NEW L J INSTITUTE OF ENGINEERING & TECHNOLOGY

SEMESTER- 4

BRANCH- CSE/CSE[AIML]

SUBJECT NAME: OPERATING SYSTEM

SUBJECT CODE: 3140702

GTU SYLLABUS

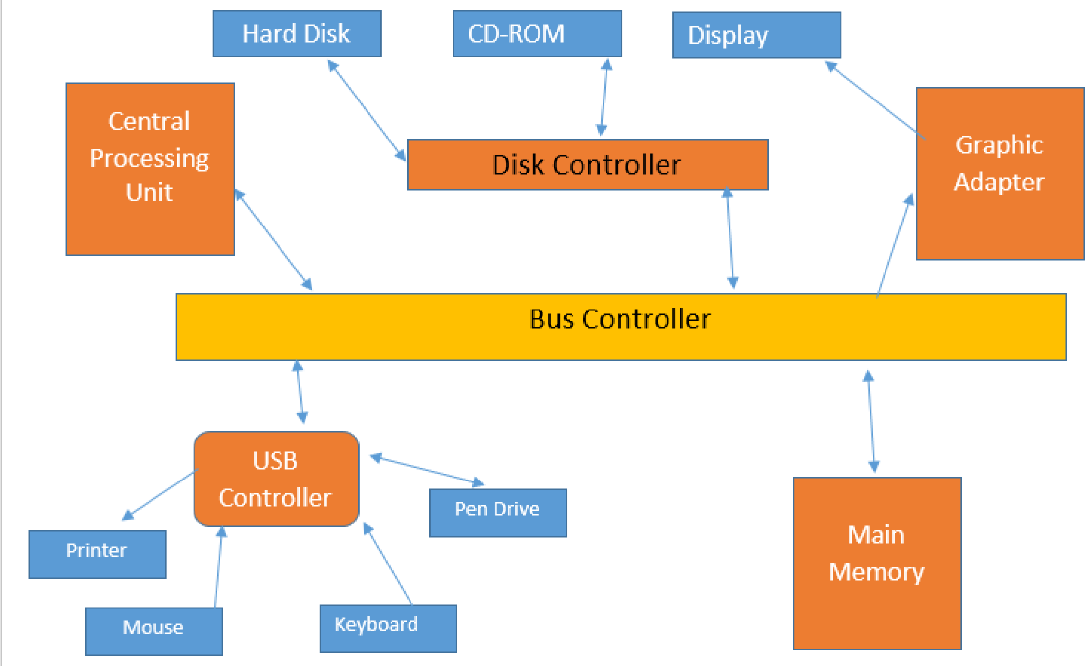
Sr. No.	Content	% Weightage
1	**Introduction:** Computer system overview, Architecture, Goals & Structures of O.S, Basic functions, Interaction of O.S. & hardware architecture, System calls, Batch, multiprogramming. Multitasking, time sharing, parallel, distributed & realtime O.S.	10
2	**Process and Threads Management:** Process Concept, Process states, Process control, Threads, Uni-processor Scheduling: Types of scheduling: Preemptive, Non preemptive, Scheduling algorithms: FCFS, SJF, RR, Priority, Thread Scheduling, Real Time Scheduling. System calls like ps, fork, join, exec family, wait.	15
3	**Concurrency:** Principles of Concurrency, Mutual Exclusion: S/W approaches, H/W Support, Semaphores, Pipes, Message Passing, Signals, Monitors.	8
4	**Inter Process Communication:** Race Conditions, Critical Section, Mutual Exclusion, Hardware Solution, Strict Alternation, Peterson's Solution, The Producer Consumer Problem, Semaphores, Event Counters, Monitors, Message Passing, Classical IPC Problems: Reader's & Writer Problem, Dinning Philosopher Problem etc., Scheduling, Scheduling Algorithms.	15
5	**Deadlock:** Principles of Deadlock, Starvation, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, System calls	8
6	**Memory Management:** Memory Management requirements, Memory partitioning: Fixed and Variable Partitioning, Memory Allocation: Allocation Strategies (First Fit, Best Fit, and Worst Fit), Swapping, Paging and Fragmentation. Demand Paging, Security Issues. Virtual Memory: Concepts, VM management, Page Replacement Policies (FIFO, LRU, Optimal, Other Strategies), Thrashing.	15
7	**I/O Management & Disk scheduling:** I/O Devices, Organization of I/O functions, Operating System Design issues, I/O Buffering, Disk Scheduling (FCFS, SCAN, C-SCAN, SSTF), RAID, Disk Cache.	10
8	**Security & Protection:** Security Environment, Design Principles Of Security, User Authentication, Protection Mechanism : Protection Domain, Access Control List	7
9	**Unix/Linux Operating System:** Development Of Unix/Linux, Role & Function Of Kernel, System Calls, Elementary Linux command & Shell Programming, Directory Structure, System Administration Case study: Linux, Windows Operating System	7
10	**Virtualization Concepts:** Virtual machines; supporting multiple operating systems simultaneously on a single hardware platform; running one operating system on top of another. True or pure virtualization.	5

UNIT 1: INTRODUCTION

Unit 1	**Introduction:** Computer system overview, Architecture, Goals & Structures of O.S, Basic functions, Interaction of O.S. & hardware architecture, System calls, Batch, multiprogramming. Multitasking, time sharing, parallel, distributed & realtime O.S.

