



Operating Systems

Lecture 1.0

Overview of Operating Systems
Discussion on Course Matrix

Lecture Agenda



- Course Information and Protocol
- What is an Operating System
- Operating System: An Abstraction
- Operating System & Hardware Interaction
- Major Operating System Families
- Evolution of Operating System
- Operating System Architecture Models
- User Space vs Kernel Space
- System Calls, Signals, H/W Interrupts & Traps
- **Why Linux?**
- Discussion on Course Matrix

Course Info & Protocol

About The Instructor



Dr. Muhammad Arif Butt
Asst. Prof. at Department of Data Science
University of Punjab, Lahore

arif@pucit.edu.pk
<https://arifbutt.me>
<https://youtube.com/learnwitharif>
<https://github.com/arifpucit>
<https://www.linkedin.com/in/dr-arif-butt/>

Instructor: Muhammad Arif Butt, PhD

- **Education:**

- Graduation from Pakistan Military Academy, Kakul
- MPhil CS from University of Punjab, Lahore
- PhD CS from University of Punjab, Lahore

- **Experience:**

- Served in field/staff/instructional posts in Pakistan Army
- Assistant Professor, Department of Data Science
- Founder Excaliat (<https://excaliat.com/en>)
- Founder FalconHunt (<https://falconhunt.org/>)
- Co-Founder Tbox Solutionz (<https://tboxsolutionz.com/>)

- **Research Interest:**

- Embedded and Real-Time Operating Systems
- Vulnerability Analysis, Binary Exploitation & Exploit Development
- AI-Driven Cybersecurity and Securing AI systems

Course Information



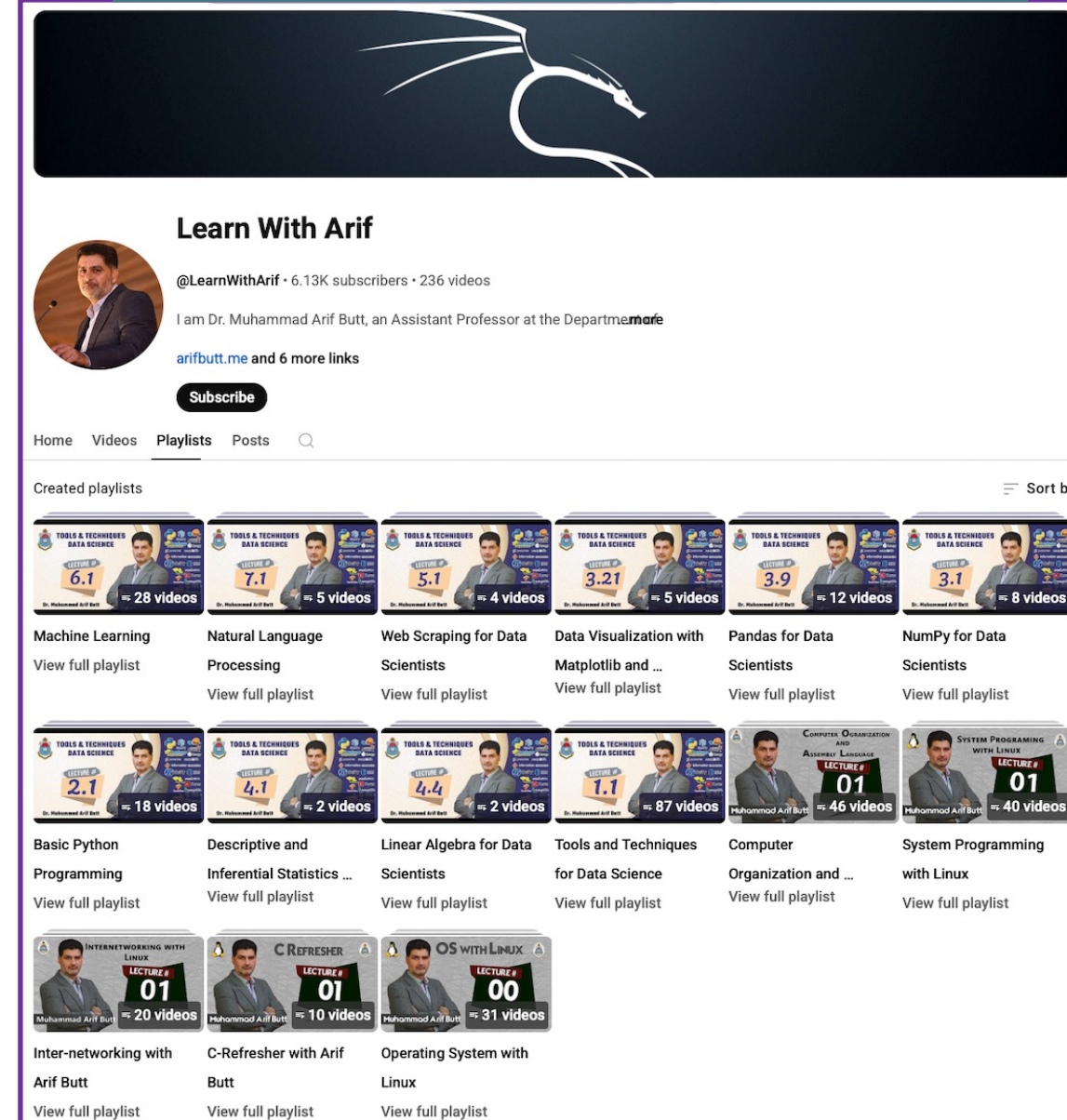
- **Lectures Slides/Handouts** Available at: <https://arifbutt.me>
- **Video Lectures** Available at: <https://youtube.com/learnwitharif>
- **Codes Hosted** at: <https://github.com/arifpucit>
- **Grades Website:** <https://online.pucit.edu.pk>
- **Prerequisites:** Basic programming skills in C, and Assembly
- **Office:** Building-C, FCIT (NC)
- **Students Counseling hours:**
 - Mon: 09:00 am – 10:00 am
 - Tues: 11:30 am – 12:30 pm
- 24 hour turnaround for email: arif@pucit.edu.pk

OS with Linux is a 30 video sessions course that covers working with Linux Command Lines covering the core concepts discussed in the Operating System course.

System Programming with Linux is a 40 video sessions course that covers core OS concepts while practically implementing them via C programming interface.

