Shared access for group members

User (Owner)

Full control by file owner

Linux applies permissions separately to three categories: the file owner (user), members of the file's group, and all other users on the system. This granular control enables flexible security policies.

Viewing Permissions with ls -l

```
-rwxr-xr-- 1 alice devs 4096 Jan 29 file.sh
```

01

File Type

First character: - (file), d (directory), l (link)

02

User Permissions

Characters 2-4: rwx (owner has read, write, execute)

03

Group Permissions

Characters 5-7: r-x (group has read and execute)

04

Other Permissions

Characters 8-10: r-- (others have read only)

05

Ownership

alice (owner), devs (group)

Changing Permissions: chmod

Symbolic Mode

Use letters to specify changes clearly and intuitively:

```
chmod u+x file.sh
chmod g-w file.sh
chmod o=r file.sh
chmod a+r file.sh
```

- u=user, g=group, o=others, a=all
- +=add, -=remove, ==set exactly

Numeric Mode

Use octal numbers for precise control:

```
chmod 755 file.sh
chmod 644 document.txt
chmod 700 private.sh
```

- 4=read, 2=write, 1=execute
- 755 = rwxr-xr-x
- 644 = rw-r--r--

Changing Ownership: chown and chgrp

Change Owner

`chown bob file.txt`

Transfers ownership to user bob

Change Group

`chgrp devs file.txt`

Assigns file to the devs group

Change Both

`chown bob:devs file.txt`

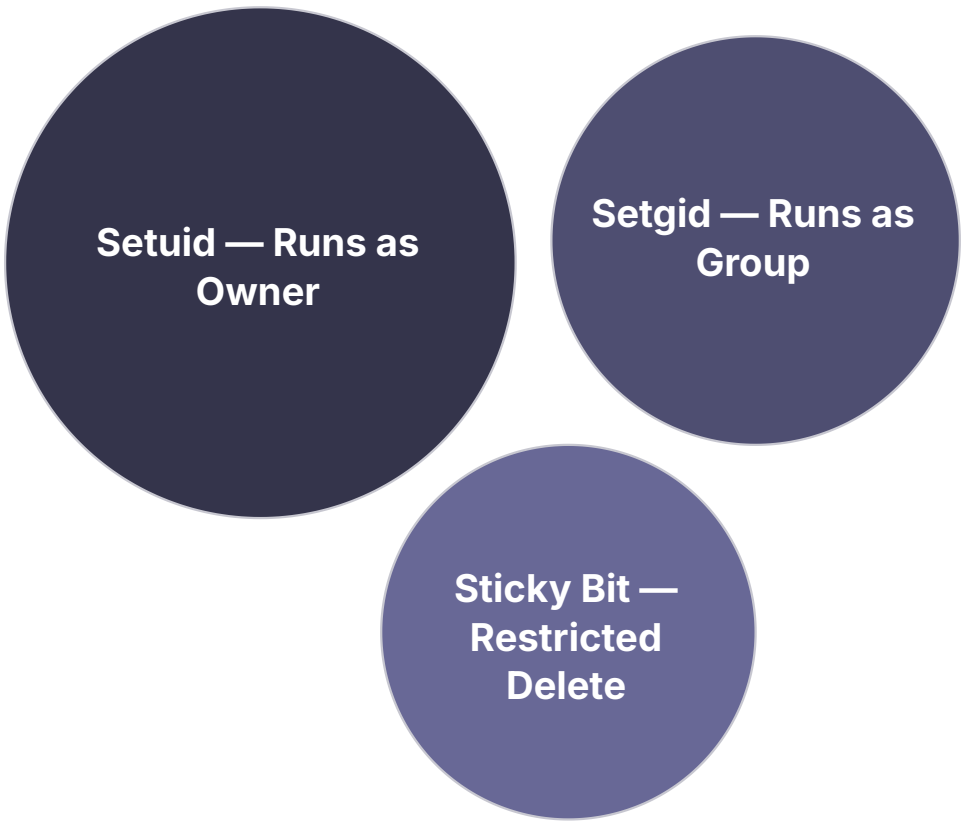
Sets owner and group simultaneously

Recursive Change

`chown -R bob:devs /directory`

Applies changes to all contents

Special Permission Bits



Setuid (Set User ID)

File executes with owner's privileges. Example: `passwd` command needs root access to modify /etc/shadow.

