Universidad de Oviedo

Linux Operating System

Operating Systems - 2024/2025

Group: Team 4 – S-I-1

Teacher: Miguel Riesco Albizu

Authors:

Álvaro Puebla Ruisánchez UO299874

Sergio Quintana Fernández UO288090

Elena Quintes García UO269665

Table of Contents:

1. Introduction

1.1 Introduction to Linux

Linux is a Unix-like, open-source operating system initiated by Linus Torvalds in 1991. Based on the FOSS (Free and Open-Source Software) model, it has grown into a robust ecosystem comprising hundreds of distributions, powering everything from mobile devices to enterprise servers and supercomputers. Today, Linux reported that in 2023 had over 32 million users and rising, and it's used by major companies such as Google, NASA, Intel and even Tesla uses the operating system and their cars among others.

1.2 Historical Background and Philosophy

1.2.1 The Beginnings

Torvalds began Linux as a personal project, aiming to build a free kernel. He released the initial version of the software in 1991, inviting the collaboration of other users. This initiative evolved into one of the most influential software projects globally.

1.2.2 Collaborative Development

Linux's development model thrives on global contributions from both individuals and organizations. This collaborative nature has ensured rapid innovation, security, and adaptability. This collaborative environment is supported by the Linux Foundation, that manages projects such as Kubernetes and Node.js. A key tool in the collaboration is the use of distributed version control systems like Git.

1.2.3 Free and Open-Source Philosophy

Linux's core values include software freedom, transparency, and community-driven development. Its open nature allows for quick vulnerability fixes, extensive customization, and a strong culture of learning and support. This philosophy is followed by projects like Debian, or Ubuntu, which allow users from all kinds of backgrounds to learn, participate and innovate.

1.3 General Features of Linux

- **Multiuser Support** – Enables concurrent user sessions without compromising security.
- **Multitasking** – Efficient task management through sophisticated process scheduling.

- **Stability and Security** – Linux is known for uptime, system integrity, and robust permission models.
- **Flexibility and Modularity** – Users can tailor their systems by installing only necessary components.
- **Community Support** – Extensive forums, documentation, and collaborative tools aid learning and troubleshooting.

2. Architecture and Components of Linux

2.1 Overview of Linux Architecture

The Linux operating system is structured according to a layered architectural model that enhances modularity, maintainability, and scalability. The main architectural components are:

- **Kernel Space**: This is the privileged area of memory where the Linux kernel operates. It provides low-level hardware control and enforces process isolation and resource management. All critical OS functionalities such as memory management, process scheduling, and device driver interaction are handled here.
- **System Libraries**: These act as an interface between user applications and the kernel. System libraries provide standard APIs for system calls, enabling application-level programs to perform low-level tasks without directly accessing kernel code.
- **User Space**: This space hosts all user-level processes and applications. It includes shells, user interfaces, compilers, web browsers, and other utilities. Interactions with hardware or system resources are mediated via system calls handled through the libraries.

This separation of concerns between kernel and user space allows for secure execution, parallel development, and easier debugging or updates, contributing to system stability and robustness.

2.2 The Linux Kernel

The kernel is the core component of the Linux operating system. It serves as the intermediary between hardware and software, managing system resources, ensuring security, and enforcing policy. Its main responsibilities include:

- **Hardware Abstraction**: The kernel abstracts hardware complexity and presents a uniform interface to applications. This allows the same software to run on a wide variety of hardware platforms.
- **Process Scheduling**: The kernel employs scheduling algorithms to allocate CPU time efficiently among active processes.

- **Memory Management**: It handles physical and virtual memory allocation, paging, and swapping, ensuring isolation between processes and optimal resource usage.
- **Device Management**: Through its modular device drivers, the kernel allows communication with various hardware devices like disk drives, network cards, and USB peripherals.
- **System Calls**: It provides a system call interface enabling controlled interaction between user applications and kernel services.

Monolithic vs. Modular Kernel

Traditionally, Linux is categorized as a **monolithic kernel**, which implies that core OS functionalities (device drivers, file systems, memory management, etc.) are all executed in kernel space as part of a single binary.

