# **Operating System**

An operating system or OS is system software that works as an interface between hardware components and the end-user. It enables other programs to run. Each computer system, whether it is a desktop, laptop, tablet, etc must have an OS to provide basic functionalities for the device. Some widely used operating systems are Windows, Linux, MacOS, Android, iOS, etc.

**What is a kernel?**

* It acts as a bridge between applications and data processing done at the hardware level. It is the central component of an OS. It is one of the first programs loaded on start-up (after the Bootloader).
* It also manages memory, peripheral, and I/O requests from software. Moreover, it translates all I/O requests into data processing instructions for the CPU. It manages other tasks also such as memory management, task management, and disk management.
* A kernel is kept and usually loaded into a separate memory space, known as protected Kernel space. It is protected from being accessed by application programs or less important parts of the OS.
* Other application programs such as browsers, word processors, and audio & video players use separate memory space known as user space.
* Due to these two separate spaces, user data and kernel data don't interfere with each other and do not cause any instability or slowness.

**Functions of a Kernel**

* **Device Management:** To perform various actions, processes require access to peripheral devices such as a mouse, keyboard, etc., that are connected to the computer. A kernel is responsible for controlling these devices using device drivers. Here, a device driver is a computer program that helps or enables the OS to communicate with any hardware device.A kernel maintains a list of all the available devices.
* **Memory Management:** Each process requires some memory to work, and the kernel enables the processes to safely access the memory. To allocate the memory, the first step is known as virtual addressing, which is done by paging or segmentation. Virtual addressing is a process of providing virtual address spaces to the processes.
* **Resource Management:** One of the important functionalities of Kernel is to share the resources between various processes. It must share the resources in a way that each process uniformly accesses the resource. The kernel also provides a way for synchronization and inter-process communication (IPC). It is responsible for context-switching between processes.

**Types of kernel:**

1. Monolithic Kernels

2. Microkernel

3. Hybrid Kernel

4. Nanokernel

5. Exokernel

**Microkernel Architecture in Operating Systems**

A microkernel is a type of operating system kernel that is designed to provide only the most essential services required for the system to function. These core services typically include:

**Memory Management:** Allocating and managing the memory resources of the system.

**Process Scheduling:** Managing the execution of processes, including their creation, termination, and state switching.

Other services, such as device drivers, file systems, and network protocols, are implemented as user-level processes that interact with the microkernel through message passing. This contrasts with monolithic kernels, where these services are integrated directly into the kernel space.

### **Advantages of Microkernel Architecture**

* **Reduced Attack Surface:** With fewer services running in kernel space, there are fewer points of entry for potential attacks. User-level processes are isolated, so a vulnerability in one service is less likely to compromise the entire system.
* **Modularity and Flexibility:** Easier to Modify and Extend: Since services are implemented as user-level processes, it is easier to add, remove, or update them without needing to change the kernel. This allows for more straightforward customization and adaptation to specific requirements.
* **Service Isolation:** Services run in their own address spaces, reducing the risk of interference between them. This modular approach simplifies debugging and testing.

**Stability and Reliability:**

* **Crash Resilience:** The core functionality provided by the microkernel remains unaffected by the crashes of user-level services. This improves the overall stability of the operating system.

### **Disadvantages of Microkernel Architecture**

* **Performance Overhead**: Message Passing Latency- Communication between user-level processes and the microkernel via message passing can introduce overhead. This is generally slower compared to direct system calls in a monolithic kernel, potentially affecting performance, especially in high-performance applications.
* **Context Switching:** Frequent context switching between user-level processes and the kernel can further degrade performance.

**Comparison with Monolithic Kernels**

**Monolithic Kernels:**

* **Integrated Services:** All services, including device drivers, file systems, and network protocols, are part of the kernel. This allows for fast system calls and efficient execution.
* **Larger Attack Surface:** With more code running in kernel space, there are more potential vulnerabilities.
* **Less Modularity:** Modifying or updating kernel services often requires recompiling and rebooting the entire kernel.

### Kernel Mode and User Mode of CPU Operation

Modern CPUs operate in two primary modes: kernel mode and user mode.

**Kernel Mode:**

* **Privilege Level:** The CPU has full access to all system resources and can execute any instruction, including privileged instructions.
* **Privileged Instructions:** These are special instructions that can control hardware directly, manage memory, or perform other critical operations. Examples include modifying system control registers and managing memory protection.
* **Use Case:** The operating system operates in kernel mode to perform system-level tasks, such as handling hardware interrupts, managing memory, and scheduling processes.

**User Mode:**

* **Privilege Level:** The CPU has restricted access to system resources and cannot execute privileged instructions.
* **Non-Privileged Instructions:** User programs can execute standard instructions like arithmetic operations, data movement, and program control instructions (e.g., ADD, PUSH).
* **Use Case:** User applications run in user mode to prevent them from directly interfering with the operating system or other user programs