CHAPTER 1

Introduction of Operating System

Topics:-

1. **Fundamental Goals of Operating system**
2. **Overview of Operating Systems**
    a. **Multi programming**
    b. **Time Sharing**
    c. **Real Time**
    d. **Multithreading**
    e. **Distributed**
3. **Operating System services**
4. **Case Study**
    a. **Linux**
    b. **Latest Windows Operating System**
5. **Generations of Operating System**

1. Fundamental Goals of Operating system

=> The fundamental goals of an operating system (OS) are to manage and coordinate the hardware and software resources of a computer system to ensure efficient and effective operation. These goals include:

1. **Resource Management**: Efficiently manage hardware resources such as the CPU, memory, storage, and I/O devices. This includes allocating resources to various processes and ensuring that they are used efficiently.
2. **Process Management**: Handle the creation, scheduling, and termination of processes. This involves managing process states, ensuring proper execution of processes, and handling process synchronization and communication.
3. **Memory Management**: Manage the computer's memory, including RAM and virtual memory. This involves allocating memory to processes, handling memory protection, and managing memory swapping between RAM and disk storage.
4. **File System Management**: Provide a way to store, organize, and access files on storage devices. This includes managing file permissions, directories, and file operations (e.g., reading, writing, and deleting files).
5. **Device Management**: Control and manage input and output devices (e.g., keyboards, printers, disk drives). This includes providing device drivers and handling communication between the OS and hardware devices.
6. **User Interface**: Offer a user interface (UI) for users to interact with the computer system. This can be a command-line interface (CLI) or a graphical user interface (GUI), depending on the OS.
7. **Security and Protection**: Ensure the security of the system by protecting against unauthorized access and ensuring that users' data and system resources are protected from malicious activities. This includes implementing authentication, authorization, and encryption mechanisms.
8. **Error Handling**: Detect, report, and handle errors that occur during operation, whether they are hardware failures, software bugs, or user errors.
9. **System Performance**: Optimize the performance of the system by managing resource allocation, balancing loads, and minimizing delays and overhead.

2. Overview of Operating Systems

a. Multi programming

=>

Multiprogramming is a technique used in operating systems to increase CPU utilization by running multiple processes concurrently. The key idea is to keep the CPU busy by having multiple processes in memory, so that while one process is waiting for I/O operations, the CPU can switch to another process that is ready to execute.

Key Concepts

1. **Objective**: The primary goal of multiprogramming is to maximize CPU usage and system throughput by minimizing idle time. By having several processes in memory, the system can perform useful work while waiting for I/O operations to complete.
2. **Process Scheduling**: In a multiprogramming environment, the operating system uses a scheduler to manage the execution of processes. The scheduler selects processes from a queue and allocates CPU time to them.
3. **Context Switching**: When the CPU switches from one process to another, the operating system saves the state of the current process (context) and loads the state of the next process. This is known as context switching and allows multiple processes to share the CPU.
4. **Memory Management**: Multiprogramming requires efficient memory management to ensure that multiple processes can reside in memory simultaneously. Techniques like paging, segmentation, and virtual memory are used to manage memory.

   Advantages:

   - **Increased CPU Utilization**: By overlapping CPU and I/O operations, multiprogramming increases overall system efficiency.
   - **Improved Throughput**: More processes can be completed in a given time period.
   - **Better Resource Utilization**: System resources are used more effectively, reducing idle time.

Disadvantages:

- **Complexity**: Managing multiple processes and ensuring proper resource allocation adds complexity to the operating system.
- **Overhead**: Context switching and memory management can introduce overhead, reducing system performance if not managed efficiently.
- **Resource Contention**: Processes may compete for limited resources, leading to potential bottlenecks and reduced performance.

b. Time Sharing

=>

Time Sharing (also known as time-sharing) is a method used in operating systems to enable multiple users or processes to share a single computer system efficiently. The core idea is to allow multiple users to interact with the system simultaneously by rapidly switching between them, giving the illusion that each user has their own dedicated machine.

Key Concepts

1. **Objective**: The main goal of time-sharing is to provide interactive computing capabilities, allowing multiple users to access and use the system concurrently. This improves system utilization and response time for interactive applications.
2. **Time Slices**: The operating system allocates a small, fixed amount of CPU time to each process or user, known as a time slice or quantum. After the time slice expires, the CPU is switched to the next process or user.
3. **Context Switching**: Time-sharing involves frequent context switching, where the operating system saves the state of the currently running process and loads the state of the next process. This rapid switching allows multiple processes to appear to be running simultaneously.
4. **Scheduler**: The scheduler in a time-sharing system is responsible for managing the allocation of CPU time slices to different processes or users. Scheduling algorithms such as Round Robin are commonly used to ensure fairness and efficiency.
5. **Interactivity**: Time-sharing systems are designed to support interactive applications, where users can input commands and receive immediate feedback. This contrasts with batch processing systems, where jobs are processed sequentially with no user interaction during execution.