Instructor: Muhammad Arif Butt, PhD

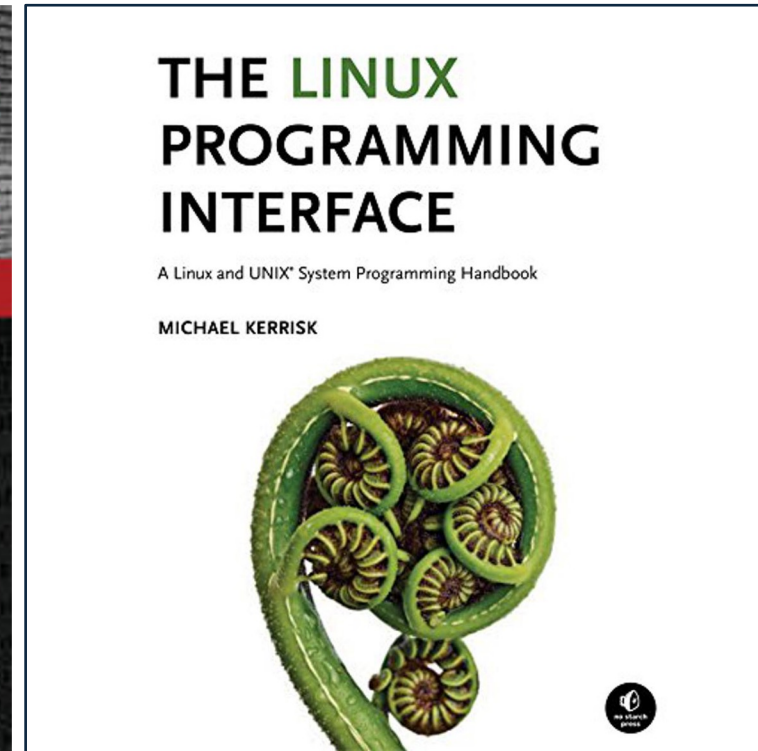
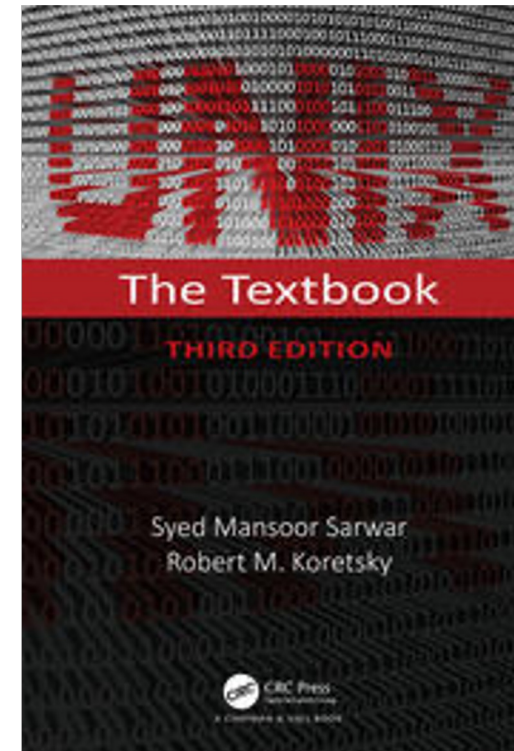
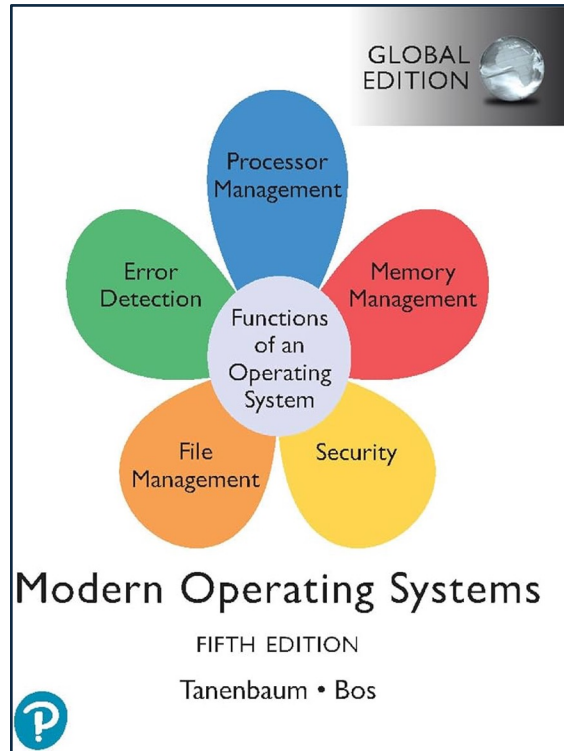
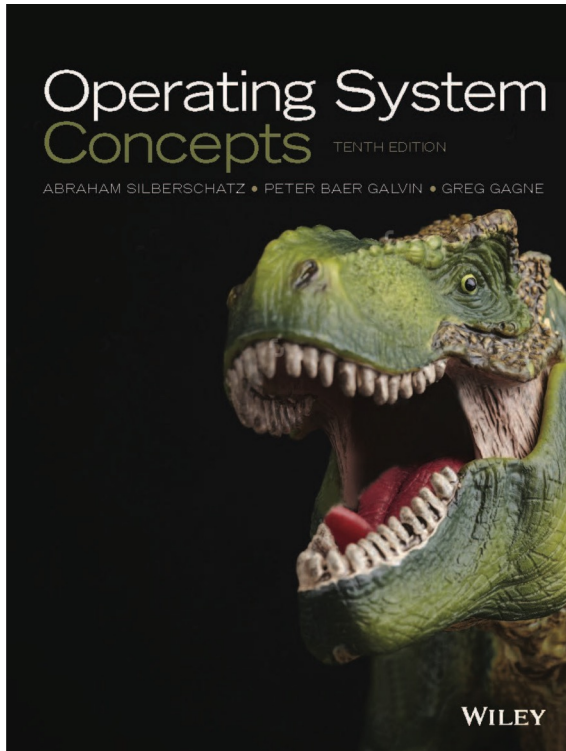
<https://www.youtube.com/LearnWithArif/playlists>



The image shows a screenshot of the YouTube channel page for 'Learn With Arif'. The channel has 6.13K subscribers and 236 videos. The profile picture shows a man in a suit. The bio states: 'I am Dr. Muhammad Arif Butt, an Assistant Professor at the Department of Computer Science, FCIT (NC), Poonch University. arifbutt.me and 6 more links'. There is a 'Subscribe' button. Below the bio, there are tabs for 'Home', 'Videos', 'Playlists', and 'Posts'. The 'Playlists' tab is selected. Under 'Created playlists', there are 18 playlists displayed in a grid. Each playlist has a thumbnail with a lecture number and a video count. The playlists are: Machine Learning (28 videos), Natural Language Processing (5 videos), Web Scraping for Data Scientists (4 videos), Data Visualization with Matplotlib and ... (5 videos), Pandas for Data Scientists (12 videos), NumPy for Data Scientists (8 videos), Basic Python Programming (18 videos), Descriptive and Inferential Statistics ... (2 videos), Linear Algebra for Data Scientists (2 videos), Tools and Techniques for Data Science (87 videos), Computer Organization and Assembly Language (46 videos), System Programming with Linux (40 videos), Inter-networking with Arif Butt (20 videos), C-Refresher with Arif Butt (10 videos), and Operating System with Linux (31 videos).

Playlist Name	Video Count
Machine Learning	28 videos
Natural Language Processing	5 videos
Web Scraping for Data Scientists	4 videos
Data Visualization with Matplotlib and ...	5 videos
Pandas for Data Scientists	12 videos
NumPy for Data Scientists	8 videos
Basic Python Programming	18 videos
Descriptive and Inferential Statistics ...	2 videos
Linear Algebra for Data Scientists	2 videos
Tools and Techniques for Data Science	87 videos
Computer Organization and Assembly Language	46 videos
System Programming with Linux	40 videos
Inter-networking with Arif Butt	20 videos
C-Refresher with Arif Butt	10 videos
Operating System with Linux	31 videos

Text / Reference & Books



Who cares to get A



- Final-Term Exam: 40
- Mid-Term Exam: 35
- Sessional Activities: 25
 - Assignments: 30%
 - Quizzes: 50%
 - Class Project: 20%



Lecture Format



Late submission guidelines protocol



- Late Assignment submissions will not be accepted!
- There will be no retake on quizzes!



Start working on your tasks early and submit well before time.

Cheating policy



- Academic integrity
- Copying an Assignment with or without permission. Both the cheater and the student who aided the cheater will be held responsible for the cheating
- The instructor may take actions such as:
 - require repetition of the subject work,
 - assign **'zero'** or may be **'negative'** marks for the subject work,
 - for serious offenses, assign an **F** grade for the course



What is an Operating System?

What is an Operating System?



A **system software** that **manages** computer **hardware** and **software resources**, and provides common services for the programs.

What is an Operating System? (cont...)



