Unit 01

Introduction to operating system

Introduction

- An operating system (OS) is a set of programs that control the execution of application programs and act as an intermediary between a user of a computer and the computer hardware.
- An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.
- OS is software that manages the computer hardware as well as providing an environment for application programs to run.
- Examples of OS are: Windows, Windows/NT, OS/2 Linux, macOS, Android, and iOS

The concept of an operating system involves managing computer resources, providing abstractions for software, handling processes, threads, and memory, managing files and devices, ensuring security, offering a user interface, and facilitating communication between hardware and software through system calls. The kernel is the core, and system calls are interfaces for user programs to request services from the operating system. Concurrency, interrupts, and networking are also key aspects. The operating system aims to provide a stable and efficient computing environment.

Types of operating system

Operating systems come in various types, each designed to meet specific requirements and cater to different computing environments. Here are some common types of operating systems:

1. **Single-User, Single-Tasking:**
- Also known as a single-user, single-task operating system, it supports only one user at a time and allows the execution of only one task at a time. Older personal computers and some embedded systems might use this type.
2. **Single-User, Multi-Tasking:**
- This type of operating system allows a single user to run multiple programs concurrently. Modern desktop and laptop operating systems like Windows, macOS, and Linux fall into this category.
3. **Multi-User Operating System:**
- Multi-user operating systems support multiple users simultaneously. Each user can log in, run programs, and perform tasks independently. Examples include Unix and Linux server distributions, as well as mainframe operating systems like IBM z/OS.
4. **Real-Time Operating System (RTOS):**
- RTOS is designed for systems that require real-time processing, where tasks must be completed within a specified time frame. RTOS is used in applications such as embedded systems, robotics, and control systems.
5. **Distributed Operating System:**
- Distributed operating systems manage multiple computers and coordinate their resources to provide a unified computing environment. These systems are designed for distributed computing and may be used in cluster computing or cloud environments.
6. **Network Operating System (NOS):**

- NOS is designed to support network resources and services. It enables file sharing, printer access, and other network-related functionalities. Novell NetWare and Windows Server are examples of network operating systems.

7. **Mobile Operating System:**

- Mobile operating systems are designed for smartphones, tablets, and other mobile devices. Examples include Android, iOS, and HarmonyOS. These systems are optimized for touch input, mobile hardware, and mobile app ecosystems.

8. **Time-Sharing Operating System:**

- Time-sharing operating systems allow multiple users to share the computer's resources by dividing the CPU time into slices. This gives the appearance of concurrent execution, even though each user's task is executed in a time-shared manner.

9. **Hybrid Operating System:**

- Hybrid operating systems combine elements of different types to provide a balance of features. For example, a system may have characteristics of both monolithic and microkernel architectures. Linux is often considered a hybrid operating system.

Operating system operation

Operating systems (OS) are crucial software components that manage computer hardware and provide services for computer programs. They act as an intermediary between the hardware and the applications, facilitating the execution of various tasks and ensuring efficient resource utilization. Here's an overview of the basic operations of an operating system:

Booting:

- The process of starting or restarting a computer is known as booting.
- During this phase, the computer's firmware (such as BIOS or UEFI) initializes hardware components and loads the operating system into the computer's memory.

Process Management:

- The OS manages processes, which are instances of running programs.
- It allocates resources, such as CPU time and memory, to different processes.
- Context switching allows the OS to switch between different processes, enabling multitasking.

Memory Management:

- OS is responsible for managing the computer's memory.
- It allocates memory space to processes and ensures that they do not interfere with each other.
- Virtual memory may be used to extend available memory by using disk space as a temporary storage.

File System Management:

- The OS organizes and manages files on storage devices.
- It provides a file system that allows for the creation, deletion, and manipulation of files and directories.
- File permissions and access control are enforced by the OS.

Device Management:

- ★ OS interacts with and manages various hardware devices, including input/output devices (keyboard, mouse, printer) and storage devices (hard drives, SSDs).
- ★ Device drivers are used to facilitate communication between the operating system and hardware components.