TOPIC 1: Computer system overview

COMPUTER SYSTEM OVERVIEW



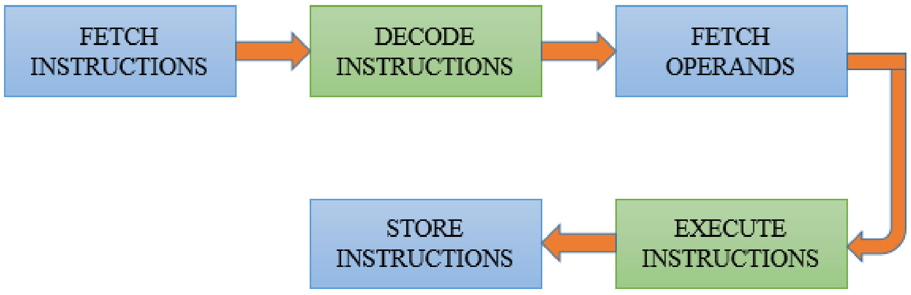
1. It consists of hardware device and software that are combined to provide a tool to user for solving problems
2. it consists of CPU, memory I/O devices with one or more modules of each type this all components are interconnected
3. common bus is used for communication between these devices , each devices has its own device controller.
4. CPU and device controller use memory cycle for execution purpose ,but memory cycle is only available to one device at a time
5. Bootstrap program is loaded when user start the computer . it initializes all the device connected to the computer system and the loads required device drivers.
6. OS loads the computer system , in UNIX "init" is first process which execute by OS.
7. Interrupt is software and hardware . it used to send signal to CPU . software interrupt is sometime call System call.
8. when interrupt trigger , the CPU stops executing the instruction
9. Processor access the data from main memory (RAM)
10. every device uses a device controller to connect it to the computer address anad data bus .
11. storage devices are used to store data while the computer is off. Device controller manage the data transfer between device .

Main elements are as follow:

1. **CPU** : it controls the operation of the computer . It performs data processing function
2. **Main Memory:** Used for storing programs and data . this memory is typically volatile . can not stored memory permanent.
   Refer as primary memory or real memory .user program and data are stored in it

3. **I/O modules**: used for moving data between computer and its external environment. these environment consists of variety of devices , including secondary memory devices , communication equipment and terminals.
4. **System bus**: it provide communication among processors, main memory and I/O modules.

INSTRUCTION EXECUTION



1. **Instruction Fetch**

The working device fetches the subsequent practice from this system's reminiscence space and masses it into the CPU's coaching check-in. The practice check-in holds the presently executing guidance.

2. **Instruction Decoding**

The CPU decodes the fetched instruction to decide its type and the operations it calls for. The working system assists in this procedure by presenting the necessary interpreting mechanisms and education sets supported through the CPU architecture.

3. **Operand Fetch**

If the education requires accessing records or operands from memory or registers, the operating gadget ensures that the vital information is retrieved and made to be had to the CPU. This may also involve reminiscence access operations or check-in transfers.

4. **Instruction Execution**

The CPU executes the decoded guidance based at the instruction set structure (ISA) supported with the aid of the working device. The operating machine provides the desired resources and services to permit proper preparation execution, including managing interrupts, scheduling, and resource allocation.

5. **Result Storage**

After the instruction is finished, the working gadget handles storing the effects lower back to an appropriate memory region or register. This ensures that this system's country is updated efficaciously for subsequent instructions.

6. **Instruction Pointer Update**

The operating device manages this system's execution flow by means of updating the guidance pointer or software counter. This factors to the next coaching to be fetched and executed, permitting this system to progress in its execution.

7. **Exception Handling**

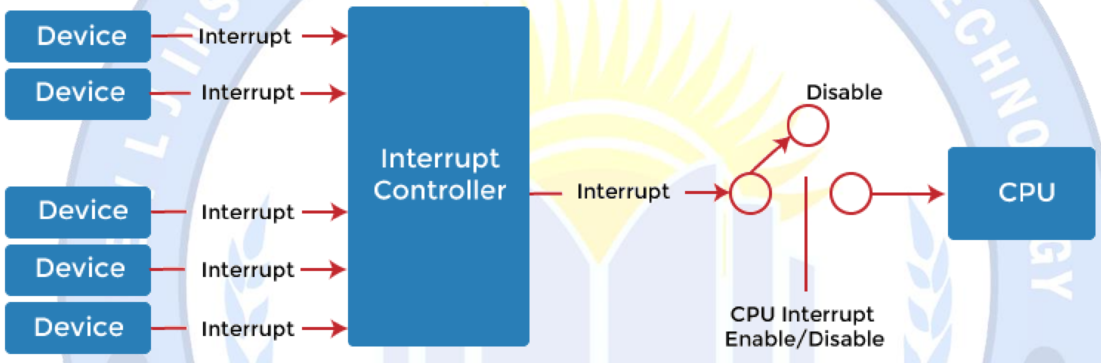
During education execution, diverse first-rate situations may additionally occur, including mathematics errors, illegal instructions, or access violations. The operating device handles these exceptions by way of interrupting the everyday execution float, moving control to the best exception handler, and taking the important actions to deal with the exception and hold device stability.