- **Piece of code** that:
 - Sits between programs & hardware
 - Sits between different programs
 - Sits between different users
- A **resource manager** that multiplex (share) resources among processes:
 - Time Multiplexed (CPU sharing, Printer sharing)
 - Space Multiplexed (Memory, Hard Disk)

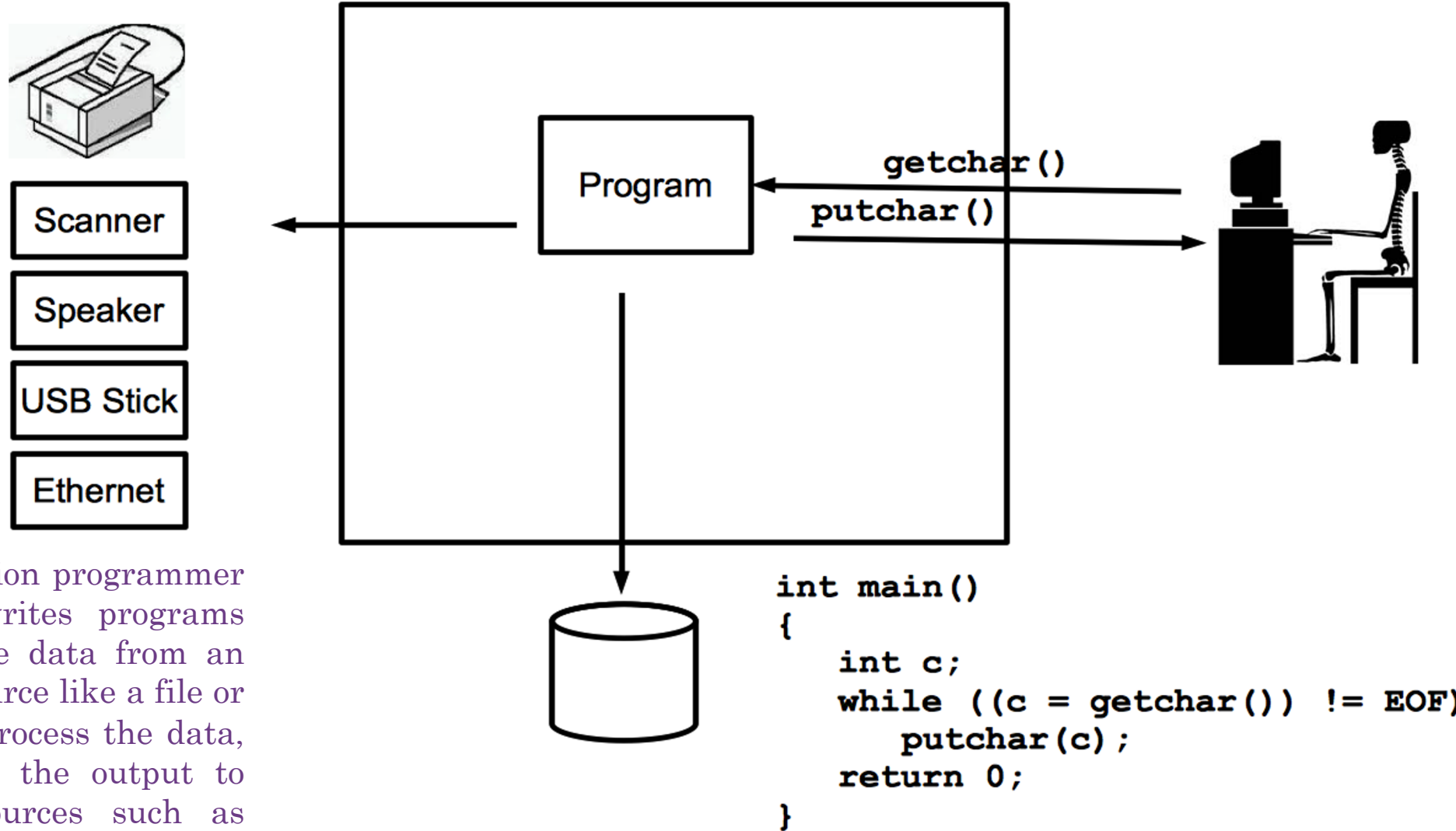
Operating System An Abstraction

Operating System - An Abstraction



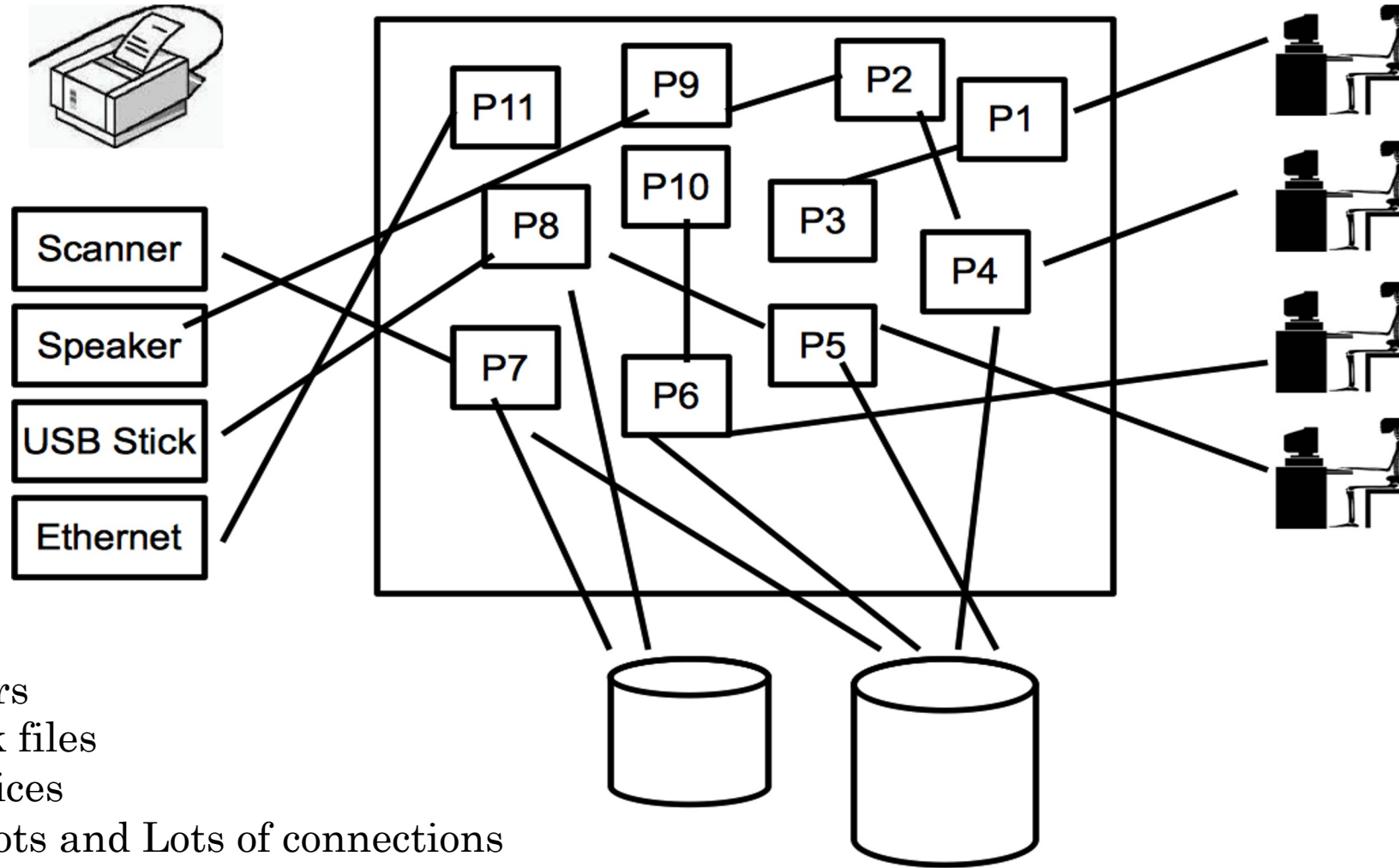
An **operating system (OS)** acts as an **abstraction layer**, hiding the complexities of the underlying **hardware** and presenting a simplified, consistent interface to applications and users

Operating System - An Abstraction



An application programmer normally writes programs that acquire data from an external source like a file or keyboard, process the data, and deliver the output to external sources such as files or the console