However, modern Linux kernels follow a **modular design** that allows for dynamic loading and unloading of components (kernel modules) at runtime. This provides several advantages:

- **Flexibility**: Modules such as drivers or file systems can be added or removed without rebooting the system.
- **Efficiency**: Only necessary components are loaded into memory, optimizing resource usage.
- **Maintainability**: Bugs or vulnerabilities in specific modules can be patched independently.

2.3 Process Management in Linux

Linux is a **multi-user, multitasking** OS that handles process management through a combination of system calls and kernel-level services. Key aspects of process management include:

- **Process Creation**: Processes are created using the fork() system call, which duplicates the calling process. The exec() family of functions then allows the child process to replace its memory space with a new program.
- **Scheduling**: Linux uses the Completely Fair Scheduler (CFS), which aims to allocate CPU time fairly across all processes while considering their priorities and responsiveness.
- **Process Termination**: A process can terminate normally via exit() or abnormally due to signals or faults. The kernel ensures cleanup and notifies the parent process using the wait() system call.
- **Interprocess Communication (IPC)**: Linux supports various IPC mechanisms for coordination and data exchange between processes, including:
    - **Pipes** and **named pipes (FIFOs)**

- **Shared memory**
- **Message queues**
- **Semaphores**
- **Signals**

Together, these mechanisms provide a powerful and flexible model for managing concurrent execution and resource sharing.

2.4 File Systems in Linux

The Linux file system architecture is hierarchical, with a single root directory (/) at the base. All other directories and storage devices are mounted within a tree structure. Key features include:

- **Support for Multiple File Systems**: Linux supports a wide variety of file systems, including:
  - **ext4**: The default file system in many distributions. It offers robustness, journaling, and good performance for general-purpose use.
  - **XFS**: A high-performance file system optimized for scalability, particularly in enterprise environments with large files and directories.
  - **Btrfs**: A modern file system with advanced features such as snapshots, compression, deduplication, and checksums for data integrity.
- **Mounting and Unmounting**: File systems are integrated into the system via the mount command and detached using umount. These operations map file systems to directories within the overall hierarchy.
- **File System Checking and Repair**: Utilities such as fsck (file system check) are available to inspect and repair inconsistencies or corruption.
- **Virtual File Systems**: Linux includes special-purpose pseudo file systems like:
  - **/proc**: Provides real-time process and kernel information.
  - **/sys**: Exposes kernel device structures.
  - **/dev**: Contains device files used to interact with hardware.

This flexible and extensible approach to file system management allows Linux to serve a broad range of use cases, from embedded systems to enterprise servers.

3. Administration, Security, and Comparison with Other Operating Systems

3.1 User Management and Permissions

Linux handles users via tools like **useradd**, **usermod**, and **passwd**. File access is controlled with permissions (read, write, execute) and commands like **chmod** and **chown**. ACLs (Access Control Lists) are used in order to assign specific permissions to individual users or groups of users. This is done by using tools such as **setfacl** and **getfacl.**

3.2 Software and Package Management

Linux simplifies software management with package managers like APT (Debian/Ubuntu), YUM/DNF (Red Hat-based), and Pacman (Arch Linux). These tools facilitate the installation, updating, and removal of software packages by resolving dependencies and fetching them from centralized repositories, ensuring the system's consistency and reliability.

3.3 Security in Linux

Security is a core feature of Linux, with mechanisms like SELinux for mandatory access control, AppArmor for program isolation, and iptables/nftables for network security.

Linux's user and process isolation limits the impact of compromises. Compared to Windows, Linux benefits from faster vulnerability patches and superior permission systems, while its flexibility in security features makes it more suitable for servers and embedded systems than macOS.

3.4 Comparative Analysis with Other Operating Systems

Linux excels in performance, stability, and customization. Its reliability for long uptimes makes it ideal for servers, while user-friendly distributions like Ubuntu ease adoption. Linux dominates server markets, offers customizable desktop environments, and powers embedded systems and IoT devices due to its modularity and lightweight nature.

These features can be represented in a table:

Feature	Linux	Windows	macOS
Security	Strong, transparent	Closed-source, slower patches	Secure, but limited flexibility
Customization	High	Low	Medium
Stability	Excellent (ideal for servers)	Moderate	Very stable
Cost	Free	License required	Bundled with Apple hardware

4. Advanced Topics and Emerging Trends

4.1 Virtualization

Linux KVM (Kernel-based Virtual Machine) is the most commonly used virtualization driver in **OpenStack**, thanks to its tight kernel integration and modular loadable drivers that deliver near-bare-metal performance. Under the hood, KVM's two core modules (kvm.ko plus either kvm-intel.ko or kvm-amd.ko) expose a /dev/kvm interface that user-space tools (notably **QEMU**) invoke to launch and manage VMs.

