Operating Systems

LECTURE 1 NOTES

Introduction to Operating System

- **What do Operating Systems Do?**

  - A computer system can be broken down into roughly four parts: *hardware, operating system, application programs*, and a *user*.
  - **The hardware** — the central processing unit (CPU), the memory, and the input/output (I/O) devices — provides the basic computing resources for the system.
  - **The application programs** — such as word processors, spreadsheets, compilers, and web browsers — define how these resources are used to solve users' computing problems. The operating system controls the hardware and coordinates its use among the various application programs for the various users.
  - We can also view a computer system as consisting of hardware, software, and data. The operating system provides the means for the proper use of these resources in the operation of the computer system. An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an ***environment*** within which other programs can do useful work.

➢ **User View**

- The user's view of the computer varies according to the interface being used.
- Many computer users sit with a laptop or in front of a PC with a monitor, keyboard, and mouse. Such a system is designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. In this case, the operating system is designed mostly for ***ease of use***, with some attention paid to performance and security and none paid to ***resource utilization*** — how various hardware and software resources are shared.

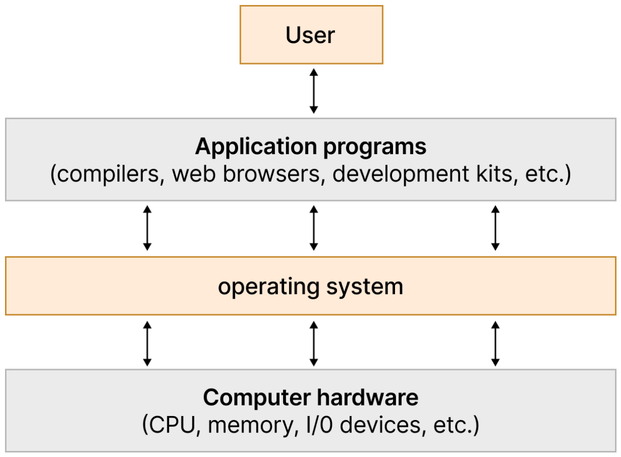

User
Application programs
(compilers, web browsers, development kits, etc.)
operating system
Computer hardware
(CPU, memory, I/0 devices, etc.)

Figure: *Abstract view of the components of a computer system*

- Many users now interact with mobile devices such as smartphones and tablets, which are replacing desktop and laptop computer systems for some users. These devices are commonly linked to networks via cellular or other wireless technologies. Mobile computer user interfaces typically include a **touch screen**, on which the user interacts with the system by pressing and swiping fingers across the screen rather than using a physical keyboard and mouse. Many mobile devices, such as **Apple's Siri**, also allow users to interact via **voice recognition**.

- Some computers have little or no user view. For example, **embedded computers,** in-home devices, and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems and applications are designed primarily to run without user intervention.

- **Application Users (or End Users)**—everyone who uses (or runs) applications or system programs falls into this category. We use an application when we use a word processor, a web browser, an email system, or a multimedia viewer. This group of users is most often called **simply users**, or sometimes **end users**.

- **Application Programmers**—individuals who create application programs such as word processors or email systems fall into this category. 'How do I read and write to a file?', 'How do I get a user's keystroke?, and 'How do I display this box?' are common questions programmers ask when learning to use a new operating system. The facilities provided by the OS are the programmers' perceptions of the OS. They are also known as system calls or application program interfaces (APIs). They can also appear as language library functions or as collections of classes. Programmers also want their software to be easily portable to other platforms.

- **Systems Programmers** are those who create software (either programs or components) that is tightly linked to the operating system. Examples of systems programs include a utility that displays the status of the computer's network connection and an installable driver for a piece of hardware. Systems programmers must have a thorough understanding of how the operating system works. In many cases, system programs require access to privileged system calls or special OS data structures.

- **System Administrators**—people who manage computer facilities and are thus in charge of installing and upgrading the operating system, as well as other system programs and utilities. They are also in charge of creating and managing user accounts, as well as protecting the system. They must have a thorough understanding of how the operating system is installed and upgraded, as well as how it interacts with other programs and utilities. They must also understand the OS's security and authorization features to protect their system and users effectively.

- As users, we expect a quick, dependable response (to keystrokes or mouse movement), a consistent user view (each type of command, such as scrolling or quitting an application, should be performed similarly), and other features that vary depending on the type of operating system.