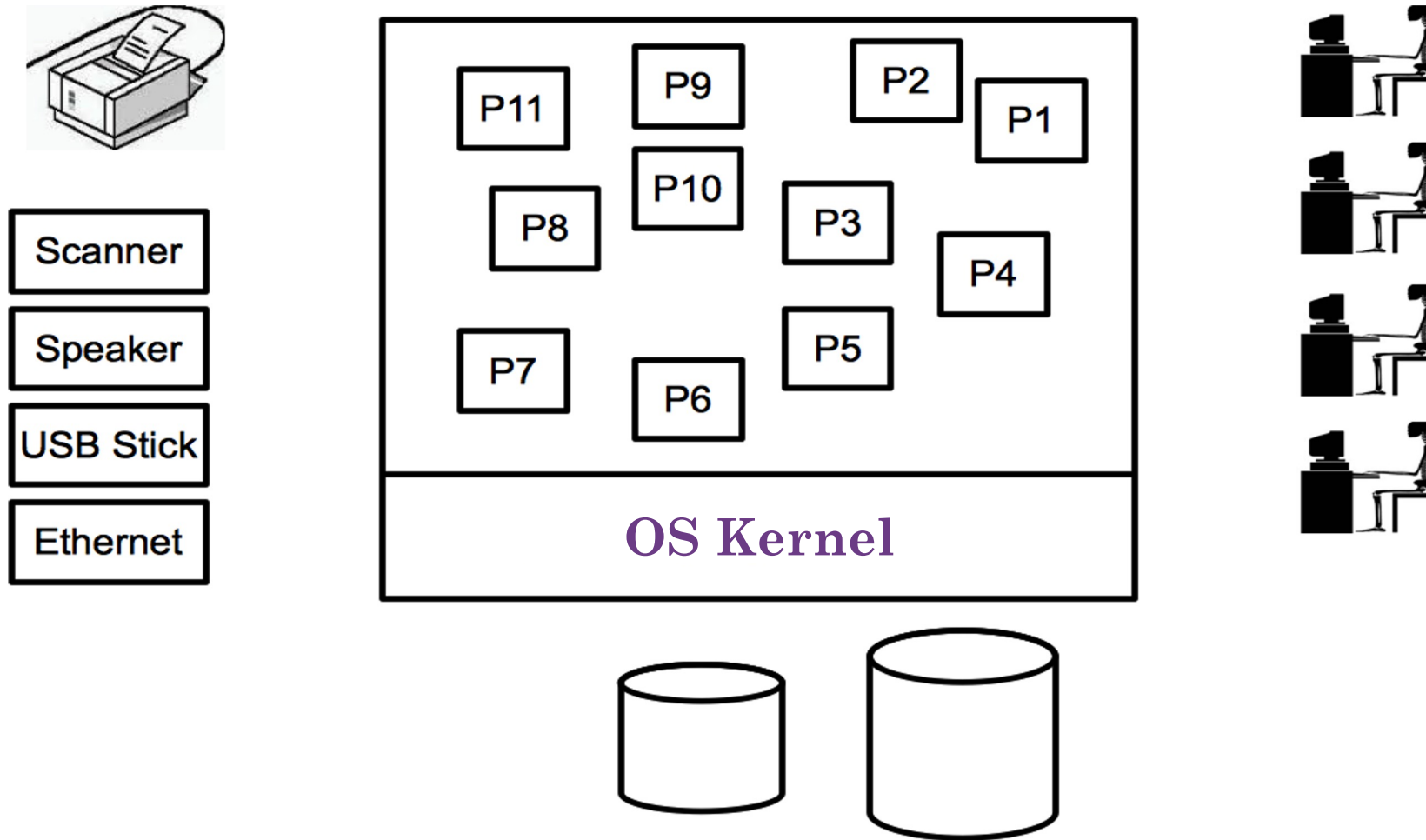
Operating System - An Abstraction



- Lots of users
- Lots of disk files
- Lots of devices
- Lots and Lots and Lots of connections

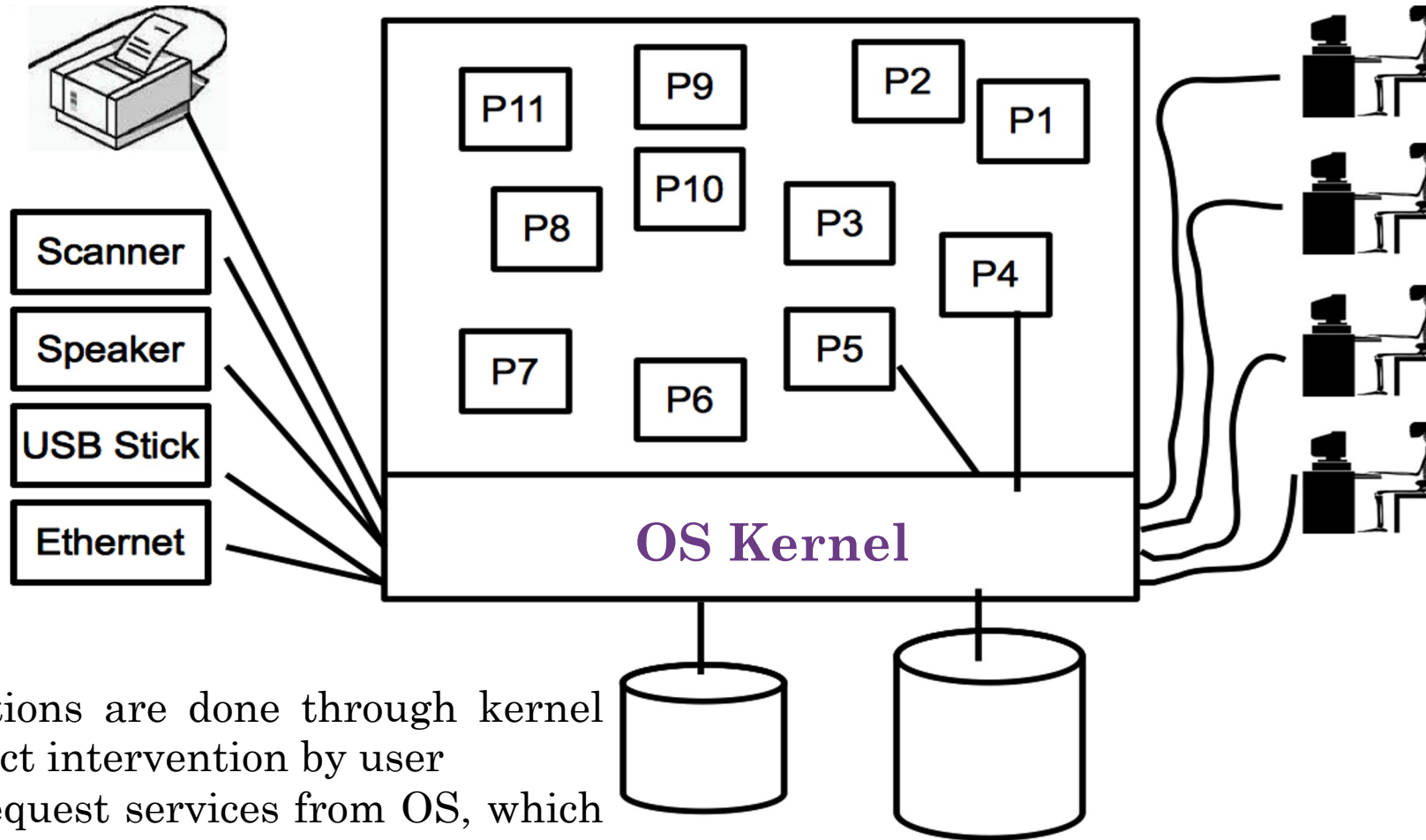
Who is managing all these resources and connect various devices to correct programs?

Operating System - An Abstraction



Role of OS is to manage all these resources and to connect various devices to the correct programs

Operating System - An Abstraction



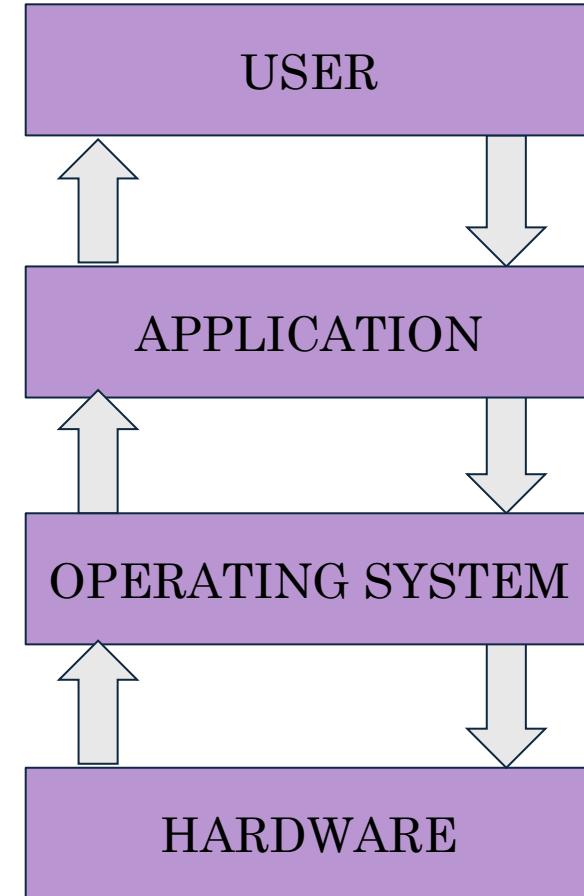
- All connections are done through kernel and no direct intervention by user
- Program request services from OS, which contains code to provide services
- An operating system provides these services via its API (e.g syscall)

How OS Provides Abstraction?



