Linux

Kernel, System, and Ecosystem

What will we cover today?

- What Linux really is
- Birth, history, and key contributors
- Why GPL matters to Linux
- Why people use Linux (desktop & server)
- CLI, shell, and GNU utilities
- Users, permissions, and security model
- File systems and storage choices
- Kernel modules and extensibility
- Distributions and desktop environments

Section 1: What is Linux?

What we will see

- Meaning of “Linux”
- Kernel vs GNU/Linux
- Why Linux needs more than a kernel

Linux is a kernel

- Linux = **kernel**
- Runs in privileged mode
- Controls:
  - CPU
  - Memory
  - Devices
  - System calls

Kernel = authority, not user interface.

Linux alone is not usable

- No user interface
- No commands
- No applications

A kernel needs a **user space**.

- Part of the OS where applications execute. Runs with limited permissions.

GNU completes the OS

GNU provides:

- Shells (bash)
- Core utilities
- Libraries
- Development tools

Linux + GNU = GNU/Linux, an usable operating system

Why people say “Linux OS”

- Technically inaccurate
- Convenient shorthand
- Widely accepted

System designers must know the distinction.

Analogy: Engine vs vehicle

- Engine ≠ vehicle
- Linux kernel = engine
- OS = engine + controls + body

Section 1: Summary

- Linux is the kernel
- GNU provides user space
- Separation enables flexibility

Any questions?

Section 2: Birth, History, and Community of Linux

What we will see

- Why Linux was created
- Who else mattered besides Linus
- How large collaboration works

Before Linux

- Unix existed but was restricted
- Students couldn’t study real OS code
- Minix existed but had limits

There was a learning gap.

Why Linux was created

- Linus Torvalds wanted:
  - A Unix-like OS
  - For learning
  - On affordable hardware

Linux began as a learning project.

Who else besides Linus?

Linux succeeded because of **many projects**:

- GNU Project
- Free Software Foundation
- Global developer community

Linux was not built alone.

GNU Project and Stallman

- GNU started before Linux
- Goal: free Unix-like system
- Created:
  - Compilers
  - Libraries
  - Shells
  - Utilities

Without GNU, Linux would not be usable.

Other core projects in a Linux OS

A typical Linux system depends on:

- glibc (core C library)
- systemd (system and service manager)
- X.org / Wayland (graphics stack)
- GNU coreutils

Linux OS = ecosystem of projects.

How Linux collaboration works

- Anyone can contribute
- Maintainers review code
- Linus approves kernel changes

Open, but structured.

Section 2: Summary

- Linux is community-built
- Linux + GNU = OS

Section 3: Linux Licensing Model

What we will see

- Why Linux uses GPL
- Why this matters

Linux is GPL licensed

- Linux kernel uses **GPL**
- GPL = copyleft license, viral
- Guarantees user freedoms

License choice shapes the ecosystem.

What GPL ensures

If you distribute Linux:

- You must share source code
- You must keep the same freedoms
- You cannot make it proprietary

Freedom is preserved downstream.

Why GPL mattered for Linux

- Prevented enclosure
- Encouraged collaboration
- Protected community effort

GPL ensured Linux stayed open.

Think

How would Linux be different today if it used a permissive license?

Section 3: Summary

- GPL protects freedom
- License shaped Linux growth
- Philosophy affects engineering

Any questions?

Section 4: Why People Use Linux

What we will see

- Where Linux runs
- Why it is chosen
- Trade-offs involved

Linux on the desktop

- Freedoms and Free
- Customizable
- No forced upgrades
- Many other tools are available

Trade-off:

- More control
- More responsibility

Linux on servers

- Freedoms and Free
- Stability
- Automation
- Predictable behavior
- Remote management
- Many other tools are available

Linux dominates servers.

Same kernel, different roles

- Desktop
- Server
- Embedded systems

Differences come from configuration and tools.

Section 4: Summary

- Linux fits many roles
- Choice is intentional
- Control is central

Section 5: CLI, Shell, and GNU Utilities

What we will see

- Why CLI matters
- What the shell does
- GNU tool philosophy

Linux is CLI-first

- GUI exists
- CLI is central

CLI is power, not legacy.

The shell (bash)

- Command interpreter
- Starts programs
- Connects programs together

User → shell → programs → kernel

GNU core utilities

Examples:

- `ls`, `cp`, `mv`
- `cat`, `grep`, `sort`
- `ps`, `top`

Small tools, one job each.

Unix philosophy

- Do one thing well
- Combine tools
- Text as interface

Design for composition.

Section 5: Summary

- Shell is user interface to OS
- Utilities are OS building blocks
- Composition enables power

Any questions?

Section 6: Users, Permissions, and Security Model

What we will see

- Why Linux is multi-user
- How permissions work

Linux is multi-user by design

- Multiple users share one system
- Each user is isolated
- System resources are protected

Security is foundational, not optional.

Users and groups

- Users belong to groups
- Groups control access
- Permissions apply to files and processes

Ownership matters.

Root and sudo

- Normal users have limited rights
- `root` = all-powerful administrator
- `sudo` allows controlled privilege escalation

Power is deliberate and explicit.

Think

Why is it dangerous to run everything as administrator?

Section 6: Summary

- Linux enforces separation
- Permissions protect the system
- Power is carefully granted

Any questions?

Section 7: File Systems in Linux

What we will see

- Why file systems matter
- Common Linux file systems

File systems are a choice

- Filesystem is structured way to store, organize and manage data on storage devices
- Linux supports many file systems:
  - ext4
  - XFS
  - btrfs
    Different trade-offs.

ext4

- Default on many systems
- Stable and well-tested
- General-purpose

XFS and btrfs

- XFS: high-performance, large files
- btrfs: snapshots, checksums, modern features

Linux lets you choose.

Why this matters

- Performance
- Reliability
- Features
- Use-case alignment

Storage is a design decision.

Section 7: Summary

- Linux separates kernel from storage
- Multiple file systems coexist
- Choice enables optimization

Any questions?

Section 8: Kernel Modules and Extensibility

What we will see

- How Linux extends itself
- Why modules matter

What are kernel modules?

- Pieces of kernel code
- Loaded when needed
- Can be added or removed dynamically

Kernel without reboot.

What modules are used for

- Device drivers
- File systems
- Networking features

Hardware support becomes flexible.

Why modules matter

- Smaller core kernel
- Hardware independence
- Easier maintenance

Extensibility is built-in.

Section 8: Summary

- Kernel is modular
- Features are dynamic
- Linux adapts to hardware easily

Section 9: Distributions and Desktop Environments

What we will see

- Why distros exist
- Desktop vs OS

Linux distributions

- Package kernel + tools
- Set defaults and policies
- Provide updates

Examples:

- Debian
- Ubuntu

Desktop environments

A desktop environment is the graphical layer that lets humans use the OS visually.

Examples:

- GNOME
- KDE

Desktop ≠ operating system.

Section 9: Summary

- Distros package decisions
- Desktops control appearance
- Both are replaceable layers

Summary

- Linux is a kernel
- GNU completes the OS
- GPL protects freedom
- CLI and utilities are central
- Security, storage, and modules matter
- Linux is an ecosystem, not a product

Reflect

- What makes Linux different from other OSes?
- Where would you use Linux?

Thank you!

Any questions?

Thejesh GN