➢ System View

- The operating system is the program that is most intimately involved with the hardware from the perspective of the computer. In this context, an operating system can be thought of as a **resource allocator**. A computer system has many resources that can be used to solve a problem, including CPU time, memory space, storage space, and I/O devices. These resources are managed by the operating system. When confronted with numerous and potentially conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that the computer system can be operated efficiently and fairly.
- A slightly different perspective on an operating system emphasizes the importance of controlling the various I/O devices and user programs. An operating system is a **command-and-control program**. A control program manages the execution of the user program to prevent errors and improper computer use. It is particularly interested in the operation and control of I/O devices.

Defining Operating Systems

- The term 'operating system' refers to a variety of roles and functions. This is due, at least in part, to the numerous computer designs and applications. Computers can be found in toasters, automobiles, ships, spacecraft, homes, and businesses. They serve as the foundation for video game consoles, cable TV tuners, and industrial control systems.
- We can look at the history of computers to explain this diversity. Although computers have only been around for a short time, they have evolved quickly. Computing began as an experiment to see what could be done and quickly evolved into fixed-purpose systems for military applications such as code breaking and trajectory plotting, as well as governmental applications such as census calculation. These early computers evolved into general-purpose, multifunction mainframes, which gave rise to operating systems. **Moore's Law** predicted in the 1960s that the number of transistors on an integrated circuit would double every 18 months, and that prediction has proven correct. Computers gained functionality while shrinking in size, resulting in a wide range of applications and operating systems.
- So, how exactly do we define an operating system? In general, there is no perfect definition of an operating system. Operating systems exist to provide a reasonable solution to the problem of creating a usable computing system. The primary goal of computer systems is to execute programs and facilitate the resolution of user problems. This is the goal of computer hardware design. Because bare hardware is not particularly user-friendly, the application program is created. Certain common operations, such as those controlling I/O devices, are required by these programs.

- The common functions of controlling and allocating resources are then combined into a single piece of software known as the operating system.
- Further, there is no universally accepted definition of the term an operating system. In simple terms, it includes everything that a vendor ship when you order ‘the operating system’. However, the features included vary greatly between systems. Some systems require less than a megabyte of space and do not even have a full-screen editor, whereas others require gigabytes of space and are entirely based on graphical windowing systems. A more common definition, and the one we usually use, is that the operating system is the single program that is always running on the computer — usually referred to as the **kernel**.
- Along with the kernel, there are two other types of programs: **system programs**, which are associated with the operating system but are not necessarily part of the kernel, and **application programs,** which include all programs not associated with the operation of the system.
- As personal computers became more common and operating systems became more sophisticated, the question of what constitutes an operating system became increasingly important. The US Department of Justice sued Microsoft in 1998, claiming that Microsoft included too many functionalities in its operating systems, preventing application vendors from competing. (For example, Microsoft's operating systems included a web browser.) As a result, Microsoft was found guilty of limiting competition by abusing its operating system monopoly.
- However, if we look at mobile operating systems today, we can see that the number of features that comprise the operating system is increasing once again.
- Mobile operating systems frequently include **middleware**, which is a collection of software frameworks that provide additional services to application developers. For example, the two most popular mobile operating systems — iOS Apple's and Google's Android — each include a core kernel as well as middleware that supports databases, multimedia, and graphics (to name only a few).
- In summary, the operating system includes the kernel, which is always running, middleware frameworks that ease application development and provide features, and system programs that help manage the system while it is running. The kernel of general purpose operating systems, but other components are discussed as needed to fully explain operating system design and operation.

➢ Historical Background and Evolution of Operating Systems

- To understand operating systems better, let's take a brief look at their historical background and evolution. Operating systems have evolved significantly over time, starting from simple batch processing systems to modern multitasking, multi-user systems. Let's explore some significant milestones in operating system development:

- **Batch Processing Systems:** In the early days of computing, computers were used for batch processing. In this model, users would submit a batch of jobs or programs, and the computer would process them one by one. The operating system would read the batch, execute each job, and provide the output. This approach allowed efficient utilization of computing resources.

- **Multiprogramming Systems:** With advancements in hardware, operating systems evolved to support multiprogramming. In multiprogramming systems, multiple programs could be loaded into memory simultaneously, and the operating system would allocate CPU time to each program in a time-sharing manner. This led to better resource utilization and faster job completion.