Setgid (Set Group ID)

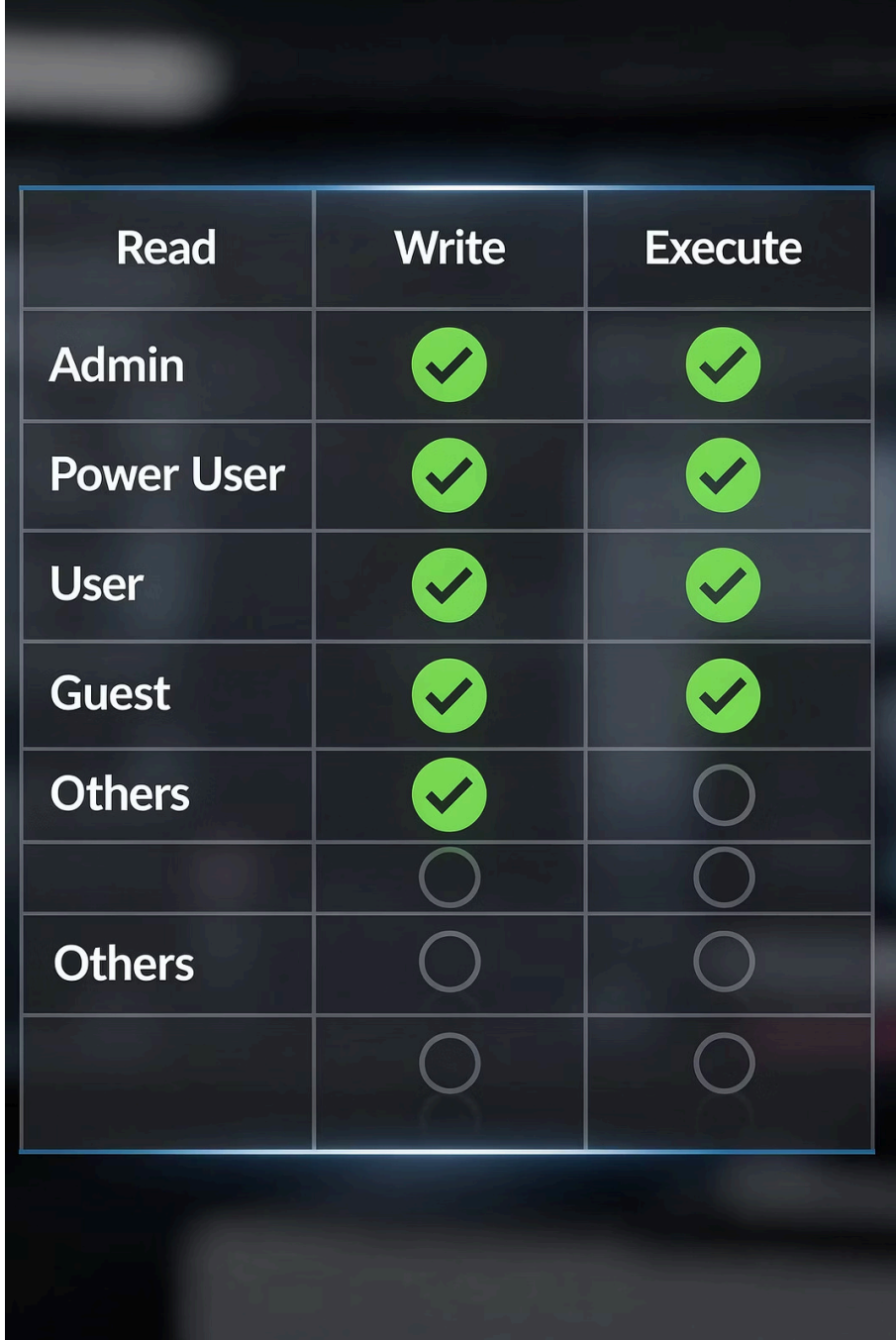
File executes with group privileges, or new files inherit directory's group.

Sticky Bit

In shared directories (like /tmp), only file owners can delete their own files, preventing accidental deletion.

Permission Matrix Visualization

The three permission groups (user, group, others) each have three permission types (read, write, execute), creating a flexible nine-bit system for access control.