Advantages:

- **Interactive Computing**: Users can interact with the system in real-time, which is ideal for applications like text editors, web browsers, and online services.
- **Efficient Resource Utilization**: By switching between users or processes, time-sharing maximizes CPU usage and system throughput.
- **Fairness**: Time-sharing systems aim to provide equal access to the CPU for all users, reducing the likelihood of any single user monopolizing system resources.

Disadvantages:

- **Overhead**: Frequent context switching introduces overhead, which can impact overall system performance.
- **Complexity**: Managing multiple users and processes, ensuring fairness, and handling interactive input/output adds complexity to the operating system.
- **Resource Contention**: Users may compete for limited resources, potentially leading to contention and reduced performance.

c. Real Time

=> **Real-Time Systems** are designed to provide timely and deterministic responses to events or inputs. They are critical in environments where the correctness of an operation depends not only on the logical correctness but also on the timing of the operation. Real-time systems are commonly used in applications where delays can lead to system failure or unsafe conditions.

Key Concepts

1. **Objective**: The primary goal of real-time systems is to ensure that critical tasks are completed within a specified time frame. The system must provide predictable and consistent response times to meet the timing constraints of real-time applications.
2. **Types of Real-Time Systems**:
    - **Hard Real-Time Systems**: Must meet their deadlines absolutely; failure to do so could lead to catastrophic results. Examples include medical devices, aircraft control systems, and industrial automation systems.

- **Soft Real-Time Systems**: Can tolerate occasional deadline misses without severe consequences. Examples include multimedia systems, online transaction processing, and video streaming.

Advantages:

- **Predictability**: Real-time systems provide predictable and deterministic behavior, ensuring that tasks are completed within specified time constraints.
- **Safety and Reliability**: Essential for systems where timing is crucial for safety and reliability.

Disadvantages:

- **Complexity**: Designing and implementing real-time systems is complex due to the need to meet strict timing constraints and handle real-time scheduling.
- **Overhead**: Real-time scheduling and management can introduce overhead, potentially affecting system performance.

d. **Multithreading**

=> **Multithreading** is a programming concept where a single process is divided into multiple threads, each capable of executing independently. These threads share the same process resources but run concurrently, allowing for more efficient use of CPU and improved application performance.

Key Concepts

1. **Objective**: The primary goal of multithreading is to improve application performance and responsiveness by allowing multiple operations to run concurrently within a single process. This can lead to better resource utilization and faster execution of tasks.
2. **Threads**: A thread is the smallest unit of execution within a process. Threads within the same process share the same memory space and resources but have their own execution context, including registers, program counter, and stack.

Advantages:

- **Improved Performance**: Multithreading can lead to better performance by utilizing CPU resources more effectively and performing multiple tasks simultaneously.
- **Responsiveness**: Multithreading can improve the responsiveness of applications by allowing background tasks to run concurrently with the main application thread.
- **Resource Sharing**: Threads within the same process share resources, making communication between threads more efficient.

Disadvantages:

- **Complexity**: Multithreaded programs are more complex to design and debug due to issues like synchronization, race conditions, and deadlocks.
- **Overhead**: Creating and managing threads introduces overhead, which can affect system performance.
- **Potential for Bugs**: Issues such as deadlocks, livelocks, and race conditions can arise, leading to difficult-to-trace bugs.

e. Distributed

=> **Distributed Systems** are a collection of independent computers that appear to their users as a single coherent system. These systems collaborate to achieve a common goal, often by sharing resources, data, and processing tasks across multiple machines or nodes.

Key Concepts

1. **Objective**: The primary goal of distributed systems is to provide a unified and reliable computing environment by distributing tasks and resources across multiple machines. This can improve performance, scalability, reliability, and fault tolerance.

   Advantages:

   - **Scalability**: Ability to handle growing workloads by adding more nodes or resources.
   - **Fault Tolerance**: Increased reliability and availability through redundancy and replication.
   - **Resource Sharing**: Efficient use of distributed resources and data across multiple nodes.

Disadvantages:

- **Complexity**: Increased complexity in design, implementation, and management due to distributed nature.
- **Latency**: Potential for increased communication latency between nodes.
- **Consistency**: Challenges in maintaining data consistency and handling conflicts.