The OS acts as a crucial intermediary between a computer's hardware and software, as well as the users. It provides a platform for applications to run and interact with the computer's hardware components, enabling seamless interaction and resource management.

- **Hardware Abstraction:** The OS hides hardware complexities from applications, providing simplified interfaces to CPU, memory, and storage.
- **Process Abstraction:** The OS creates the illusion that each program has dedicated CPU and resources, even on single-CPU systems.
- **File System Abstraction:** The OS provides uniform file access regardless of underlying storage type (HDD, SSD, etc.).
- **API Abstraction:** The OS offers standardized APIs for applications to access hardware without knowing implementation details.
- **User Interface:** The OS provides user interfaces (UI) for system interaction, including command-line interfaces (CLI), graphical user interfaces (GUI), and natural language interfaces.



Key Functions of Operating System



- **Resource and I/O Management:** The OS allocates hardware resources (CPU, memory, disk) to applications for efficient use and coordinates communication with peripherals (e.g., keyboards, mice, printers) via device drivers for standardized, reliable data transfer.
- **File Management:** The OS organizes, stores, retrieves, and secures files on storage media, managing directories, naming, permissions, and data integrity.
- **Process Management:** The OS manages the execution of programs (processes) by overseeing their entire life cycle (creation, scheduling, execution, and termination). It optimizes CPU utilization, coordinates access to system resources, and enables multitasking to ensure efficient and responsive system performance.
- **Inter Process Communication and Networking:** The OS enables data exchange between processes on the same or different systems using pipes, message queues, shared memory, sockets, and network protocols for secure, efficient communication.
- **Synchronization:** The OS ensures orderly execution of concurrent processes and threads using mutexes, condition variables and semaphores to and prevents, detects, and resolves deadlocks for smooth operation.
- **Memory Management:** The OS manages physical RAM and virtual memory, allocating space with paging and swapping while ensuring efficiency, isolation, and controlled access to avoid conflicts.
- **Security:** The OS enforces security through access control, authentication, encryption, and protection mechanisms against unauthorized access and malware.

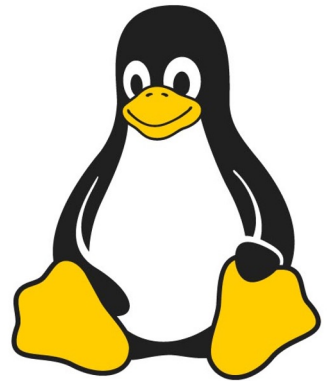
Major Operating System Families

Linux Operating System

Linux is a free and open-source operating system that was first released by Linus Torvalds in 1991. It is known for its stability, flexibility, security, and wide use in servers, embedded systems, cloud infrastructure, and desktops. Unlike Windows, Linux is based on the Unix architecture and is developed collaboratively by thousands of developers worldwide.

Evolution of Linux OS Versions (Key Milestones)

- **1991 – Linux 0.01:** First public release by Linus Torvalds.
- **1994 – Linux 1.0:** First stable kernel; supported only a few hardware types.
- **1996 – Linux 2.0:** Support for multiprocessing (SMP); growth in server adoption.
- **2003 – Linux 2.6:** Major improvements in performance, scalability, and hardware support.
- **2011 – Linux 3.x Series:** Improved power management and faster boot times.
- **2015 – Linux 4.x Series:** Enhanced support for ARM devices and modern hardware.
- **2020 – Linux 5.x Series:** Security updates, new file systems, and cloud-native features.
- **2024 – Linux 6.x Series:** Optimized for AI workloads, better file system performance, and extended driver support.



Mac Operating System (MacOS)



MacOS is a Unix-based operating system developed by Apple Inc. for its line of Macintosh computers. Known for its sleek user interface, strong security model, and tight integration with Apple hardware, macOS is popular among creative professionals, developers, and general users. It is built on Darwin (a BSD-based foundation) and offers stability, performance, and consistent design across updates.

Evolution of macOS Versions (Key Milestones)

- **2003 – Mac OS X 10.3 “Panther”**: Introduced Exposé window management, faster Finder, and Bluetooth support
- **2005 – Mac OS X 10.4 “Tiger”**: Spotlight search, Dashboard widgets, Automator, enhanced search tools
- **2007 – Mac OS X 10.5 “Leopard”**: Time Machine backups, Spaces virtual desktops, Boot Camp support, Quick Look
- **2011 – Mac OS X 10.7 “Lion”**: Mission Control combining Exposé, Launchpad, full-screen apps, AirDrop
- **2013 – OS X 10.9 “Mavericks”**: Free upgrade, improved Finder, App Nap, Maps, iBooks, better multi-display support
- **2016 – macOS 10.12 “Sierra”**: Rebranded to “macOS”, introduced Siri, Apple Pay on the web
- **2019 – macOS 10.15 “Catalina”**: Introduced Sidecar, Apple Arcade, Catalyst apps, separate Music/TV/Podcasts apps
- **2020 – macOS 11 “Big Sur”**: Major interface redesign, Control Center, support for Apple Silicon, run iOS apps natively
- **2022 – macOS 13 “Ventura”**: Stage Manager multitasking, Continuity Camera, FaceTime Handoff, Passkeys
- **2023 – macOS 14 “Sonoma”**: Desktop widgets, interactive screensavers, Safari profiles
- **2024 – macOS 15 “Sequoia”**: Introduced Apple Intelligence suite (AI writing/image tools, ChatGPT), iPhone Mirroring
- **2025 – macOS 16 “Tahoe”**: New “Liquid Glass” UI, advanced Apple Intelligence tools, drops Intel-only Mac support

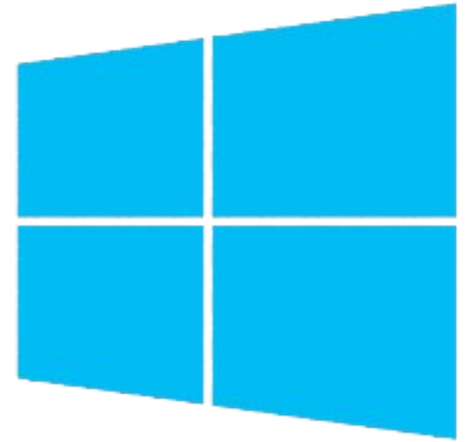
Windows Operating System



Windows is the most widely used desktop operating system, developed by Microsoft. It is known for its user-friendly graphical interface, broad software compatibility, and widespread use in both personal and enterprise environments. Since its first release in 1985, Windows has evolved alongside computing technology, shaping the modern computing experience.

Evolution of Windows OS Versions (Key Milestones)

- **1985 – Windows 1.0:** The first graphical shell on top of MS-DOS.
- **1987 – Windows 2.0:** Improved graphics, overlapping windows, Microsoft Office apps.
- **1990 – Windows 3.0:** Introduced Program Manager and better multitasking.
- **1995 – Windows 95:** First to introduce the Start menu and taskbar.
- **1998 – Windows 98:** Enhanced hardware support and internet integration.
- **2000 – Windows ME (Millennium Edition):** Aimed at home users; multimedia enhancements but buggy.
- **2001 – Windows XP:** Major redesign with stability and ease-of-use; highly popular and long-lived.
- **2007 – Windows Vista:** Introduced Aero UI and UAC (User Account Control); faced performance criticism.
- **2009 – Windows 7:** Improved performance, interface, and security; widely adopted.
- **2012 – Windows 8:** Focused on touch-based UI (Metro); removed traditional Start menu.
- **2015 – Windows 10:** Brought back the Start menu, introduced Cortana, frequent feature updates.
- **2021 – Windows 11:** Modern design, centered taskbar, support for Android apps, improved productivity features.
- **2024 – Windows 11H2:** HDR background, energy saver, Sudo for Windows, Rust in kernel, Wi-Fi 7



Android Operating System

Android is an open-source mobile operating system developed by **Google**, based on the **Linux kernel**. First released in **2008**, Android powers the majority of the world's smartphones and tablets. It is known for its customizability, flexibility, and wide hardware compatibility, used by major manufacturers like Samsung, Xiaomi, OnePlus, and Google itself.