CHAPTER 5

Process Management in Linux

Effectively managing processes is essential for system performance, troubleshooting, and resource optimization.

What is a Process?

A process is a running instance of a programme—it's what happens when you execute an application or command. Each process exists in its own protected memory space and has:

- A unique Process ID (PID) for identification
- Associated user and group ownership
- Allocated memory and CPU resources
- A current working directory and environment variables
- Open file handles and network connections

Parent & Child

Processes can spawn child processes, creating a hierarchical tree structure with the **init** process (PID 1) at the root.

Viewing Processes

ps Command

Takes a snapshot of current processes. Use `ps aux` for detailed information about all processes.

top Command

Provides dynamic real-time view of running processes, updating continuously with CPU and memory usage.

htop Command

Enhanced version of top with colour-coded display, easier navigation, and more intuitive interface.

Managing Processes

Sending Signals

Control processes by sending signals:

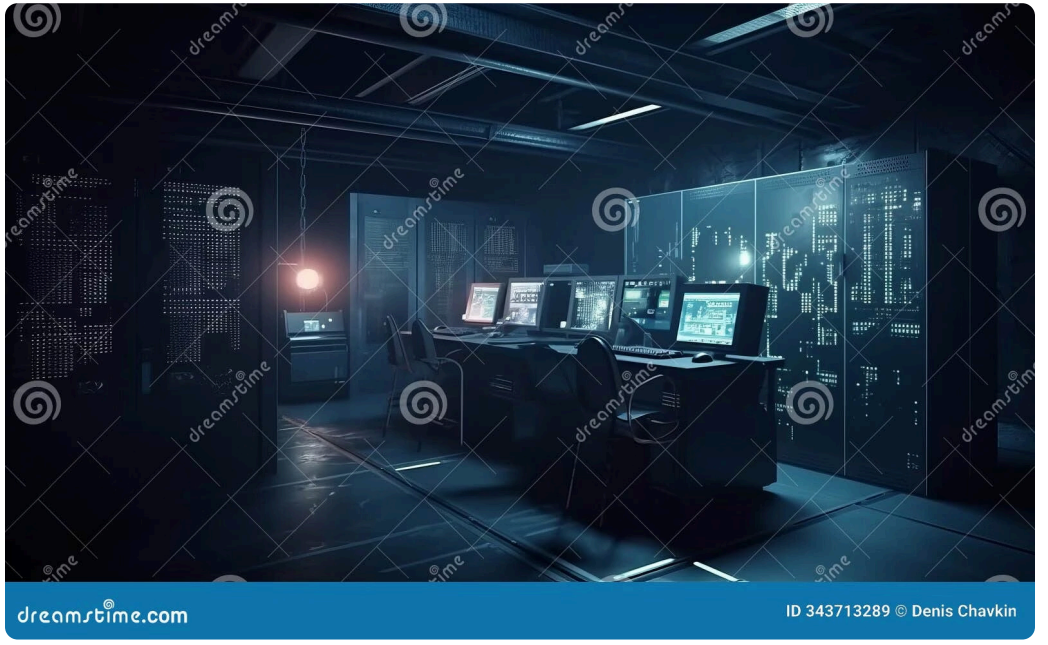
- `kill PID` - terminate process
- `kill -9 PID` - force kill
- `killall name` - kill by name
- `pkill pattern` - kill by pattern

Priority & Job Control

Adjust process priority:

- `nice -n 10 command` - start with lower priority
- `renice 5 -p PID` - change priority of running process
- `jobs` - list background jobs
- `fg, bg` - move jobs between foreground/background

Daemons and Services



Daemons are background processes that run continuously, providing essential system functions without user interaction. They typically start at boot and handle services like web servers, databases, and system logging.

Managing with systemctl:

```
systemctl start nginx
systemctl stop nginx
systemctl restart nginx
systemctl status nginx
systemctl enable nginx
```

Process Lifecycle and States

Processes transition through various states: running (actively using CPU), sleeping (waiting for events), stopped (suspended), and zombie (completed but not cleaned up). Understanding these states helps diagnose system behaviour.

CHAPTER 6

Linux Logs and Monitoring

Logs are your window into system health, security events, and troubleshooting information.