Enterprises pair KVM with QEMU to support a wide variety of VM image formats such as raw, qcow2, and VMware's VMDK, giving flexibility for backups, snapshots, and conversions across environments. QEMU also handles virtual device emulation (network, disk, graphics), while **libvirt** provides a standardized API and tooling (virsh, virt-manager) to define, monitor, and migrate instances.

Live migration is built into this stack: by default, block live migration transparently copies disk state over the network, and with shared storage or volume-backed setups, only memory and CPU state are transferred, minimizing downtime. For stronger confidentiality and integrity, "QEMU-native TLS" can be enabled to encrypt disk traffic during migration without the overhead of libvirt tunneling.

4.2 Containerization & Orchestration

Containers have crossed the chasm: container use (including both piloting and production deployments) exceeds 90 % of surveyed organizations, making Linux containers the foundation of modern application delivery. Docker remains the de facto container runtime in many environments, but purpose-built alternatives are on the rise: containerd, graduated in CNCF in 2019, and CRI-O, an OCI-focused runtime, are both now key components of Kubernetes installations.

Kubernetes itself has reached mass adoption: in 2023, 66 % of end-user organizations ran Kubernetes in production and another 18 % were actively evaluating it (84 % total),

with only 15 % of respondents reporting no plans to adopt the platform. This momentum continues: according to the 2024 “Voice of Kubernetes Experts” report, the top priorities for platform engineers are improving autoscaling, enhancing security posture, and producing edge-optimized distributions to extend Kubernetes beyond the data center.

Meanwhile, the ecosystem around orchestration is diversifying: lighter, single-binary Kubernetes variants such as K3s are tailored for edge use cases, and micro-VM projects like Firecracker and Kata Containers blend container agility with VM isolation. Together, these trends underscore Linux’s versatility in supporting everything from development sandboxes to hyper-scale production clusters.

4.3 Cloud Native & Distributed Systems

Linux is the de facto foundation for modern cloud-native architectures: over 65 % of Azure VM instances run Linux images, and more than 60 % of Azure VM cores are Linux-based—thanks to its lightweight footprint, rapid boot times, and built-in support for cloud-init and Azure Agent tooling .

On AWS, the Amazon Linux AMI is provided at no additional charge; it’s tuned for EC2 performance, integrates seamlessly with CloudInit, AWS Systems Manager, and offers optimized container support (Docker, ECS agent) out of the box .

For private and hybrid clouds, OpenStack remains a leading choice—its managed-service market is projected to exceed USD 20 billion in 2024—leveraging Linux hosts, KVM as the default hypervisor, Ceph for software-defined storage, and Neutron for scalable software-defined networking .

At the edge, ultra-light Linux variants (e.g. Alpine Linux, Ubuntu Core) paired with K3s and micro-VM frameworks (Firecracker, Kata Containers) enable telecom gateways, IoT controllers, and industrial routers to run containerized workloads with minimal overhead and strong isolation .

4.4 The Role of Linux in Cybersecurity

Linux’s security model combines mandatory access control, intrusion detection, and live-patching to deliver enterprise-grade hardening:

- **Mandatory Access Control:** SELinux and AppArmor are embedded in RHEL, Ubuntu, Debian, and SUSE to enforce fine-grained policy confinement, isolating services and reducing attack surfaces .
- **Intrusion Detection & Monitoring:** Security Onion—bundling Suricata for IDS, Zeek for network analysis, and the Elastic Stack for log aggregation—runs on Linux appliances to provide scalable, real-time threat detection and investigation .

- **Penetration-Testing:** Kali Linux, with over 600 preinstalled tools (Metasploit, Nmap, Wireshark, etc.), remains the go-to distro for red-team assessments and forensic analysis on Linux platforms .
- **Live Kernel Patching & LTS Kernels:** Solutions like Kpatch (Red Hat) and kGraft (SUSE) apply critical security fixes to running kernels without rebooting, while extended LTS kernels from Ubuntu and Debian ensure long-term stability and backported security updates .