Evolution of macOS Versions (Key Milestones)

- **2008 – Android 1.0:** First commercial release with basic apps: Gmail, Maps, and Market.
- **2009 – Android 1.5:** First version with an on-screen keyboard; introduced widgets.
- **2010 – Android 2.1–2.3:** Improved UI, tethering, and performance.
- **2012 – Android 4.1–4.3:** Project Butter for smoother UI, Google Now.
- **2013 – Android 4.4:** Improved memory management; "OK Google" support.
- **2015 – Android 6.0:** App permissions, Doze mode for battery.
- **2016 – Android 7.0:** Multi-window support, improved notifications.
- **2018 – Android 9:** Gesture navigation, digital wellbeing tools.
- **2019 – Android 10:** Dropped dessert names; system-wide dark mode.
- **2020 – Android 11:** Conversation bubbles, improved privacy controls.
- **2023 – Android 14:** Battery and accessibility improvements; larger screen support.
- **2024 – Android 15:** Enhanced satellite support, dynamic performance scaling.



iPhone Operating System (iOS)



iOS is a mobile operating system developed by **Apple Inc.** for its iPhone, iPad (initially), and iPod Touch devices. First released in **2007** alongside the original iPhone, iOS is known for its security, smooth performance, and tightly controlled app ecosystem. Built on the same foundation as macOS (**Darwin/Unix-based kernel**), it offers a touch-optimized user experience, excellent hardware-software integration, and regular feature updates.

Evolution of iOS Versions (Key Milestones)

- **2008 – iPhone OS 2.0:** App Store launched; allowed third-party apps for the first time.
- **2009 – iPhone OS 3.0:** Introduced copy-paste, MMS, and voice memos.
- **2016 – iOS 10:** Redesigned lock screen, smarter Siri, and revamped iMessage.
- **2017 – iOS 11:** Introduced Files app and ARKit; iPad got improved multitasking.
- **2018 – iOS 12:** Focused on performance improvements, Screen Time, Group FaceTime, and Shortcuts app.
- **2019 – iOS 13:** Introduced Dark Mode, Sign in with Apple, and iPadOS split (iPad got separate OS).
- **2020 – iOS 14:** Redesigned widgets on home screen, App Library, Picture-in-Picture for iPhone
- **2021 – iOS 15:** Enhanced FaceTime with spatial audio, Focus modes, Live Text in photos, and improved notifications.
- **2022 – iOS 16:** Customizable lock screen, improved Messages with editing/unsend, and enhanced Mail search.
- **2023 – iOS 17:** Contact posters, NameDrop, improved autocorrect, and Interactive Widgets.
- **2024 – iOS 18:** Customizable home screen, Control Center redesign, and Apple Intelligence integration.
- **2023 – iOS 17:** Contact posters, NameDrop, improved autocorrect.
- **2024 – iOS 18:** AI enhancements, deeper Siri integration, offline models.



Evolution of Operating System

Early Computing Era (1950s-1970s)

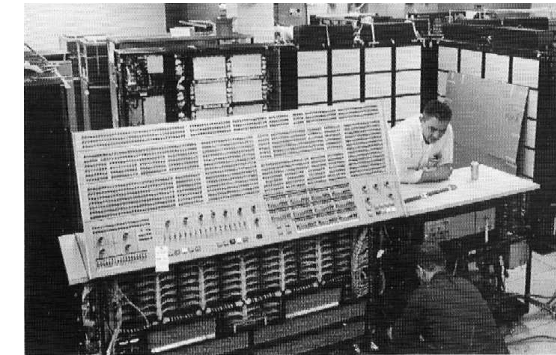
Batch Systems (1950s–1960s)

- Early operating systems that executed a series of jobs without user interaction. Users submitted punch cards or magnetic tapes, which were queued and processed in batches by the system. No direct input/output during execution.
- *Example:* IBM 7090 with early batch monitors like Fortran Monitor System(FMS).



Multiprogramming Systems (1960s–1970s)

- Introduced the concept of concurrent execution of multiple programs by making the CPU switch between jobs during I/O idle time. This dramatically increased the overall system utilization.
- *Example:* IBM OS/360 MVT.



Multi-Tasking / Time-Sharing Systems (1970s)

- Enabled multiple users to interact with the computer simultaneously via terminals. The OS would rapidly switch between users (time slicing), giving the illusion of concurrent use.
- *Example:* MULTIX (1969) and UNIX (1970).



Interactive Computing Era (1970s-2000s)

Multi-Threading Systems (1990s)

- Allow different threads of a single process to run concurrently, sharing the same address space and resources but executing independently, leading to finer-grained parallelism and efficiency in modern applications..
- *Example:* Mach (1980s), Windows NT (1993), Java (1995), LinuxThreads (1996), NPTL (2003)



Networked & Distributed Systems (2000s)

- Operating systems evolved to support interconnected environments, allowing multiple computers to work as a coordinated system. Features like resource sharing, network security, and remote file systems became common.
- *Example:* Windows Server, distributed UNIX systems.



Modern Computing Era (2010s-Present)

Cloud Operating Systems

- Offers scalability, virtualization, and remote access by using containerization and cloud orchestration frameworks.
- *Example:* OpenStack by NASA (2010), Apache CloudStack (2012).

Mobile Operating Systems

- The mobile OSs focus on touchscreen interfaces, battery efficiency, and mobile connectivity.
- *Example:* Android, iOS.

Current Trends

- **AI Integration:** Machine learning capabilities built into OS. For example, Microsoft Co-pilot integrated into Windows 11 for document creation and productivity tasks. Apple Intelligence in iOS/macOS with features like Smart Reply, text summarization, and Siri improvements. Google Assistant deeply integrated into Android OS.
- **Edge Computing:** Processing closer to data sources. For example, Autonomous vehicle systems processing sensor data locally. AWS IoT Greengrass running on local devices. Azure IoT Edge for industrial automation.
- **Cross-Platform Continuity:** Seamless experience across devices. For example, Apple's Handoff and Continuity features allowing users to start documents on Mac, edit on iPad, and present on iPhone with real-time synchronization.

Operating System Architecture Models

Operating System Architecture Models



Operating system (OS) **architecture models** define how the **different components** of an OS are **organized and interact**

Monolithic Kernel (1969)

- All core operating system services, such as process management, file systems, device drivers, and memory management, run in a single kernel space with full access to hardware. This model offers high performance but lower fault isolation.
- *Example:* Early UNIX systems.

Layered Architecture (1971)

- Organizes the OS into hierarchical layers, where each layer is built on top of lower ones and interacts only with its adjacent layers. Promotes separation of concerns, simplicity, and ease of debugging.
- *Example:* THE operating system developed by Edsger Dijkstra and early versions of MULTICS.

Microkernel (1985)

- Only the most essential functions (like inter-process communication and basic scheduling) run in kernel space. Other services (e.g., drivers, file systems) run in user space as separate processes. Offers better security and modularity, but may involve performance overhead.
- *Example:* Mach (Carnegie Mellon University, ~1985).

Operating System Architecture Models (cont...)



Hybrid Kernel (1990)

- Combines the speed of monolithic kernels with the modularity of microkernels. While most services run in kernel mode, it uses a modular design internally for better separation. Balances performance, flexibility, and stability.
- *Example:* First introduced in Windows NT (1993), and is still used by Windows 10/11 as well as by latest macOS 16 (Tahoe).