Importance of Logs

Troubleshooting

Track down errors, failures, and unexpected behaviour with detailed event records

Security

Detect unauthorized access attempts, suspicious activities, and security breaches

Performance

Monitor system performance, identify bottlenecks, and optimize resource usage

Compliance

Maintain audit trails required by regulatory standards and organizational policies

Key Log Files and Locations

/var/log/syslog (or messages)

General system activity and messages from various services

/var/log/auth.log

Authentication attempts, user logins, sudo usage, and security events

/var/log/kern.log

Kernel messages, hardware issues, and driver information

/var/log/apache2/ or /nginx/

Web server access logs and error logs

/var/log/dmesg

Boot messages and hardware detection information

Viewing Logs

Real-Time Monitoring

Watch logs as events occur:

```
tail -f /var/log/syslog
```

The `-f` flag follows the file, displaying new entries as they're written—essential for live troubleshooting.

Systemd Journal

Query structured logs with journalctl:

```
journalctl -u nginx
journalctl --since "1 hour ago"
journalctl -p err
journalctl -f
```

Provides powerful filtering and searching capabilities.

Log Rotation

Preventing Disk Exhaustion

Logs can quickly consume available storage if left unchecked.

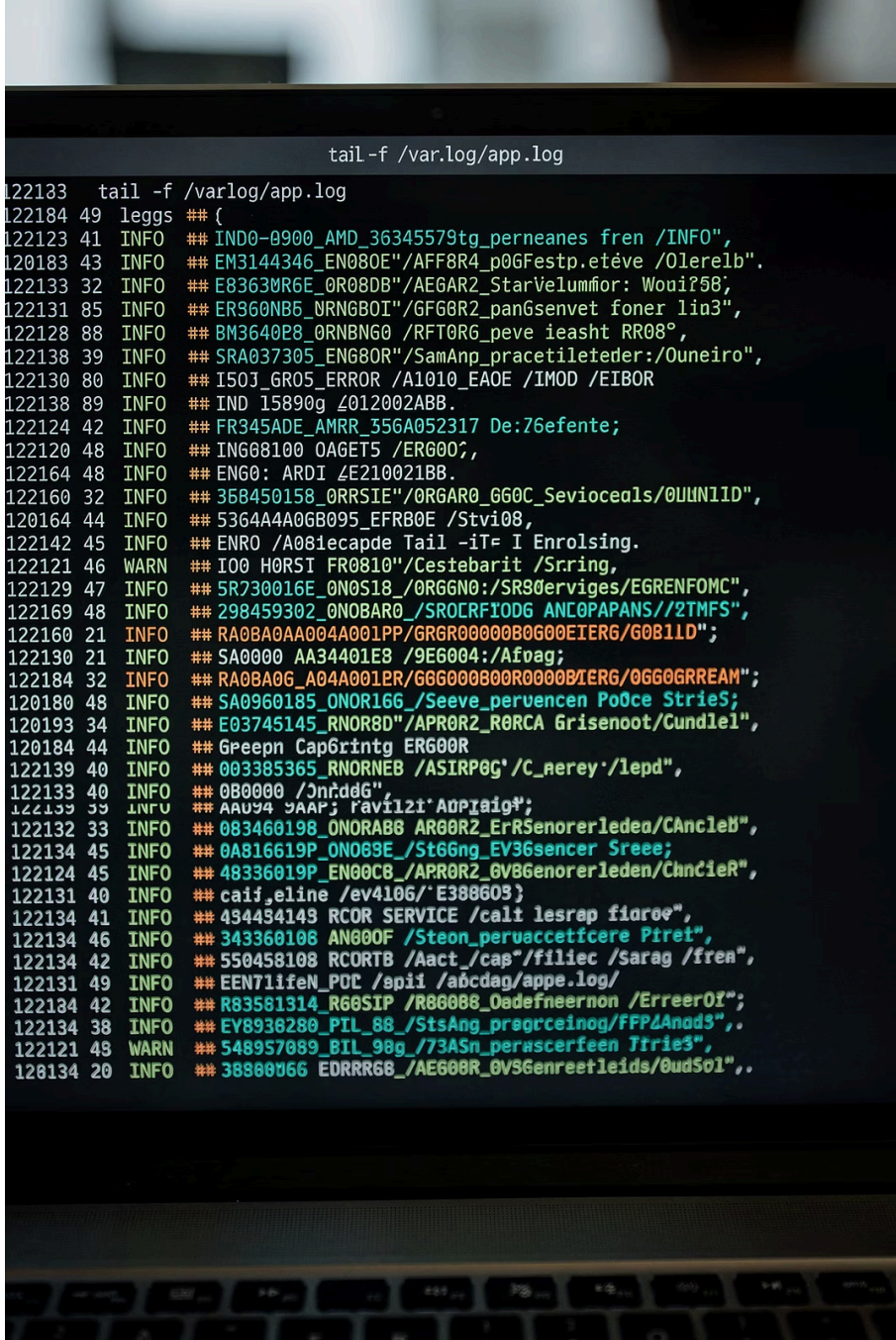
The **logrotate** utility automatically manages log files by:

- Rotating logs at specified intervals (daily, weekly, monthly)
- Compressing old log files to save space
- Deleting ancient logs beyond retention periods
- Creating new log files for continued logging

Configuration files in `/etc/logrotate.d/` define rotation policies for each service.

Tailing Logs in the Terminal

Real-time log monitoring is crucial for diagnosing issues as they occur, allowing administrators to see exactly what's happening in the system moment by moment.



CHAPTER 7

Networking Basics in Linux

Understanding Linux networking is fundamental for system administration, troubleshooting connectivity, and securing your infrastructure.

Network Interfaces and Configuration

Viewing Interfaces

```
ip addr show
ip link show
ifconfig
```

Display network interfaces, IP addresses, MAC addresses, and connection status. Modern systems prefer `ip` command over deprecated `ifconfig`.

Common Interface Types

- **eth0, enp0s3:** Ethernet interfaces
- **wlan0:** Wireless interfaces
- **lo:** Loopback (localhost, 127.0.0.1)
- **docker0, br0:** Bridge interfaces for virtualization

Common Networking Commands

ping

Test network connectivity and measure round-trip time to hosts

```
ping google.com
```

netstat / ss

Display network connections, routing tables, and interface statistics

```
ss -tuln
```

traceroute

Trace the network path packets take to reach a destination

```
traceroute google.com
```