- **Time-Sharing Systems:** Time-sharing systems took multiprogramming a step further by allowing multiple users to simultaneously access and interact with the system. Each user was provided with a time slice or quantum of CPU time, and the operating system rapidly switched between users, giving an illusion of concurrent execution. This enabled interactive computing and facilitated the development of multi-user environments.

- **Personal Computer Operating Systems:** The advent of personal computers (PCs) brought operating systems to the masses. Operating systems such as Microsoft Windows and macOS were developed to provide a user-friendly interface and support a wide range of applications. These operating systems introduced features such as graphical user interfaces, file management, and device drivers tailored to personal computing needs.

- **Networked and Distributed Systems:** With the growth of networking technologies, operating systems evolved to support networked and distributed computing. Network operating systems enable multiple computers to communicate and share resources over a network. Distributed operating systems manage interconnected computers as a unified system, allowing distributed processing and resource sharing.

➢ Classification of Operating Systems Based on Different Criteria

- Operating systems can be classified based on various criteria. Let's explore some common classifications:

1. **Single-user vs Multi-user:** Operating systems can be classified as single-user or multi-user systems. Single-user systems are designed to support a single user at a time, while multi-user systems allow multiple users to access and use the system simultaneously.

2. **Single-tasking vs Multi-tasking:** Operating systems can be categorized as single-tasking or multi-tasking systems. Single-tasking systems allow the execution of only one task or program at a time while multi-tasking systems enable concurrent execution of multiple tasks or programs.

3. **Batch Processing vs Interactive:** Operating systems can be classified as batch processing systems or interactive systems. Batch processing systems execute programs in batches without user interaction, while interactive systems allow users to interact with the system in real-time.

4. **Distributed Operating Systems:** Distributed operating systems are designed to run on multiple interconnected computers, sharing resources and providing a unified environment. They enable distributed computing and resource sharing across a network.

5. **Real-time Operating Systems:** Real-time operating systems are designed to meet strict timing constraints. They are used in applications where timely execution is critical, such as industrial control systems, robotics, and aerospace.

- **Operating System Architecture**

  - Operating system architecture refers to the design and structure of an operating system. It determines how various components and modules of an operating system interact and work together to provide the necessary services to users and applications. There are different types of operating system architectures, and each has its own advantages and features.