Question: What is interrupt? How it is handle by operating system

Solution:

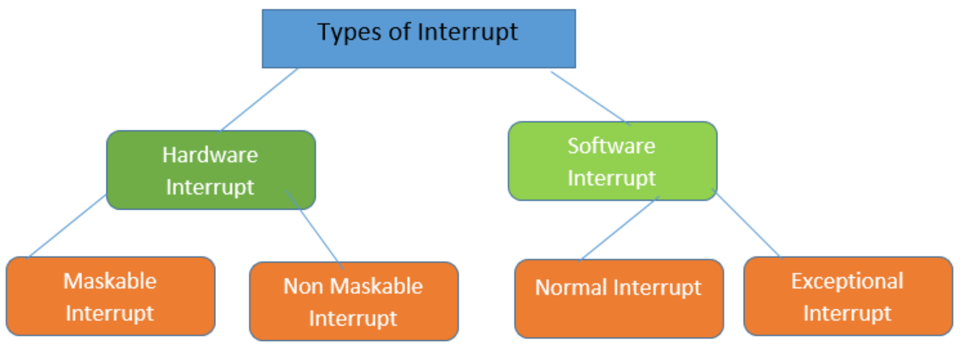
INTERRUPTS
Definition: An interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process.

An operating system usually has some code that is called an interrupt handler. The interrupt handler prioritizes the interrupts and saves them in a queue if more than one is waiting to be handled. The operating system has another little program called a scheduler that figures out which program to control next.



Types of Interrupt

Interrupt signals may be issued in response to hardware or software events. These are classified as hardware interrupts or software interrupts, respectively.



Hardware interrupts

The interrupt signal generated from external devices and i/o devices are made interrupt to CPU when the instructions are ready.

For example − In a keyboard if we press a key to do some action this pressing of the keyboard generates a signal that is given to the processor to do action, such interrupts are called hardware interrupts.

Hardware interrupts are classified into two types which are as follows −

Maskable Interrupt − The hardware interrupts that can be delayed when a highest priority interrupt has occurred to the processor.
Non Maskable Interrupt − The hardware that cannot be delayed and immediately be serviced by the processor.

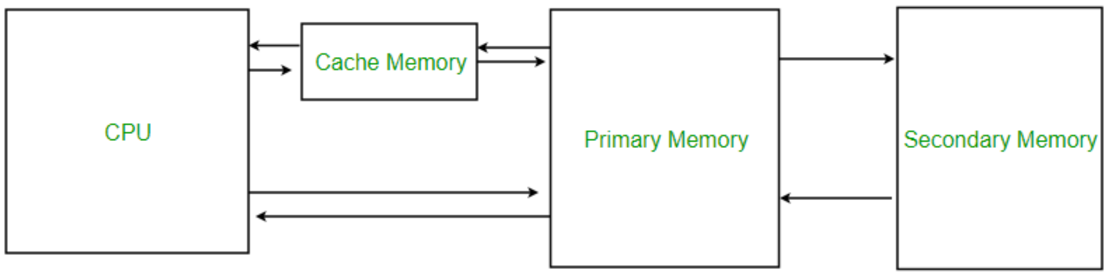
Software interrupts

The interrupt signal generated from internal devices and software programs need to access any system call then software interrupts are present.

Software interrupt is divided into two types. They are as follows −

Normal Interrupts − The interrupts that are caused by the software instructions are called software instructions.
Exception − Exception is nothing but an unplanned interruption while executing a program. For example − while executing a program if we got a value that is divided by zero is called an exception.

CACHE MEMORY



Cache Memory is a special very high-speed memory. The cache is a smaller and faster memory that stores copies of the data from frequently used main memory locations. There are various different independent caches in a CPU, which store instructions and data. The most important use of cache memory is that it is used to reduce the average time to access data from the main memory.
Levels of Memory

- Level 1 or Register: It is a type of memory in which data is stored and accepted that are immediately stored in the CPU. The most commonly used register is Accumulator, Program counter, Address Register, etc.
- Level 2 or Cache memory: It is the fastest memory that has faster access time where data is temporarily stored for faster access.
- Level 3 or Main Memory: It is the memory on which the computer works currently. It is small in size and once power is off data no longer stays in this memory.
- Level 4 or Secondary Memory: It is external memory that is not as fast as the main memory but data stays permanently in this memory.

Cache Performance

- When the processor needs to read or write a location in the main memory, it first checks for a corresponding entry in the cache.
- If the processor finds that the memory location is in the cache, a Cache Hit has occurred and data is read from the cache.
- If the processor does not find the memory location in the cache, a cache miss has occurred. For a cache miss, the cache allocates a new entry and copies in data from the main memory, then the request is fulfilled from the contents of the cache.
- The performance of cache memory is frequently measured in terms of a quantity called Hit ratio.
- Performance of cache is measured by the number of cache hits to the number of searches. This parameter of measuring performance is known as the Hit Ratio.
- Hit ratio=(Number of cache hits)/(Number of searches)