3. Operating System services

=> The operating system provides an environment for the execution of programs. The operating system provides services to programs to the users of those programs .The operating system services are provided for the convenience of the programmer.

Operating system provides services to the user program and system.

For user point of view services:

1. User Interface: it provides interface to user so user can perform some action. It provides three types of interfaces,

a) Command line Interface (CLI), in which user can type command and methods for executing them.

b) Batch interface, in which commands and directives to control those commands are entered into files ,and those files are executed.

c) Graphical User Interface (GUI),in which interface is a window system with pointing device to direct I/O ,chooses from menus, keyboard to enter text.

2. Program execution: The system must be able to load a program into memory and run it. The program must be able to end its execution either normally or abnormally (indicating error).

3. I/O operations: A running program may require I/O involves a file or an I/O device. For specific devices, special functions may be desired. For efficiency protection , users usually cannot control I/O devices directly. The operating system provides a means to do I/O.

4. File-system manipulation: The programs need to read and write files and programs must also create and delete files. The operating system maintains the file system.

5. Communications: In many conditions the process needs to exchange information with another process. Such communications can occur in two ways:

6. Error Detection: The operating system constantly needs to be aware of possible errors. Errors may occur in the CPU and memory hardware, in I/O devices and in user programs. For each type of errors, the operating system should take the proper action to ensure correct and consistent computing.

For System point of view services

1. **Resource allocation:** Operating system manages many resources like CPU cycles, main memory, and file storage. Resources are allocated to multiple users. Some resources have special allocation code, whereas others may have general request and release code.
2. **Accounting:** The tracking of which user use how many wnd which kind of computer resources can be used for billing or simply accumulating usage statistics. Usage statistics may be a valuable tool for researchers who wish to reconfigure the system to improve computing services
3. **Protection:** Protection involves ensuring that all access to system resources is controlled

4. Case Study

a. Linux

b. Latest Windows Operating System

=>

1. Linux: Linux is a family of free and open-source operating systems based on the Linux kernel. Operating systems based on Linux are known as Linux distributions or distros. Examples include Debian, Ubuntu, Fedora, CentOS, Gentoo, Arch Linux, and many others.

The Linux kernel has been under active development since 1991, and has proven to be extremely versatile and adaptable. You can find computers that run Linux in a wide variety of contexts all over the world, from web servers to cell phones. Today, 90% of all cloud infrastructure and 74% of the world’s smartphones are powered by Linux.

However, newcomers to Linux may find it somewhat difficult to approach, as Linux filesystems have a different structure than those found on Windows or MacOS. Additionally, Linux-based operating systems depend

heavily on working with the command line interface, while most personal computers rely on graphical interfaces.

This guide serves as an introduction to important.

The terms “terminal,” “shell,” and “command line interface” are often used interchangeably, but there are subtle differences between them:

A terminal is an input and output environment that presents a text-only window running a shell.

A shell is a program that exposes the computer’s operating system to a user or program. In Linux systems, the shell presented in a terminal is a command line interpreter.

A command line interface is a user interface (managed by a command line interpreter program) which processes commands to a computer program and outputs the results

b. Latest Windows Operating System Windows 11

is the latest major release of Microsoft's Windows NT operating system, released in October 2021. It is a free upgrade to its predecessor, Windows 10 (2015), and is available for any Windows 10 devices that meet the new Windows 11 system requirements.

5. Generations of Operating System

=>

The First Generation (1940 to early 1950s)

When the first electronic computer was developed in 1940, it was created without any operating system. In early times, users have full access to the computer machine and write a program for each task in absolute machine language. The programmer can perform and solve only simple mathematical calculations during the computer generation, and this calculation does not require an operating system.

The Second Generation (1955 - 1965)

The first operating system (OS) was created in the early 1950s and was known as GMOS. General Motors has developed OS for the IBM computer. The second-generation operating system was based on a single stream batch processing system because it collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine.

The Third Generation (1965 – 1980)

During the late 1960s, operating system designers were very capable of developing a new operating system that could simultaneously perform multiple tasks in a single computer program called multiprogramming. The introduction of multiprogramming plays a very important role in developing operating systems that allow a CPU to be busy every time by performing different tasks on a computer at the same time.

The Fourth Generation (1980 - Present Day)

The fourth generation of operating systems is related to the development of the personal computer. However, the personal computer is very similar to the minicomputers that were developed in the third generation. The cost of a personal computer was very high at that time; there were small fractions of minicomputers costs. Currently, most Windows users use the Windows 10 operating system. Besides the Windows operating system, Apple is another popular operating system built in the 1980s, and this operating system was developed by Steve Jobs, a co-founder of Apple. They named the operating system Macintosh OS or Mac OS.