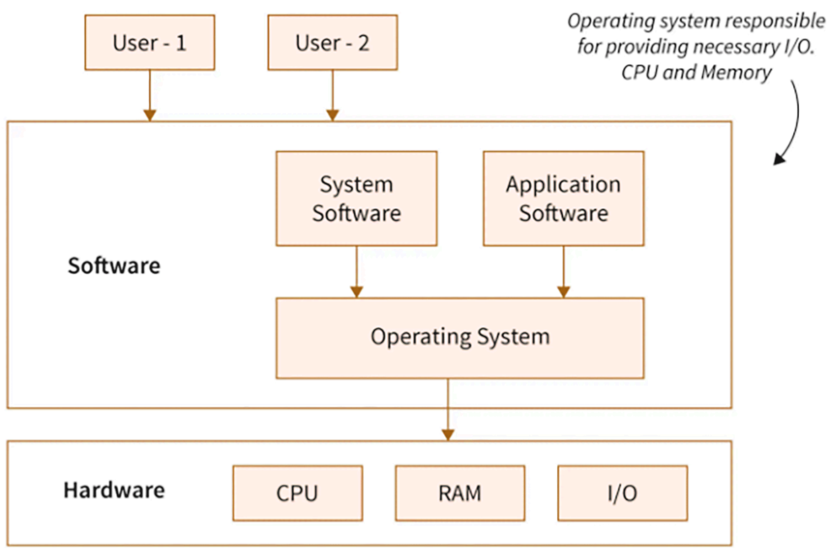
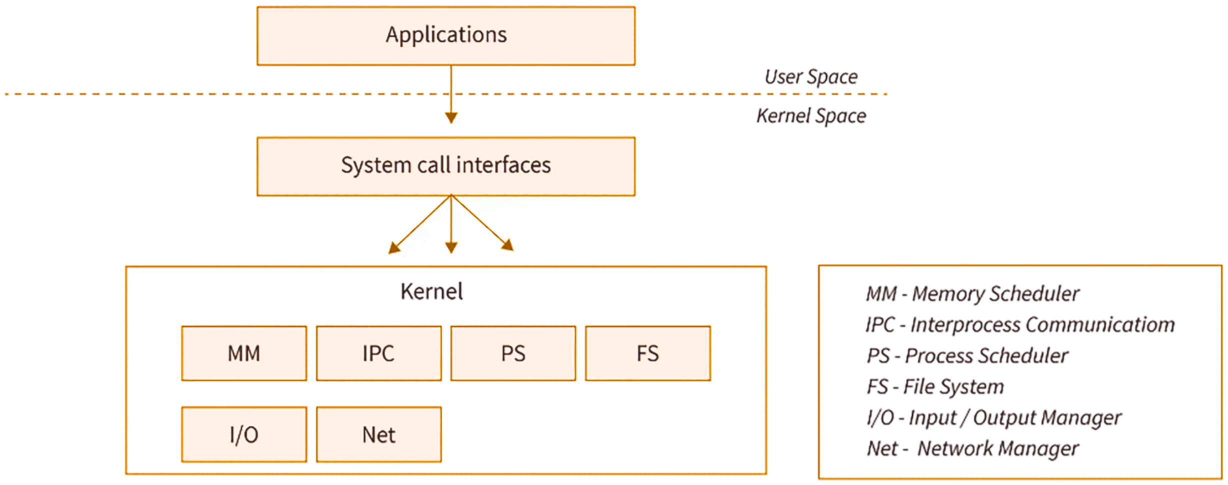
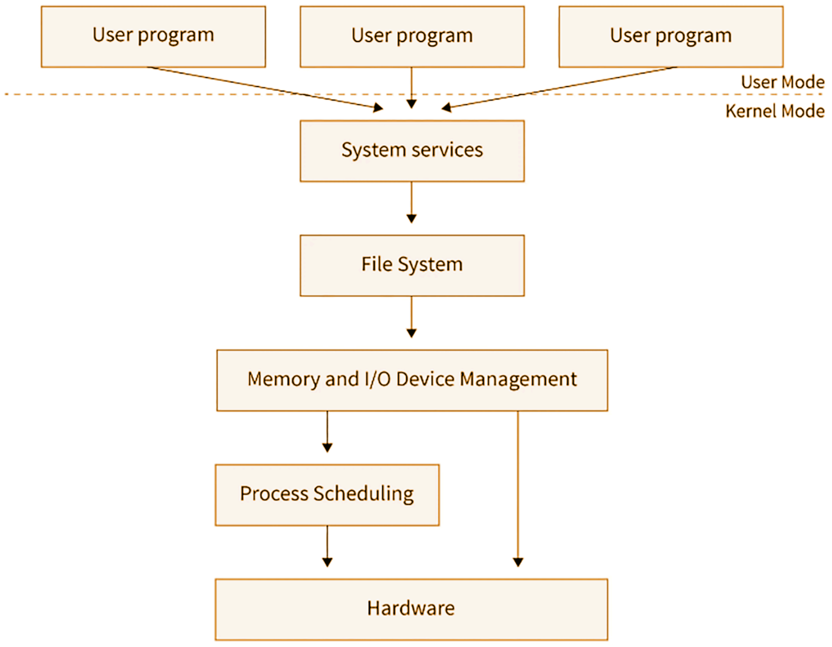