Benefits of Cache Memory

Various benefits of the cache memory are,

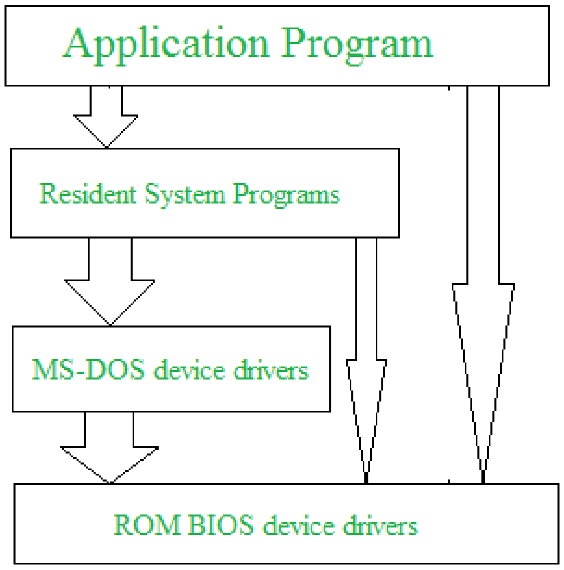
1. Faster access: Faster than main memory. It resides closer to CPU , typically on same chip or in close proximity. Cache stores subset of data and instruction.
2. Reducing memory latency: Memory access latency refers to time taken for processes to retrieve data from memory. Caches are designed to exploit principle of locality.
3. Lowering bus traffic: Accessing data from main memory involves transferring it over system bus. Bus is shared resource and excessive traffic can lead to congestion and slower data transfers. By utilizing cache memory , processor can reduce frequency of accessing main memory resulting in less bus traffic and improves system efficiency.
4. Increasing effective CPU utilization: Cache memory allows CPU to operate at a higher effective speed. CPU can spend more time executing instruction rather than waiting for memory access. This leads to better utilization of CPU's processing capabilities and higher overall system performance.
5. Enhancing system scalability: Cache memory helps improve system scalability by reducing impact of memory latency on overall system performance.

Characteristics of Cache Memory

1. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU.
2. Cache Memory holds frequently requested data and instructions so that they are immediately available to the CPU when needed.
3. Cache memory is costlier than main memory or disk memory but more economical than CPU registers.
4. Cache Memory is used to speed up and synchronize with a high-speed CPU.

TOPIC 2: Architecture

1. SIMPLE STRUCTURE/MONOLITHIC STRUCTURE



Such operating systems do not have well-defined structures and are small, simple, and limited.
The interfaces and levels of functionality are not well separated.
MS-DOS is an example of such an operating system.
In MS-DOS, application programs are able to access the basic I/O routines. These types of operating systems cause the entire system to crash if one of the user programs fails.
It consists of the following layers:

1. Application Program Layer
2. System program layer for resident program
3. Device Drive Layer
4. ROM BIOS device driver layer

Advantages of Monolithic Architecture

- Fast: Monolithic operating systems are fast. Thus, they provide better process scheduling, memory management, file management, etc.
- Direct Interaction between Components: All the components and the kernel can directly interact. It also helps in gaining a better speed.
- Easy and Simple: Its structure is easy and simple as all the components are located in the same address space.
- Better for Smaller Tasks: Monolithic OS works better for handling smaller tasks.

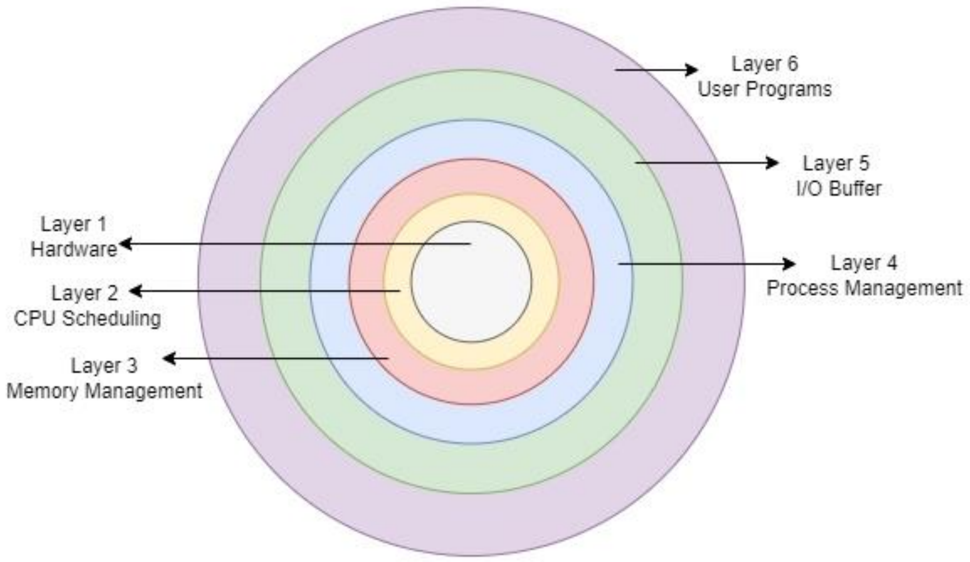
Disadvantages of Monolithic Architecture

- Prone to Errors: Monolithic OS can generate errors and bugs in the system. It is because user programs use the same address space as the kernel.
- Difficult to Update: In a monolithic OS, all the OS code is in a single big chunk; therefore, it is difficult to add or remove features in the OS.
- Not Portable: The code written in a monolithic OS is difficult to carry with or transfer to another system: this is because all the code works in a big chunk only, and you have to move it all.

Question: Explain Layered Operating System Structure

Solution:

2. LAYERED APPROACH



An OS can be broken into pieces and retain much more control over the system.
In this structure, the OS is broken into a number of layers (levels).
The bottom layer (layer 0) is the hardware, and the topmost layer (layer N) is the user interface.
These layers are so designed that each layer uses the functions of the lower-level layers.
This simplifies the debugging process, if lower-level layers are debugged and an error occurs during debugging, then the error must be on that layer only, as the lower-level layers have already been debugged.

Advantages of Layered Architecture

- Modularity: This architecture promotes modularity because each layer only does its assigned duties.
- Easy debugging: It is relatively simple to debug because the layers are discrete. If a mistake happens in the CPU scheduling layer, the developer can only debug that layer, as opposed to a Monolithic system where all services are present at the same time.
- Easy update: A change made to one layer will have no effect on the other layers.
- No direct access to hardware: The hardware layer is the design's innermost layer. So, unlike the Simple system, where the user has direct access to the hardware, a user can use hardware services but not directly modify or access it.
- Abstraction: Every layer is focused on its own set of tasks. As a result, the other layers' functions and implementations are abstract.

Disadvantages of Layered Architecture

- Complex and careful implementation: Because a layer can use the services of the levels below it, the layers must be carefully arranged. The memory management layer, for example, is used by the backup storage layer. As a result, it must be placed behind the memory management layer. As a result, significant modularity leads to complicated implementation.
- Slower in execution: When a layer wishes to communicate with another layer, it sends a request that must travel through the layers between the two layers to be fulfilled. As a result, unlike the Monolithic system, which is faster, it increases response time. As an output, increasing the number of layers may result in an inefficient design.

Examples of Layered-Based Operating Systems:

- UNIX

- THEOS
- VMS

Question: What is the function of Kernel

Solution:

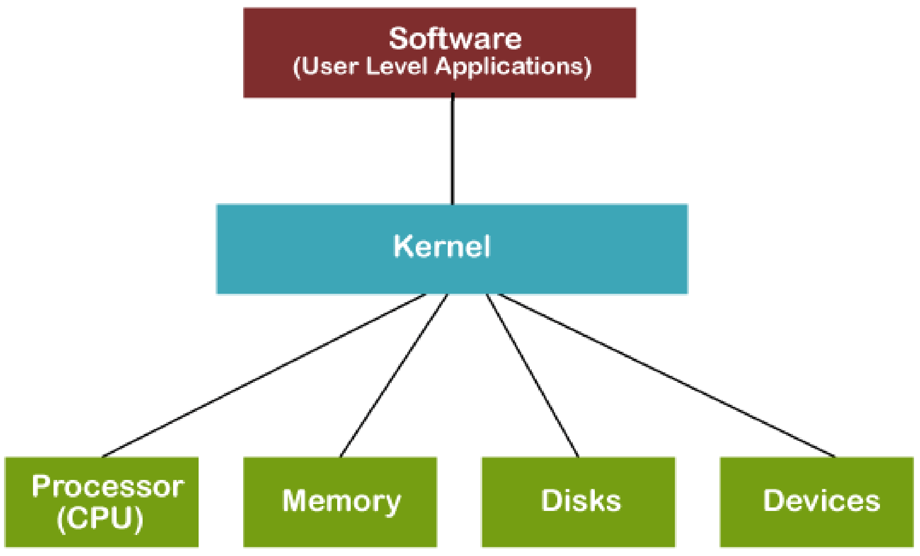
KERNAL
In computer science, Kernel is a computer program that is a core or heart of an operating system.

- As discussed above, Kernel is the core part of an OS(Operating system); hence it has full control over everything in the system. Each operation of hardware and software is managed and administrated by the 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 the part of the OS that always resides in computer memory and enables the communication between software and hardware components.
- It is the computer program that first loaded on start-up the system (After the bootloader). Once it is loaded, it manages the remaining start-ups. 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 separate memory space, known as protected Kernel space. It is protected from being accessed by application programs or less important parts of OS.
- Other application programs such as browser, word processor, audio & video player 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 and slowness.

Objectives of Kernel :

- To establish communication between user level application and hardware.
- To decide state of incoming processes.
- To control disk management.
- To control memory management.
- To control task management.