These technologies together make Linux not only the engine of cloud-native and distributed systems, but also one of the most secure and resilient operating systems in production today.

5. Case Studies and Real-World Implementations

5.1 High-Performance Computing

As of the 2024 TOP500 list, 100 % of the world's 500 fastest supercomputers run Linux-based Operative Systems. This uniformity reflects Linux's scalability, performance tuning, and open-source collaboration across HPC centers.

5.2 Global Streaming at Netflix

Netflix runs "hundreds of functions" on AWS, totaling over 100,000 server instances to handle transcoding, analytics, and recommendation engines, all on Linux-based EC2 nodes. These instances predominantly use the Amazon Linux AMI for a stable, AWS-integrated environment, enabling rapid feature rollout and security updates at scale.

5.3 Automotive Infotainment in Tesla

Tesla's in-car Media Control Units run a modified Ubuntu-based Linux kernel on NVIDIA Tegra processors, powering touchscreen UI, media playback, and over-the-air updates. Tesla even publishes much of this kernel source on GitHub under GPLv2, showcasing compliance with open-source licensing and enabling community scrutiny.

5.4 Embedded & IoT Devices

Embedded Linux leads IoT development, with 62 % of embedded systems and 82 % of smartphones (via Android) based on the Linux kernel. From network routers to smart sensors, Linux's modular build systems (Yocto, Buildroot) and real-time patches (PREEMPT_RT) address resource constraints while ensuring security and maintainability.

6. Conclusions

The Linux Operating System exemplifies the power of open-source collaboration and modular design. Its layered architecture, composed of kernel space, system libraries, and user space, facilitates maintainability, scalability, and robust security. The Linux kernel, while traditionally monolithic, has evolved into a highly modular and configurable core capable of adapting to a wide range of use cases, from embedded systems to cloud-native infrastructures.

Linux's efficient process management, versatile file systems, and powerful user and package administration mechanisms make it a reliable and high-performing alternative to proprietary operating systems. Furthermore, its integration into virtualization, container orchestration, and cybersecurity paradigms positions it as a critical enabler of current and future computing paradigms.

Beyond technical strengths, Linux's greatest asset is its global community. This dynamic ecosystem drives innovation, accelerates patching of vulnerabilities, and fosters an inclusive environment for learning and development. Despite challenges such as a steeper learning curve or occasional hardware compatibility issues, Linux continues to set the standard for flexibility, transparency, and sustainability in operating systems.

7. Glossary

- **ACL (Access Control List):** Extension of Unix permissions allowing fine-grained user/group access control on files and directories.
- **Kernel:** Core of the OS managing CPU, memory, and device drivers in privileged “kernel space.”
- **Kernel Module:** Dynamically loadable driver or feature that can be inserted/removed from the kernel at runtime.
- **Fork → Exec:** fork() clones a process; exec() replaces its memory with a new program.
- **CFS (Completely Fair Scheduler):** Default Linux scheduler that distributes CPU time evenly among processes.
- **Ext4:** The common Linux filesystem offering journaling and large-volume support.
- **Namespace:** Kernel feature isolating resources (PID, network, mounts) to containers or processes.
- **Container:** Lightweight runtime packaging an application and its dependencies, sharing the host kernel (e.g., Docker).
- **KVM (Kernel-based Virtual Machine):** Hardware-assisted virtualization module exposing /dev/kvm for VMs.
- **SELinux / AppArmor:** Mandatory Access Control systems enforcing security policies by confining processes.
- **QEMU:** Emulator and virtualizer that, with KVM, runs full-system VMs.

- **fsck:** Utility to check and repair filesystem integrity.
- **System Call:** A controlled entry point that allows user-space applications to request services from the kernel.
- **Monolithic Kernel:** A kernel architecture where all operating system services run in kernel space as a single large process.
- **Process Scheduling:** The method by which the OS allocates CPU time to various processes, ensuring efficient multitasking.
- **File System:** A structured method for storing and organizing files on storage devices, supporting journaling and metadata.