Figure: *A typical operating system architecture*

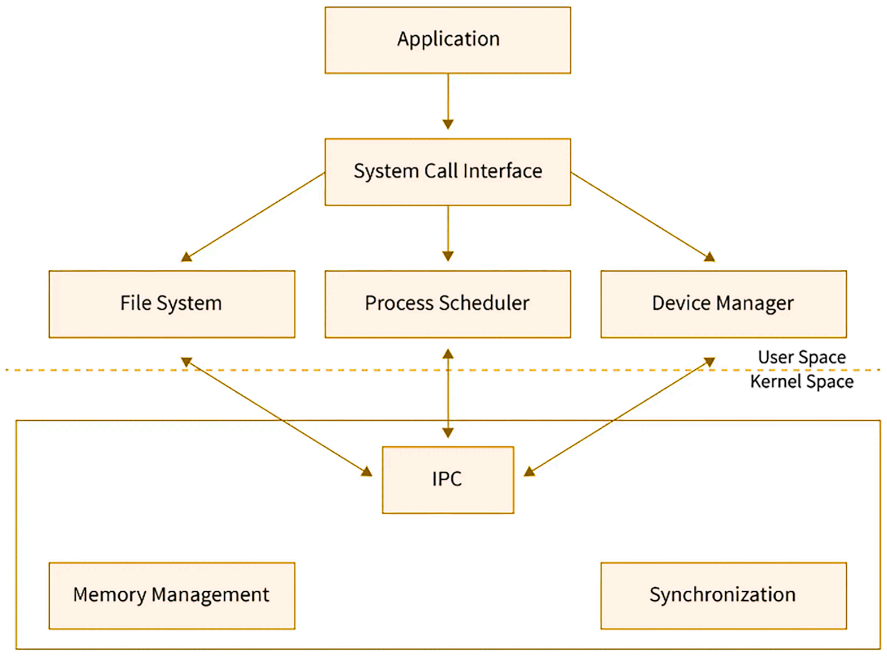
1. **Monolithic Architecture:** This architecture is one of the earliest and simplest operating system architectures. In this architecture, all operating system components, such as process management, memory management, file system, and device drivers, are tightly integrated into a single large executable binary. The components directly invoke functions and procedures from one another.



2. **Layered Architecture:** The layered architecture is based on the concept of dividing the operating system functionality into separate layers, each responsible for a specific set of functions. Each layer provides services to the layer above it and utilizes services from the layer below it. This modular approach simplifies system design and allows for easier maintenance and modification of individual layers.



3. **Microkernel Architecture:** This architecture aims to keep the kernel as minimal as possible, providing only essential services such as process scheduling, inter-process communication, and memory management. Non-essential services, such as file systems and device drivers, are moved out of the kernel and run as separate user-space processes. This design improves system reliability, flexibility, and extensibility.

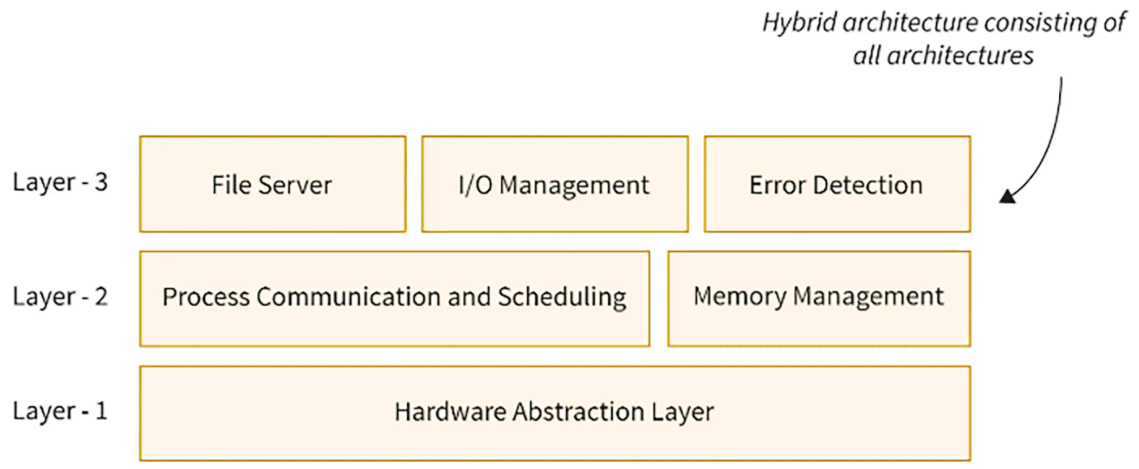


4. **Client-Server Architecture:** In client-server architecture, the operating system services are divided into server processes that provide specific services and client processes that request and use those services. The client-server model allows for distributed and scalable systems where multiple clients can access services provided by servers over a network. This architecture is commonly used in network operating systems and distributed systems.

5. **Virtual Machine Architecture:** Virtual machine architecture involves the use of a virtualization layer called the hypervisor, which allows multiple operating systems, known as guest operating systems, to run concurrently on a single physical machine. Each guest operating system runs in its virtual machine, providing isolation and independence. This architecture is commonly used in server virtualization and cloud computing environments.