nslookup / dig

Query DNS servers to resolve domain names to IP addresses

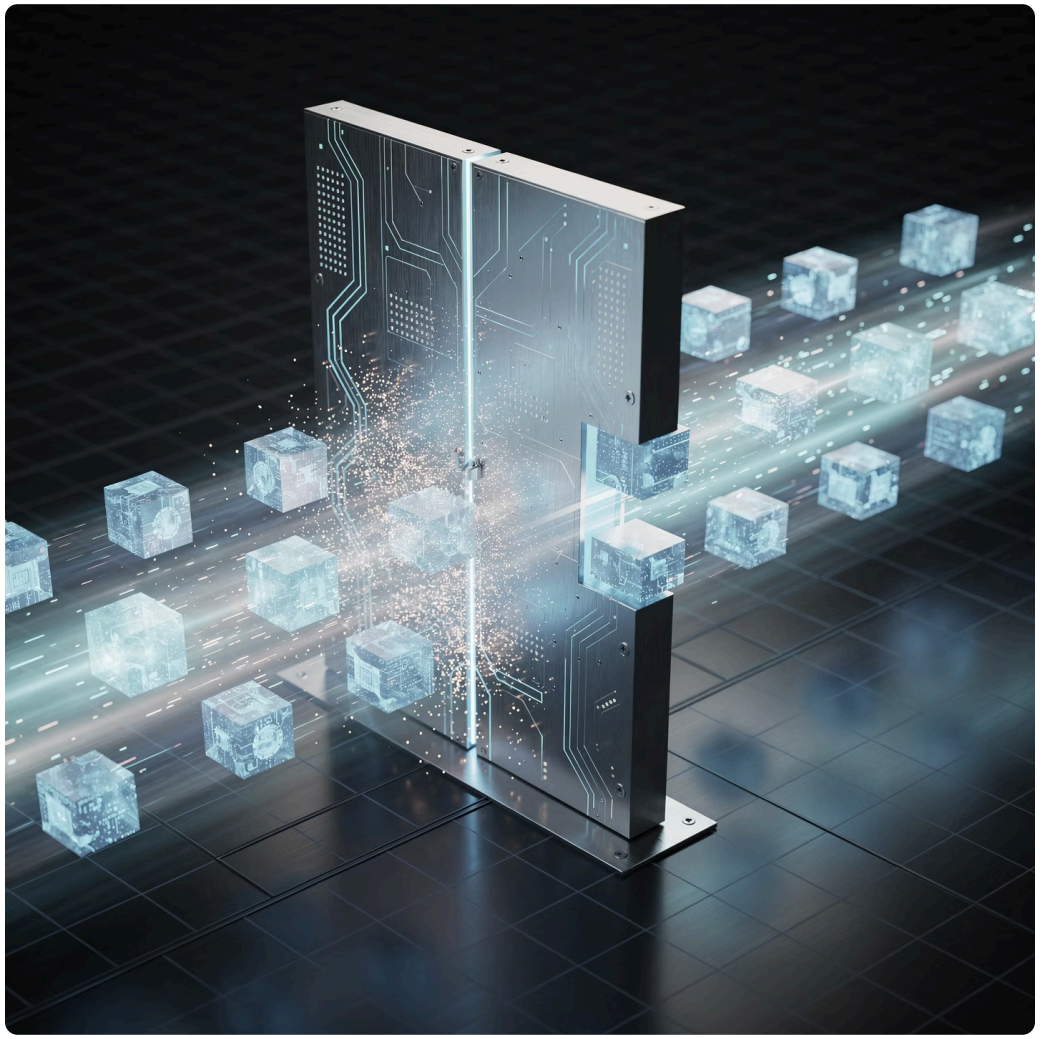
```
dig example.com
```

wget / curl

Download files and test HTTP requests from command line

```
curl -I https://example.com
```

Managing Firewalls



Firewall Tools

iptables: Low-level packet filtering framework, very powerful but complex syntax.

firewalld: Higher-level firewall management with zones and services, easier to configure.

Common Tasks

- Allow/block specific ports (e.g., 80, 443 for web)
- Control traffic by protocol (TCP, UDP, ICMP)
- Create rules based on source/destination IPs
- Establish default policies (allow or deny)