Security and Access Control:

- ★ OS implements security measures to protect the system and its data.
- ★ User authentication, access control, and encryption are common security features.

User Interface:

- ★ OS provides a user interface that allows users to interact with the computer.
- ★ User interfaces can be command-line interfaces (CLI) or graphical user interfaces (GUI).

Networking:

- ★ Many modern operating systems include networking capabilities.
- ★ They manage network connections, protocols, and communication between devices.

Error Handling:

- ★ The OS monitors system activities and handles errors that may occur during operation.
- ★ It may provide error messages, log events, and attempt to recover from certain types of errors.

Updates and Maintenance:

- ★ OS vendors release updates and patches to fix bugs, improve performance, and enhance security.
- ★ The OS may also facilitate the installation and management of software updates.

Operating System Services

Operating systems provide a range of services to both users and applications to facilitate the effective and efficient use of computer resources. These services can be categorized into several key areas:

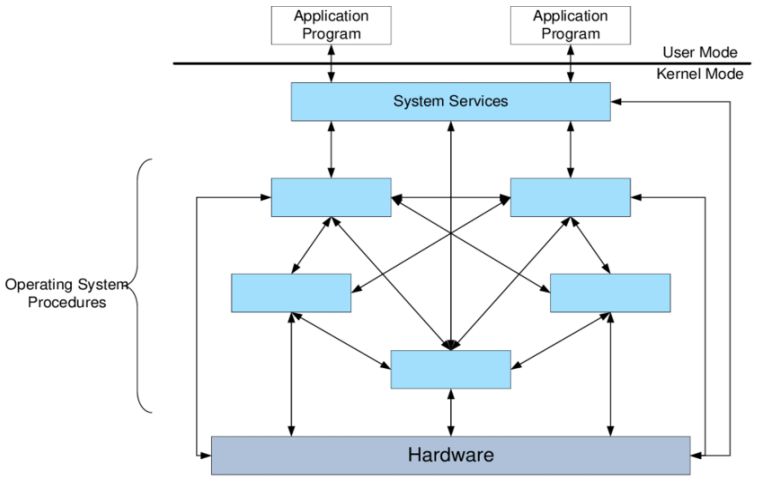
1. **Program Execution:**
- ➢ The operating system loads programs into memory and schedules them for execution on the CPU. It manages the execution of processes, facilitating multitasking and ensuring fair and efficient use of the CPU.
2. **I/O Operations:**
- ➢ The OS provides services for input and output operations, allowing programs to interact with devices such as keyboards, displays, printers, and storage devices. It manages I/O buffers, handles interrupts, and provides a consistent interface for device communication.
3. **File System Manipulation:**
- ➢ Operating systems offer file-related services, including file creation, deletion, reading, and writing. They manage file permissions, directory structures, and access control to ensure the organized storage and retrieval of data.
4. **Communication Services:**
- ➢ Operating systems enable communication between processes through inter-process communication (IPC) mechanisms. This allows processes to exchange data and synchronize their activities.
5. **Error Detection and Handling:**
- ➢ The OS monitors the system for errors and provides mechanisms for error detection and recovery. It may generate error messages, log events, and take corrective actions to maintain system stability.

6. **Resource Allocation:**
   - The OS manages system resources, including CPU time, memory space, and I/O devices. It allocates resources to processes, ensures fair distribution, and prevents conflicts among competing processes.
7. **Security and Protection:**
   - Operating systems implement security measures to protect the system and user data. This includes user authentication, access control, encryption, and the enforcement of security policies.
8. **User Interface Services:**
   - The OS provides a user interface that allows users to interact with the computer. This can be a command-line interface (CLI) or a graphical user interface (GUI). User interfaces facilitate communication between the user and the system.
9. **Networking:**
   - Modern operating systems often include networking services to support communication over networks. They manage network connections, protocols, and data transfer between devices.
10. **System Call Interface:**
    - Applications interact with the operating system through a set of system calls. These calls provide a standardized interface for accessing OS services, allowing applications to request operations such as file I/O, process creation, and memory allocation.