6. **Hybrid Architectures:** In practice, operating systems often employ a combination of different architectures, leading to hybrid architectures. For example, a modern operating system may use a microkernel design for its core services while incorporating layered architecture for additional functionality and device support.

Hybrid architectures aim to leverage the advantages of different architectural approaches to achieve a balance between performance, functionality, and flexibility.


Hybrid architecture consisting of all architectures
Layer - 3
File Server
I/O Management
Error Detection
Layer - 2
Process Communication and Scheduling
Memory Management
Layer - 1
Hardware Abstraction Layer

- **Boot Process**
  - The boot process is a sequence of steps that a computer system undergoes when it is powered on or restarted. It is the initial process that initializes the hardware components and loads the operating system into the computer's memory, allowing it to become functional and ready for use. Let's explore the typical steps involved in the boot process:
    - **Power-On Self-Test (POST):** When the computer is powered on, the system's firmware, typically stored in the computer's read-only memory (ROM), executes the Power-On Self-Test (POST). The POST checks the hardware components to ensure they are functioning properly. It verifies the integrity of the processor, memory, storage devices, and other peripheral devices.
    - **BIOS/UEFI Initialization:** After the POST, the Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) initializes the system. The BIOS/UEFI is responsible for providing the low-level software interface between the operating system and the hardware. It identifies and configures the system's hardware devices, sets up the interrupt handlers, and initializes the system clock.
    - **Bootstrap Loader:** Once the hardware initialization is complete, the BIOS/UEFI locates and executes the bootstrap loader program. The bootstrap loader is a small program stored in the system's boot device (typically the hard drive's master boot record or a solid-state drive's firmware). Its purpose is to locate the operating system's kernel and load it into memory.

- **Operating System Kernel Loading:** The bootstrap loader locates the operating system kernel, which is typically stored in a specific location on the boot device. It reads the kernel into memory and hands over control to the operating system.
- **Kernel Initialization:** The operating system kernel takes control of the system and begins its initialization process. It sets up essential data structures, establishes the system's initial configuration, and initializes device drivers required for basic functionality.
- **Device Initialization:** Once the kernel is initialized, it proceeds to initialize and configure the hardware devices connected to the system. This includes initializing input/output devices such as keyboards, mice, and displays, as well as storage devices, network interfaces, and other peripherals.
- **User Mode Initialization:** After device initialization, the operating system transitions to user mode. It sets up the necessary user mode processes and prepares the environment for user interaction. This may include starting system services, launching the graphical user interface (GUI), and providing login prompts for user authentication.
- **User Login and System Use:** With the boot process complete, the operating system presents a login screen or prompts for user credentials. Once the user logs in, the system is ready for use, and the user can interact with the operating system, launch applications, and perform various tasks.

▪ It's important to note that the boot process may vary depending on the specific hardware architecture, firmware, and operating system in use.

➢ Current Trends and Future Developments in Operating Systems

▪ Operating systems continue to evolve with advancements in hardware, software, and computing paradigms. Let's explore some current trends and future developments in operating systems:

- **Virtualization:** Virtualization allows the creation of multiple virtual machines or containers on a single physical machine, enabling efficient resource utilization, isolation, and flexibility. It enables running multiple operating systems simultaneously on a single physical machine.
- **Cloud Computing:** Cloud computing platforms heavily rely on operating systems to manage virtualized resources and provide services to users over the internet. Operating systems play a crucial role in ensuring the reliability, scalability, and security of cloud-based services.

- **Containerization:** Containerization technologies such as Docker and Kubernetes have gained popularity. They use operating system-level virtualization to run applications in isolated containers, enabling easy deployment, scalability, and portability. Containers provide a lightweight and consistent environment for running applications across different operating systems and platforms.

- **Internet of Things (IoT):** Operating systems tailored for IoT devices are emerging. These lightweight operating systems provide connectivity, manage resources, and support the execution of applications on resource-constrained IoT devices. They are designed to handle the unique challenges of IoT, such as limited processing power, memory, and energy constraints.

- **Security and Privacy:** Operating systems are continuously improving security measures to protect against various threats. Enhancements in access control, authentication, encryption, and intrusion detection systems are being incorporated to ensure the security and privacy of user data and system resources.