Modular Kernel (1995)

- Extends the monolithic architecture by allowing components (like device drivers or file systems) to be loaded or unloaded at runtime as modules (.ko modules). Offers customizability and ease of extension without rebooting.
- *Example:* First introduced in Linux kernel v1.2, and is still used by latest Linux kernels.

https://www.researchgate.net/publication/228738679_The_Architecture_of_a_Reliable_Operating_System

User Space vs Kernel Space

User Space vs Kernel Space



User space is where **applications** and some **device drivers** run, while kernel space is where the operating system **kernel** and its extensions reside. This separation provides memory protection and system stability by limiting user access to critical system resources.

User Space vs Kernel Space (Cont.)



User Space

- Portion of memory, where application programs and user-level processes run.
- Has limited access to system resources and hardware.
- Processes interact with the kernel via system calls.
- Faults here typically don't crash the entire system.
- Examples: Browsers, media players, compilers.

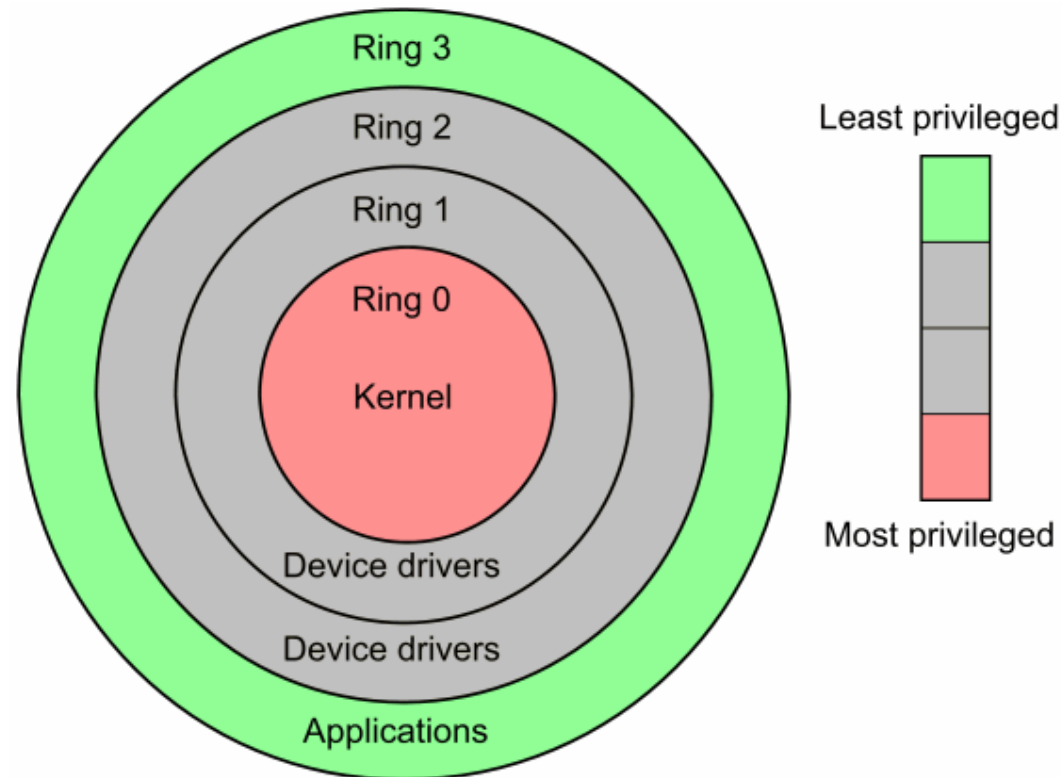
Kernel Space

- Portion of memory, where the core OS components run with full system privileges.
- Directly manages hardware resources and process scheduling.
- Faults here can lead to system crashes or kernel panics.
- Only the OS and trusted kernel modules operate here.

User Mode vs Kernel Mode



Kernel mode, also known as supervisor mode, offers unrestricted access to system resources and hardware, allowing the operating system to perform critical functions. User mode, on the other hand, restricts access to system resources and is used by applications to protect the OS from potentially harmful operations.



User Mode vs Kernel Mode (cont...)



User Mode:

- **Restricted Access:** User mode applications have limited access to system resources and cannot directly manipulate hardware or memory.
- **Application Execution:** Most applications run in user mode to prevent them from causing harm to the operating system or other applications.
- **Isolation:** User mode provides isolation between different applications, preventing one application from interfering with another.
- **Lower Risk:** Crashes in user mode are typically contained to the affected application and don't bring down the entire system.
- **System Crash:** In user mode, a system crash can be recovered by simply resuming the session.

User Mode vs Kernel Mode (cont...)



Kernel Mode:

- **Privileged Access:** Kernel mode has complete and unrestricted access to all hardware and memory.
- **Core OS Functions:** The operating system's core components, such as memory management, process scheduling, and device drivers, operate in kernel mode.
- **System Calls:** User mode applications interact with the kernel through system calls to request services.
- **High Risk:** Any error or malfunction in kernel mode can potentially crash the entire system.
- **Access:** Only essential functionality is permitted to operate in this mode.
- **Functionality:** The kernel mode can refer to any memory block in the system and can also direct the CPU for the execution of an instruction, making it a very potent and significant mode.

Switching from User to Kernel Mode

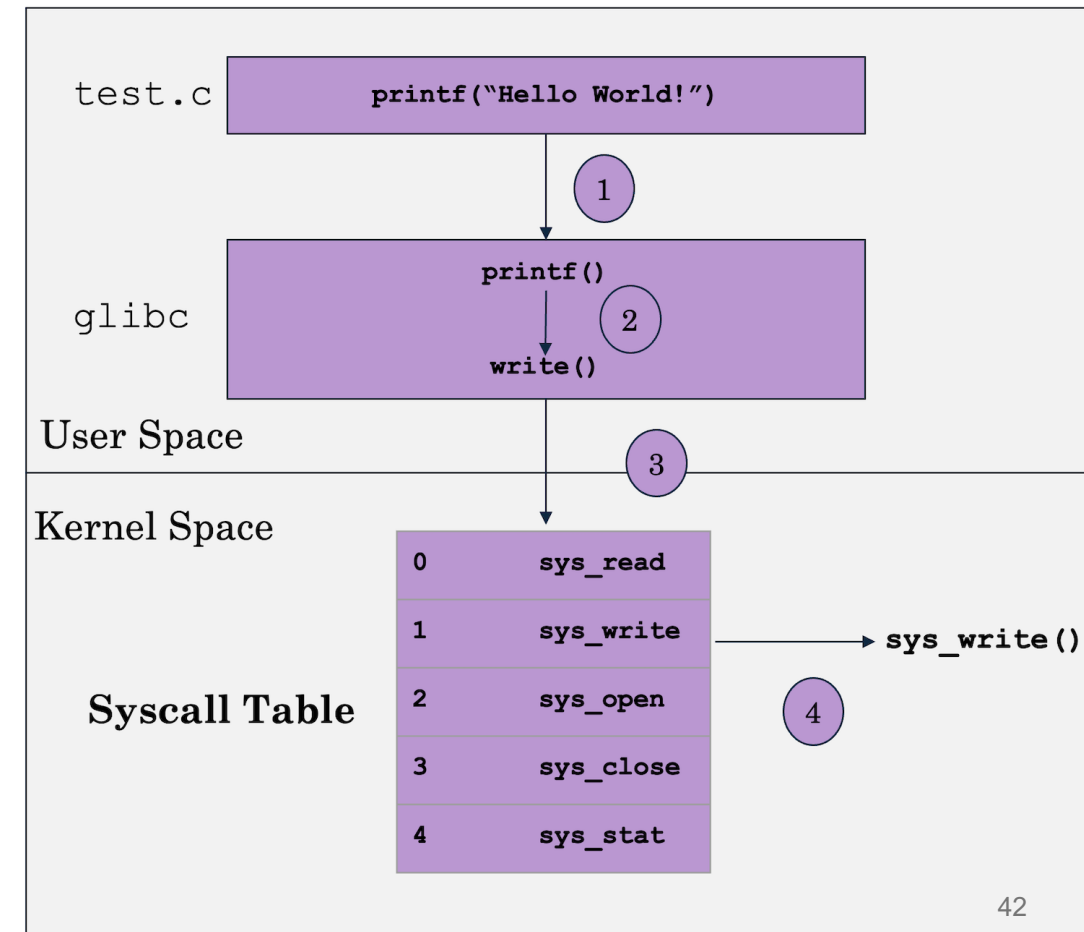
Operating systems provide four primary mechanisms for switching from user mode to kernel mode, each serving different purposes.