SSH: Secure Remote Access

01

Basic Connection

`ssh user@hostname` establishes encrypted remote shell session

02

Key-Based Authentication

Generate SSH keys with `ssh-keygen` for passwordless, more secure login

03

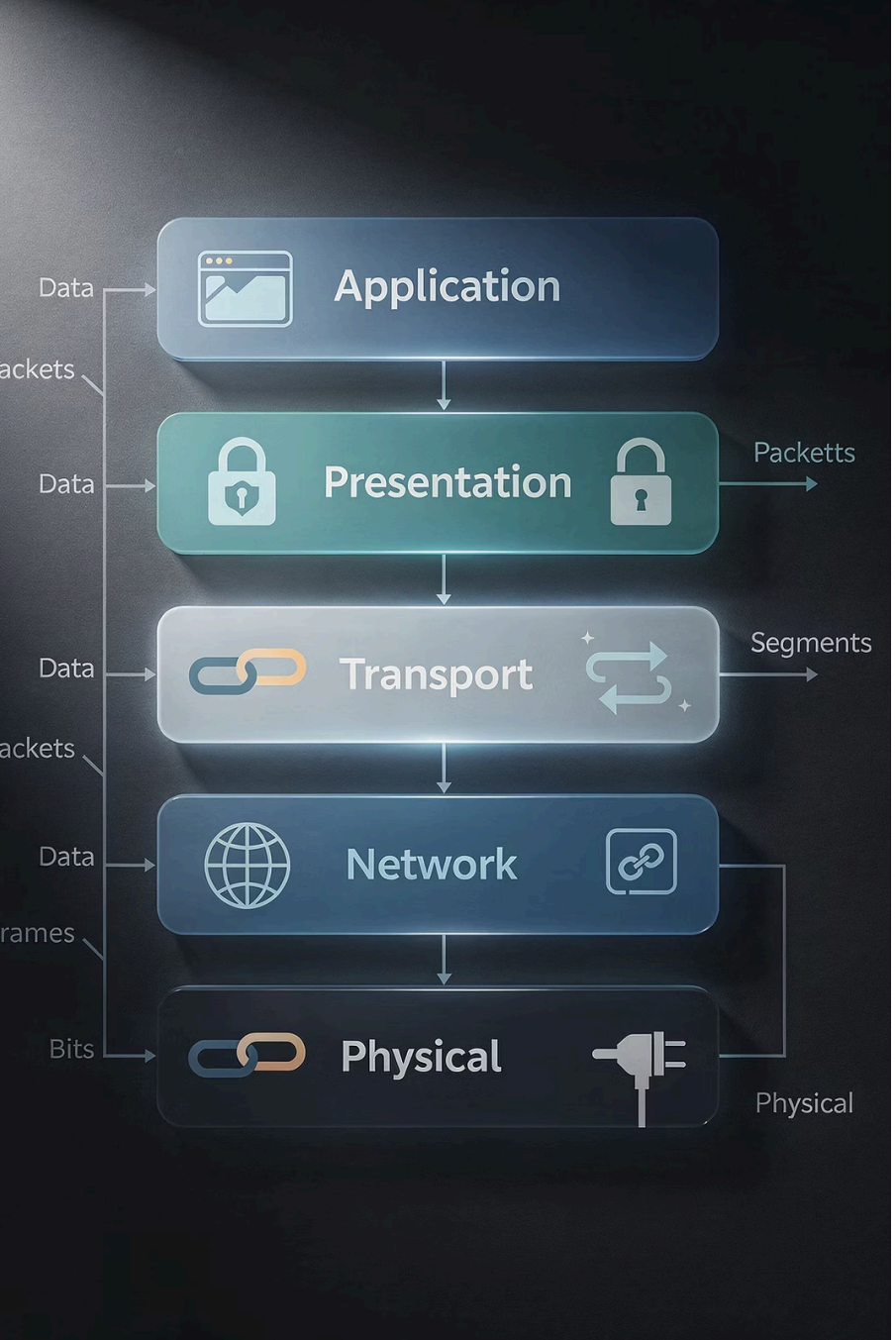
Copy Public Key

Use `ssh-copy-id` to install your public key on remote servers

04

Additional Features

Port forwarding, tunneling, file transfer with `scp` and `sftp`



Linux Networking Stack Overview

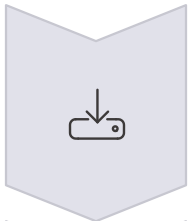
The Linux networking stack handles communication from hardware devices through protocols to applications, implementing the full TCP/IP protocol suite with exceptional performance and flexibility.

CHAPTER 8

Putting It All Together: Practical Linux Usage

Real-world scenarios demonstrate how the concepts we've covered work together in practice.

Example: Setting Up a Web Server

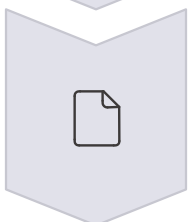


Install Web Server

`apt install apache2` or `yum install nginx`

Configure Permissions

Set appropriate ownership and permissions for web files: `chown -R www-data:www-data /var/www`



Monitor Logs

Watch access and error logs: `tail -f /var/log/apache2/error.log`

Secure with Firewall

Open ports 80 and 443, enable SSH for remote management

Test and Deploy

Verify configuration, enable service to start at boot

Tips for Effective Linux Use

Read the Manual

Use `man command` to access comprehensive documentation. The man pages are your best friend for learning command options.

Principle of Least Privilege

Only grant the minimum permissions necessary. Avoid running as root unless absolutely required.

Regular Monitoring

Check system health, logs, and resource usage regularly to catch issues before they become critical.

Backup Important Data

Implement automated backup strategies and test restoration procedures regularly.

Keep Systems Updated

Apply security patches promptly and maintain up-to-date software packages.

Practice in Safe Environments

Use virtual machines or containers to experiment and learn without risking production systems.

Conclusion: Mastering Linux Basics Empowers You

Linux is a powerful and flexible operating system with a learning curve that rewards persistence and curiosity. Understanding its architecture, command-line tools, permission system, and networking fundamentals provides the foundation for system administration, development, and cybersecurity careers.

The journey doesn't end here—keep exploring, practising, and building. The Linux community is vast and welcoming, with endless resources for continued learning. Every expert was once a beginner who refused to give up.

> *"The most valuable skill you can develop is the ability to learn. Master the fundamentals, and you'll adapt to any system."*

Thank you! Questions?