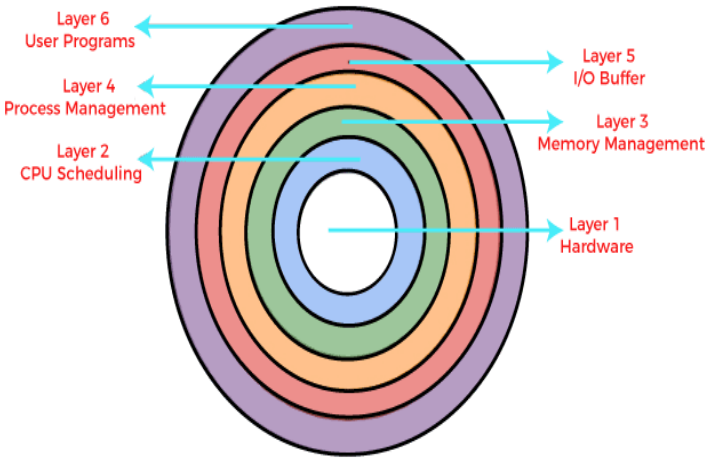
Operating System Structure

Operating systems can be classified based on their structure into several broad categories. The classification is often based on how the operating system components are organized and how they interact. Here are some common classifications based on structure:

1. **Monolithic Operating Systems:**
   - In a monolithic operating system, the entire operating system is a single, large program running in kernel mode. All operating system services, including process management, memory management, and file systems, are part of this monolithic kernel. Examples include early versions of Unix and MS-DOS.

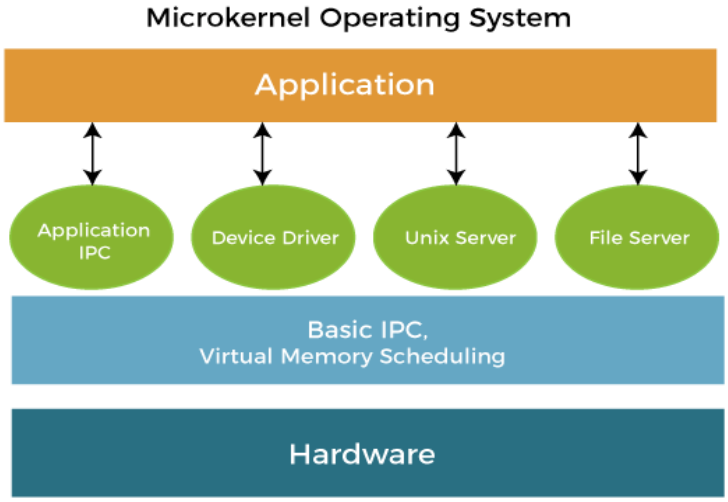


2. **layered Operating Systems:**
   - Layered operating systems organize the OS into layers, where each layer provides a specific set of services. Each layer communicates only with adjacent layers, creating a modular and hierarchical structure. This design simplifies maintenance and modification. However, communication between layers can introduce overhead. The THE (Technische Hogeschool Eindhoven) operating system is an example of a layered system.