- **System calls**
- **Signals**
- **Hardware interrupts**
- **Traps (software interrupts/exceptions)**

System Calls

System calls are the primary interface for user programs to request kernel services. When a program needs to perform privileged operations like file I/O, memory allocation, or process creation, it makes a system call. The transition typically involves:

- Loading system call number and parameters into registers
- Executing a special instruction (like `int 0x80` or `syscall`)
- CPU switches to kernel mode and jumps to system call handler
- Kernel validates parameters and executes the requested service



Signals are a form of inter-process communication that can trigger kernel mode transitions:

- Process receives signal (SIGKILL, SIGTERM, SIGINT, etc.)
- Kernel suspends normal execution
- Switches to kernel mode to handle signal delivery
- May invoke user-defined signal handler or default action
- Can result in process termination, suspension, or custom handling

Hardware Interrupts



Hardware interrupts are involuntary signals from external devices that demand immediate attention, like when a keyboard generates an interrupt after a key press or a network card signals packet arrival. When an interrupt occurs:

- Hardware sends an interrupt signal to the CPU
- CPU finishes current instruction and saves context
- Switches to kernel mode and jumps to appropriate interrupt handler
- Kernel services the interrupt (keyboard input, disk completion, timer, etc.)
- Returns to interrupted process or schedules another process

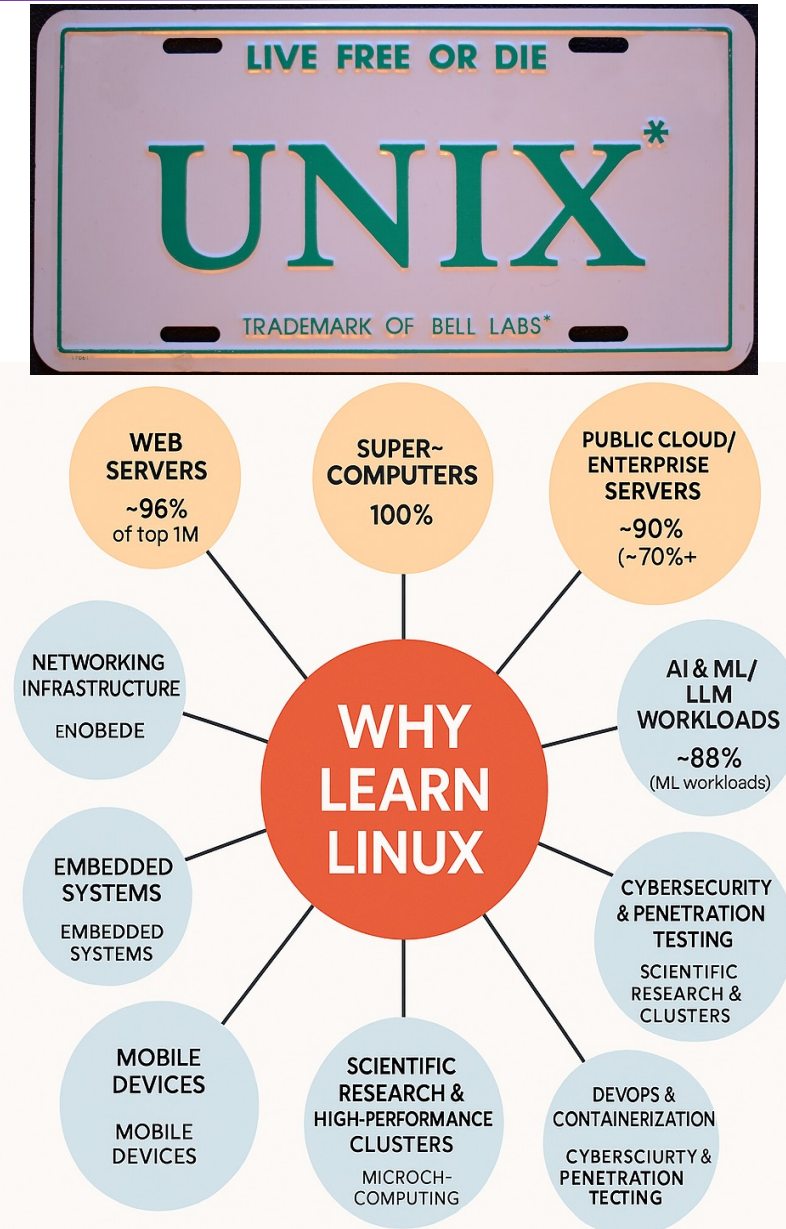
Traps (software interrupts/exceptions) are synchronous events caused by the currently running process.

- Divide by zero: CPU detects illegal arithmetic operation
- Page faults: Memory access to unmapped or protected page
- Invalid opcodes: Attempt to execute illegal instruction
- Breakpoints: Debug interrupts triggered by debugger
- General protection faults: Privilege violations or segmentation errors

Why Linux?

- All of the world's Top 500 **supercomputers** run a Linux-based OS from 2017 till 2025 (to date).
- **Android**, the world's most widely used mobile OS (over 70% global market share), is built on Linux kernel.
- 96% of the top one million **web servers** run on Linux.
- 90% of **public cloud workloads** are powered by Linux.
- According to SQ Magazine (2025), 88% of **AI/ML workloads** (including those for LLMs) run on Linux-based platforms due to their flexibility and strong ecosystem.
- Most **internet backbone equipment** uses Linux for routing, packet filtering, and VPN services.
- Distributions like Kali Linux, Parrot OS, and BlackArch are industry standards for **ethical hacking and security research**.
- **Docker, Kubernetes, and most CI/CD tools** run best on Linux, powering modern software development and deployment pipelines.
- **Media Production & Animation studios** use Linux-based render farms for VFX, animation, and 3D modeling.
- **Automotives, drones, smart appliances, & medical devices** often run embedded Linux for reliability and customizability.
- **Smart TVs, home assistants, cameras, & industrial IoT devices** commonly run lightweight Linux variants (e.g., Yocto, Buildroot).





Instructor: Muhammad Arif Butt, PhD



The pwn College



- The **pwn college** (<https://pwn.college/linux-luminarium/>) is an education platform for students to learn and practice core operating system and cybersecurity concepts in a hands-on fashion.
- In martial arts terms, it is designed to take a “white belt” in cybersecurity through the journey to becoming a “blue belt”, able to approach (simple) cybersecurity competitions (CTFs) and wargames.
- Frank Herbert: *“The power to destroy a thing is the absolute control over it”.*
- pwn.college: *“The power to hack a thing is the absolute understanding of it.”*
- Start with **Linux Luminarium** module for learning the use of **Linux CLI**, and understand core Linux concepts along the shell commands.

Getting Started	Linux Luminarium	Computing 101	Playing With Programs
			
13 Hacking 2 Modules 10 Challenges	55 Hacking 15 Modules 108 Challenges	41 Hacking 7 Modules 69 Challenges	23 Hacking 5 Modules 255 Challenges

To Do



- Course Matrix: <https://www.arifbutt.me/>
- Install the appropriate hypervisor (VirtualBox/VMWare) and install Kali Linux on it.
- Go through the provided Linux Commands Cheat Sheet and practice shell commands.
- All the C programs that will make you practically understand the the core OS concepts are available at course GitHub Repository at (<https://github.com/arifpucit/OS-Codes>). Insha'Allah, we will have a separate session on git, however, for the time being just download the repository.
- Visit <https://pwn.college>, create an account and explore its Linux Luminarium module.
- OS Playlist at LearnwithArif: https://www.youtube.com/playlist?list=PL7B2bn3G_wfBuJ_WtHADcXC44piWLRzr8
- SP Playlist at LearnwithArif: https://www.youtube.com/playlist?list=PL7B2bn3G_wfC-mRpG7cxJMnGWdPAQTViW



Coming to office hours does NOT mean that you are academically weak!