○

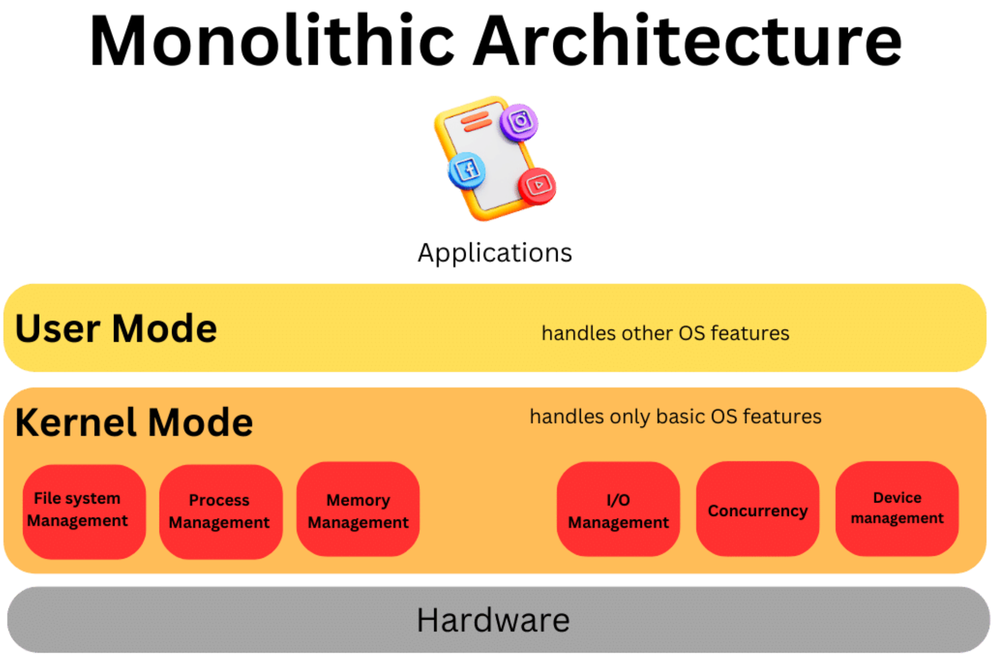


Question: Sketch and describe monolithic operating system structure

Solution:

Types of Kernel

1. Monolithic Kernel –



A monolithic operating system architecture includes all system functionalities in a single module, with the kernel and device drivers as key components. In this architecture, the operating system as a whole is working in the kernel space. The kernel directly controls all the file, memory, device, and process management.
In a monolithic kernel, the same memory space is used to implement user services and kernel services.

It means, in this type of kernel, there is no different memory used for user services and kernel services.

As it uses the same memory space, the size of the kernel increases, increasing the overall size of the OS.

The execution of processes is also faster than other kernel types as it does not use separate user and kernel space.

Examples of Monolithic Kernels are Unix, Linux, Open VMS, XTS-400, etc.

Advantages of Monolithic Architecture

- Fast: Monolithic operating systems are fast. Thus, they provide better process scheduling, memory management, file management, etc.
- Direct Interaction between Components: All the components and the kernel can directly interact. It also helps in gaining a better speed.
- Easy and Simple: Its structure is easy and simple as all the components are located in the same address space.
- Better for Smaller Tasks: Monolithic OS works better for handling smaller tasks.

Disadvantages of Monolithic Architecture

- Prone to Errors: Monolithic OS can generate errors and bugs in the system. It is because user programs use the same address space as the kernel.
- Difficult to Update: In a monolithic OS, all the OS code is in a single big chunk; therefore, it is difficult to add or remove features in the OS.
- Not Portable: The code written in a monolithic OS is difficult to carry with or transfer to another system: this is because all the code works in a big chunk only, and you have to move it all.

Examples of Monolithic-Based Operating Systems:

- Windows
- Linux
- MacOS

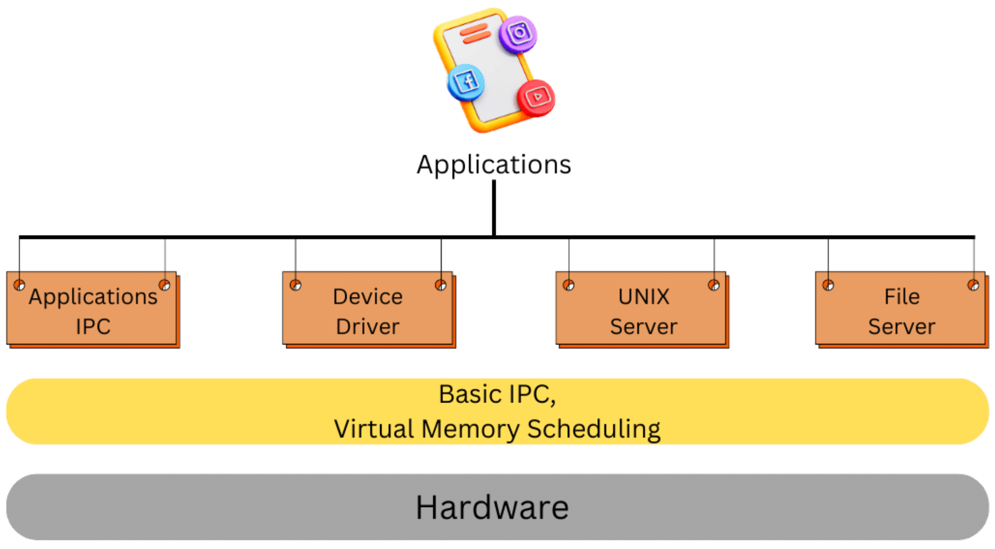
Question: Explain the microkernel system architecture in detail.

Solution:

Microkernel Architecture

The microkernel architecture includes only the major functions like memory and process management in the kernel. In this architecture, file management and input-output management are included in the user space.

Microkernel Architecture



Advantages of Microkernel Architecture

- Performance: Microkernel architecture is compact and isolated, it can perform better.
- Security: Microkernels are safe since only those components are provided that would otherwise disturb the system's functionality.
- Scalable: It is easily expandable as compared to a monolithic kernel.
- Modularity: Microkernels are modular, and the various modules can be swapped out, reloaded, and modified without affecting the Kernel. When compared to monolithic systems, there are fewer system crashes.The Microkernel interface aids in the implementation of a more modular system structure.