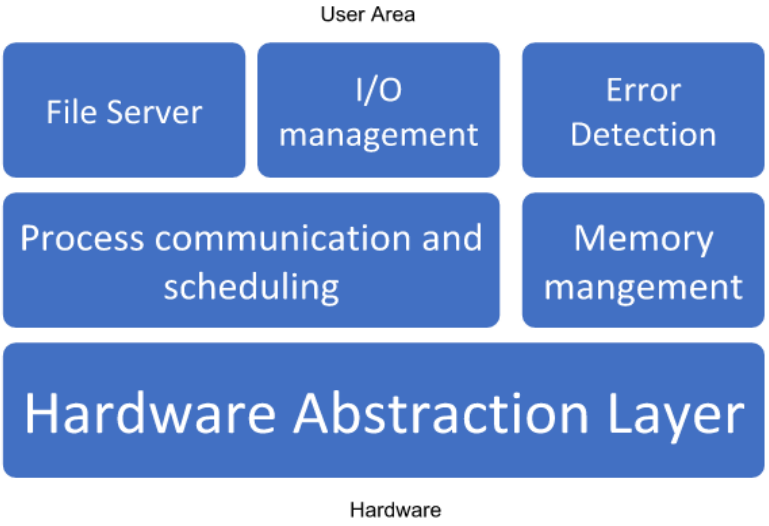
3. **Microkernel-based Operating Systems:**

- Microkernel architectures move non-essential components, such as device drivers and file systems, out of the kernel and into user space as separate processes. The microkernel itself provides only essential services like inter-process communication and process management. This design aims to improve system stability and security. Examples include QNX and L4.



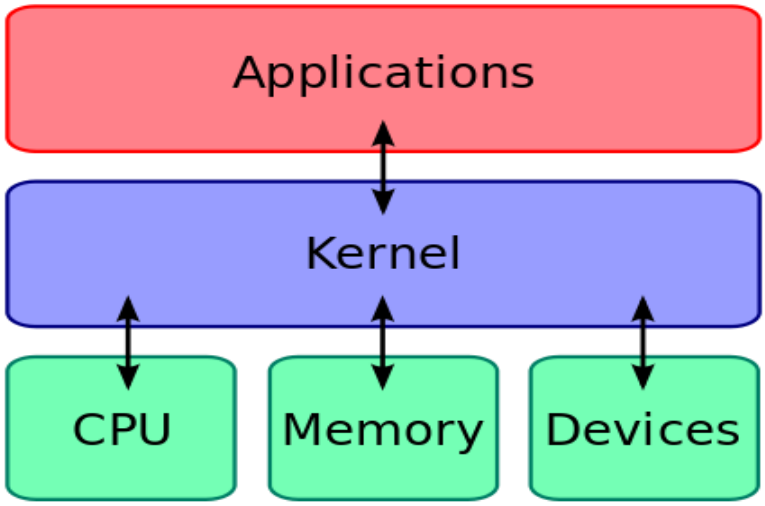
4. **Hybrid Operating Systems:**

- Hybrid operating systems combine elements of both monolithic and microkernel designs. They include a small, efficient kernel that manages essential services, and non-essential services are kept in separate modules or user space. This approach attempts to balance the performance advantages of monolithic designs with the flexibility and stability benefits of microkernel architectures. Linux can be considered a hybrid system.



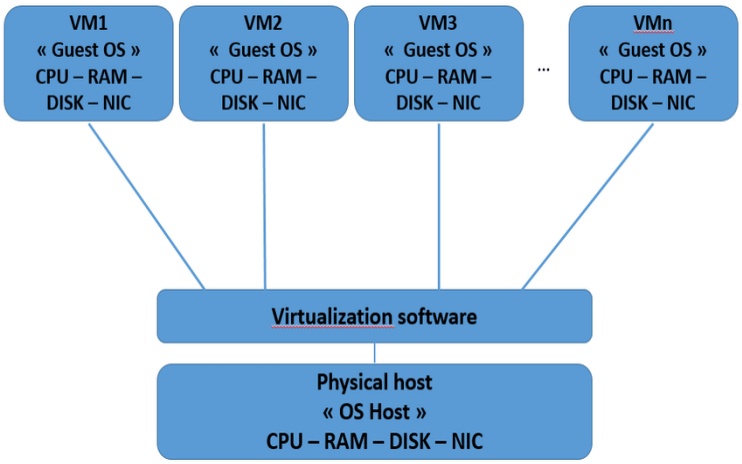
5. **Exokernel Operating Systems:**

- Exokernel architectures expose hardware resources directly to applications, allowing them to manage resources themselves. The kernel's role is reduced to enforcing security and resource protection. Exokernels aim to provide maximum flexibility and performance to applications. Examples include ExOS and Nemesis.

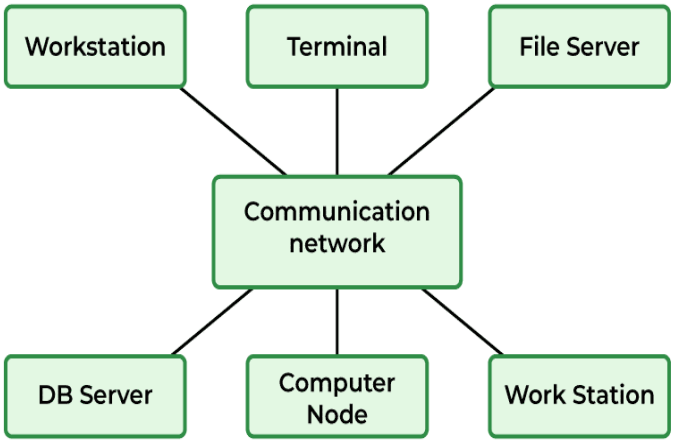


6. **Virtual Machine (VM) Operating Systems:**

- VM operating systems, also known as hypervisors, create a virtualized environment on top of a host operating system. They allow multiple operating systems to run simultaneously on the same hardware. Examples include VMware, Microsoft Hyper-V, and KVM.

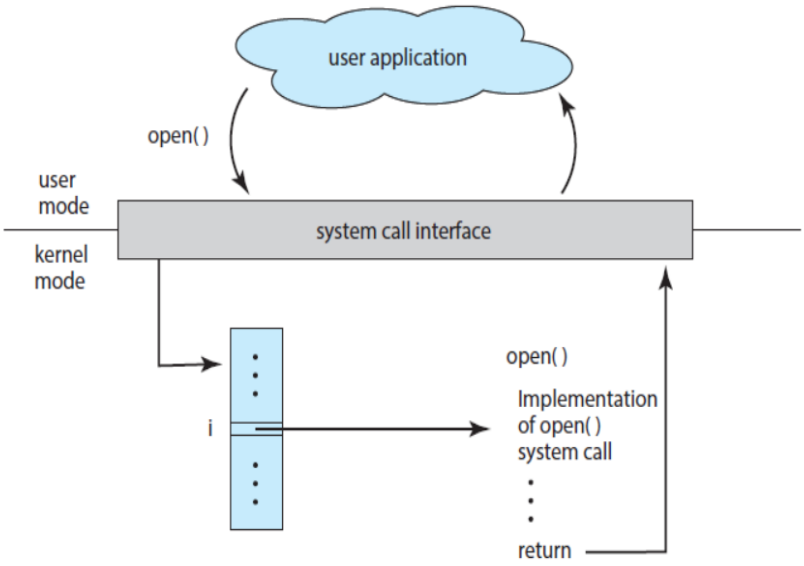


7. **Client-Server Operating Systems:**
   - In client-server architectures, the operating system is split into server and client components. The server provides core services like file sharing, printing, and network management, while clients access and utilize these services. This approach is common in networked environments.
8. **Distributed Operating Systems:**
   - Distributed operating systems extend the concept of client-server architectures, allowing processes to run on multiple interconnected computers. These systems provide services that are distributed across the network, enabling collaboration and resource sharing. Examples include Plan 9 and Amoeba.



System Call

A system call, often abbreviated as syscall, is a mechanism used by an application program to request a service from the operating system's kernel. It serves as the interface between user-level processes and the kernel, enabling applications to perform tasks that require privileged access or interaction with hardware. System calls are essential for programs to access operating system services and resources.


user application
open( )
user mode
kernel mode
system call interface
i
open( )
Implementation of open( ) system call
return

THE END