8. Bibliography

The following references have been used to compile this review, adhering to the UNE-ISO 690 standards for bibliographic citation:

- Torvalds, L. (1991). *Linux Kernel Source Code*. Available at: https://www.kernel.org
- Love, R. (2010). *Linux Kernel Development*. Addison-Wesley.
- Nemeth, E., Snyder, G., Hein, T. R., & Whaley, B. (2017). *Linux Administration Handbook* (6th ed.). Prentice Hall.
- The Linux Foundation. (2020). *Introduction to Linux*. Available at: https://www.linuxfoundation.org
- Additional online resources, community forums, and official documentation from various Linux distributions have also contributed to the insights provided in this review.
- ANDREW S. TANENBAUM. *Modern Operating Systems*. 3rd edition. 2007. Upper Saddle River, NJ, USA. Prentice Hall Press. ISBN: 978-0136156734
- WIRED. *Linux Took Over the Web. Now It's Taking Over the World* [online]. 2016. [Last visited: 2025-4-20]. Available at: https://www.wired.com/2016/08/linux-took-web-now-taking-world/
- WIKIPEDIA. *History of Linux* [online]. [Last visited: 2025-4-20]. Available at: https://en.wikipedia.org/wiki/History_of_Linux
- 99FIRMS. *Linux Statistics* [online]. [Last visited: 2025-4-20]. Available at: https://99firms.com/blog/linux-statistics/#gref
- THE LINUX FOUNDATION. *The Linux Foundation Releases First-Ever Value of Collaborative Development Report* [online]. [Last visited: 2025-4-20]. Available at: https://www.linuxfoundation.org/press/press-release/the-linux-foundation-releases-first-ever-value-of-collaborative-development-report
- DEBIAN. *Debian Philosophy* [online]. [Last visited: 2025-4-20]. Available at: https://www.debian.org/intro/philosophy.html
- UBUNTU. *Mission | Ubuntu* [online]. [Last visited: 2025-4-20]. Available at: https://ubuntu.com/community/ethos/mission#:~:text=To%20bring%20free%20software%20to,ambitions%20regardless%20of%20their%20resources.

- RED HAT. *Linux Access Control Lists* [online]. [Last visited: 2025-4-20]. Available at: https://www.redhat.com/es/blog/linux-access-control-lists
- STORWARE. *OpenStack is More Than Virtualization. So What?* [online]. [Last visited: 2025-4-20]. Available at: https://storware.eu/blog/openstack-is-more-than-virtualization-so-what/
- MICROSOFT TECH COMMUNITY. *How Microsoft Ensures the Quality of Linux VM Images and Platform Experiences* [online]. [Last visited: 2025-4-20]. Available at: https://techcommunity.microsoft.com/blog/linuxandopensourceblog/how-microsoft-ensures-the-quality-of-linux-vm-images-and-platform-experiences-on/4303513
- MICROSOFT AZURE. *Linux Virtual Machines | Azure* [online]. [Last visited: 2025-4-20]. Available at: https://azure.microsoft.com/en-us/products/virtual-machines/linux
- KALI LINUX. *Official Website* [online]. [Last visited: 2025-4-20]. Available at: https://www.kali.org/
- LINUXBLOG.IO. *Securing Linux with SELinux and AppArmor* [online]. [Last visited: 2025-4-20]. Available at: https://linuxblog.io/securing-linux-selinux-apparmor
- RED HAT. *Top 10 Edge Articles of 2024* [online]. [Last visited: 2025-4-20]. Available at: https://www.redhat.com/en/blog/top-10-edge-articles-2024
- QEMU. *Official QEMU Website* [online]. [Last visited: 2025-4-20]. Available at: https://www.qemu.org/
- OPENSTACK. *OpenStack Documentation* [online]. [Last visited: 2025-4-20]. Available at: https://docs.openstack.org
- CLOUD NATIVE COMPUTING FOUNDATION (CNCF). *Reports* [online]. [Last visited: 2025-4-20]. Available at: https://www.cncf.io/reports/
- AMAZON WEB SERVICES (AWS). *Case Studies* [online]. [Last visited: 2025-4-20]. Available at: https://aws.amazon.com/es/solutions/case-studies/
- ELECTREK. *Tesla's Updated Browser Will Be Linux-Based by December, Says Musk* [online]. 2016. [Last visited: 2025-4-20]. Available at: https://electrek.co/2016/10/06/tesla-updated-browser-linux-os-december-says-musk
- NODE.JS. Introduction to Node.js [online]. s.f. [Last visited: 2025-4-20]. Available at: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs