Tutorial 01

1. The operating system (OS) of a computer is a piece of software that acts as a conduit between users and computer hardware. It controls a variety of hardware components, such as input/output devices, memory, storage, CPUs, and network connections, and acts as a platform for the execution of other software applications. An operating system's primary responsibilities include managing file systems, enabling communication between hardware and software, assigning and managing system resources, providing an interface for users, and ensuring the overall stability and security of the system. Microsoft Windows, Linux, Unix, and macOS are popular operating systems.

1) Scheduling in Process Management: The operating system (OS) selects the order in which the CPU executes programs in order to guarantee efficient use of system resources.

Creation and Termination: The OS is in charge of starting, stopping, and suspending processes, which are essentially instances of active applications.

2) Memory Management: Allocation and Deallocation: The operating system (OS) manages how memory is allocated to different processes and ensures that memory is used efficiently.

Programs larger than RAM can be executed thanks to virtual memory, which provides a representation of the computer's real memory.

3) File System Management: Organizing and Storing: The reading, writing, and deleting of files as well as the arranging and management of files on storage media are all handled by the operating system (OS).

Access Control: It prevents unauthorized access and safeguards confidential data by limiting who can access certain files and folders.

4) Device Administration: Specific applications known as device drivers allow the operating system to communicate with specific hardware elements. They serve as the conduit between the operating system and hardware.

Input/Output (I/O): Monitoring input and output to and from peripheral devices, managing device disruptions, and managing data transfers are all under the purview of the operating system (OS).

03)

1) Reliability: An operating system should be dependable if it can continuously carry out its tasks successfully and without unanticipated mistakes or malfunctions. It ought to offer a steady environment for the software and hardware parts.

For essential system functions like file handling, process management, and data storage to be carried out correctly and regularly, reliability is essential.

2) Security: An operating system's security is essential for safeguarding information, assets, and the integrity of the system as a whole. It includes safeguards to maintain the confidentiality and integrity of user data, stop unwanted access, and identify and address security breaches.

In order to protect against malicious actions, unauthorized access, and potential dangers, the operating system should incorporate access control measures, encryption, authentication, and other security features.

3) Performance Efficiency: To guarantee responsive and effective system performance, an efficient operating system should maximize the utilization of hardware resources. This entails optimizing throughput and minimizing delays in the management of CPU, memory, and storage resources.

A responsive user experience and satisfying the demands of multiple programs operating concurrently on the system depend on performance efficiency. Performance is enhanced by effective memory management, I/O scheduling, and process scheduling.

04)

One essential part of an operating system (OS) is the kernel. It is the central component of the operating system (OS) that works directly with the hardware of computers and offers necessary services to the other components of the OS and software programs. System resources, including the CPU, memory, input/output devices, and file systems, are managed by the kernel. It serves as a bridge between the higher-level software layers and the hardware.

The following are some of the main duties and roles of the kernel:

1) Process management, which includes starting, stopping, and scheduling processes.

overseeing the coordination and communication of processes.

2) Memory Management: Assigning and releasing memory to applications.

putting memory protection and virtual memory into practice.

3) Device Management: Using device drivers in conjunction with hardware devices to control them.

managing operations for input and output.

4) Handling the creation, reading, writing, and deleting of files is known as file system management.

imposing restrictions on directories and files.

5) System Calls: Giving programs access to a list of system calls through which they can ask the kernel for services.

Operating in a privileged mode grants the kernel direct hardware access and the ability to carry out essential functions. It is the first part of the system to load into memory when it boots up and stays there the entire time the system is running.

05)

Benefits of Systems with Multiple Programming:

1) Higher CPU Utilization: Multiprogramming enables the simultaneous loading of numerous programs into memory. This guarantees that the CPU is nearly constantly engaged in carrying out commands, resulting in increased use of the computer's resources.

2) Increased Throughput: When the operating system has several programs loaded into memory, it can transition between them while it's idle or waiting on I/O activities. The overall throughput of the system is increased by this overlap of CPU and I/O processes.

3) Resource Utilization: By enabling the simultaneous execution of multiple programs, multiprogramming effectively uses system resources, such as CPU, memory, and I/O devices. In comparison to systems that execute a single application at a time, this leads to improved resource use.

4) Faster Response Times: Because the system can switch between several apps quickly, users notice faster response times. A more responsive environment can be created by allocating the CPU to another program while one is waiting on I/O.

5) Optimized System Performance: By reducing idle time and helping to balance the workload, multiprogramming enhances system performance. It guarantees that even when specific programs are awaiting outside events, the system remains operational and productive.

Benefits of Systems for Time Sharing:

1) Interactive User Environment: Time-sharing solutions allow several people to communicate with the computer at once. On the CPU, each user is assigned a time slice, also known as a time quantum, which enables them to run interactive programs and carry out tasks.

2) Equitable Resource Distribution: Time-sharing systems distribute CPU time equitably across several users, guaranteeing that every user receives a share of the system's resources. In a multiuser setting, this fairness is crucial to keep one user from controlling the entire system.

3) Economical: Time-sharing platforms offer economical ways to cater to several users. Resources are shared dynamically rather than needing a dedicated machine for every user, which lowers overall hardware and maintenance expenses.

4) Enhanced User Productivity: Users can work on multiple projects at once, and the system reacts fast to their input. Because numerous users can access data, execute apps, and do different activities at once, user productivity is boosted.

5) Resource Multiplexing: Time-sharing systems divide up resources like CPU, memory, and I/O devices among several users in an efficient manner. Because of the effective utilization of resources made possible by this, the system can accommodate more users without requiring a corresponding increase in hardware.