Disadvantages of Microkernel Architecture

- Performance overhead: Performance overhead arises due to the need for inter-process communication between kernel modules. A context switch or a function call is required when the drivers are implemented as procedures or processes.
- Limited hardware support: The microkernel architecture can be more difficult to port to new hardware platforms.
- Higher cost: Compared to a monolithic system, providing services in a Microkernel system is costly.

Examples of Microkernel-Based Operating Systems:

- MINIX
- QNX
- L4 microkernel
- GNU Hurd

Question: Differentiate between Monolithic Kernel and Micro Kernel

Solution:

Monolithic Kernel	Microkernel
In Monolithic kernel architecture, all system services, including device drivers, run in kernel space.	In Microkernel architecture, only essential services, such as scheduling and interprocess communication, run in kernel space. Device drivers run in user space.
Functions are tightly coupled and share a common memory space.	Functions are loosely coupled and communicate via message passing, which incurs some overhead.
System calls are fast, but errors in one component can crash the entire system.	System calls are slower due to message passing, but errors in one component do not necessarily crash the entire system.
Less secure because a bug or vulnerability in one component can affect the entire system.	More secure because if one component is compromised, the damage is contained to that component.
Examples include Linux, FreeBSD, and Windows.	Examples include QNX, L4, and MINIX.

TOPIC 3: Goals & Structures of O.S

Question: What is an Operating System? Give functions of Operating Systems.

Solution:

OS DEFINATION:

Operating system is a program that control the execution of application programs. It is an interface between application and hardware.

Operating System lies in the category of system software. It basically manages all the resources of the computer. An operating system acts as an interface between the software and different parts of the computer or the computer hardware. The operating system is designed in such a way that it can manage the overall resources and operations of the computer.

Functions of the Operating System

- Resource Management: The operating system manages and allocates memory, CPU time, and other hardware resources among the various programs and processes running on the computer.
- Process Management: The operating system is responsible for starting, stopping, and managing processes and programs. It also controls the scheduling of processes and allocates resources to them.
- Memory Management: The operating system manages the computer's primary memory and provides mechanisms for optimizing memory usage.

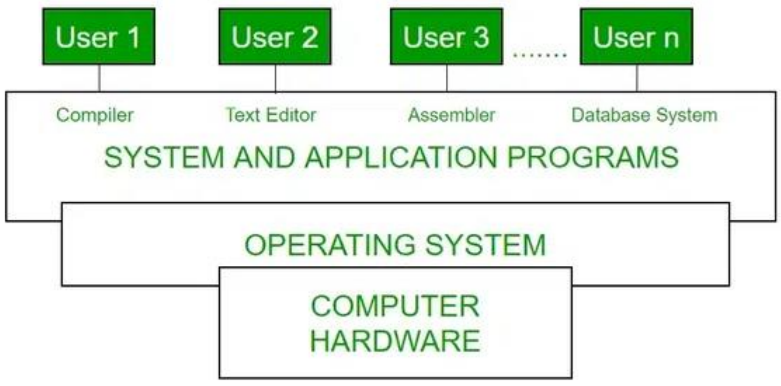
- Security: The operating system provides a secure environment for the user, applications, and data by implementing security policies and mechanisms such as access controls and encryption.
- Job Accounting: It keeps track of time and resources used by various jobs or users.
- File Management: The operating system is responsible for organizing and managing the file system, including the creation, deletion, and manipulation of files and directories.
- Device Management: The operating system manages input/output devices such as printers, keyboards, mice, and displays. It provides the necessary drivers and interfaces to enable communication between the devices and the computer.
- Networking: The operating system provides networking capabilities such as establishing and managing network connections, handling network protocols, and sharing resources such as printers and files over a network.
- User Interface: The operating system provides a user interface that enables users to interact with the computer system. This can be a Graphical User Interface (GUI), a Command-Line Interface (CLI), or a combination of both.
- Backup and Recovery: The operating system provides mechanisms for backing up data and recovering it in case of system failures, errors, or disasters.
- Virtualization: The operating system provides virtualization capabilities that allow multiple operating systems or applications to run on a single physical machine. This can enable efficient use of resources and flexibility in managing workloads.
- Performance Monitoring: The operating system provides tools for monitoring and optimizing system performance, including identifying bottlenecks, optimizing resource usage, and analyzing system logs and metrics.
- Time-Sharing: The operating system enables multiple users to share a computer system and its resources simultaneously by providing time-sharing mechanisms that allocate resources fairly and efficiently.
- Error-detecting Aids: These contain methods that include the production of dumps, traces, error messages, and other debugging and error-detecting methods.

Objectives of Operating Systems

- Convenient to use: One of the objectives is to make the computer system more convenient to use in an efficient manner.
- User Friendly: To make the computer system more interactive with a more convenient interface for the users.
- Easy Access: To provide easy access to users for using resources by acting as an intermediary between the hardware and its users.
- Management of Resources: For managing the resources of a computer in a better and faster way.
- Controls and Monitoring: By keeping track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- Fair Sharing of Resources: Providing efficient and fair sharing of resources between the users and programs.

STRUCTURE OF OS

1. OS AS USER INTERFACE



The process operating system as User Interface:

1. User
2. System and application programs
3. Operating system
4. Hardware

Every general-purpose computer consists of hardware, an operating system(s), system programs, and application programs. The hardware consists of memory, CPU, ALU, I/O devices, peripheral devices, and storage devices. The system program consists of compilers, loaders, editors, OS, etc. The application program consists of business programs and database programs.

Every computer must have an operating system to run other programs. The operating system coordinates the use of the hardware among the various system programs and application programs for various users. It simply provides an environment within which other programs can do useful work.

The operating system is a set of special programs that run on a computer system that allows it to work properly. It performs basic tasks such as recognizing input from the keyboard, keeping track of files and directories on the disk, sending output to the display screen, and controlling peripheral devices.

Purposes of an Operating System

- It controls the allocation and use of the computing System's resources among the various user and tasks.
- It provides an interface between the computer hardware and the programmer that simplifies and makes it feasible for coding and debugging of application programs.

Tasks of an Operating System

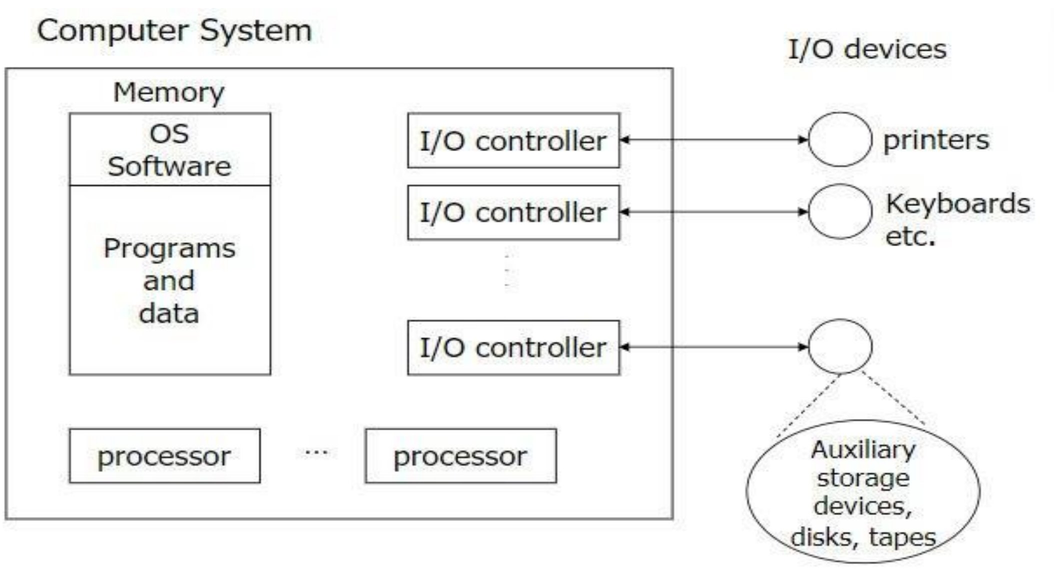
2. Provides the facilities to create and modify programs and data files using an editor.
3. Access to the compiler for translating the user program from high-level language to machine language.
4. Provide a loader program to move the compiled program code to the computer's memory for execution.

5. Provide routines that handle the details of I/O programming.

Question: What is an operating system? Discuss role/functions of OS as a resource manager. What is an operating system? Give the view of the OS as a resource manager.

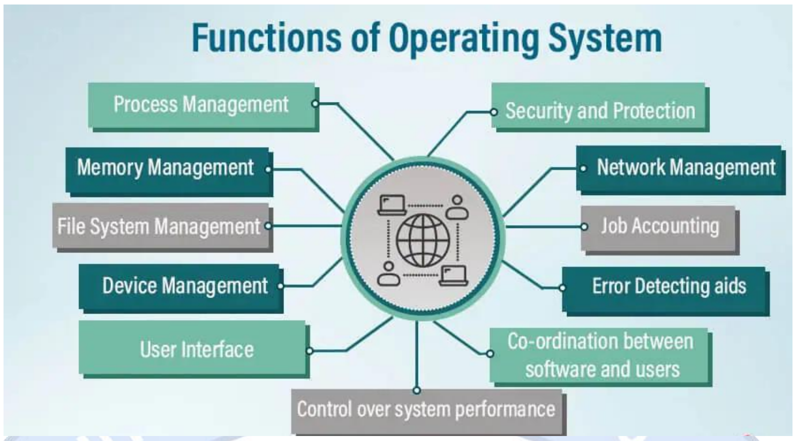
Solution:

2. OS AS RESOURCE MANAGER



- Now-a-days all modern computers consist of processors, memories, timers, network interfaces, printers, and so many other devices.
- The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view.
- Operating system allows multiple programs to be in memory and run at the same time.
- Resource management includes multiplexing or sharing resources in two different ways: in time and in space.
- time:in time slot is allocated to each program first one gets to use the resource then another
- space: as example of main memory , main memory is normally divided up among several running programs , so each one can be resident at the same time.
- In time multiplexed, different programs take a chance of using CPU. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another.
- In space multiplexing, Instead of the customers taking a chance, each one gets part of the resource. For example − Main memory is divided into several running programs, so each one can be resident at the same time.

TOPIC 4: Basic functions



Question: What is an Operating System? Give functions of Operating Systems.

Solution:

1. Process Management

Process management is a key feature of an operating system that handles the creation, planning, execution, and termination of processes or tasks within a computer system. The main components of process management are listed below.

- Process Creation: Creating new processes is the operating system's responsibility. It allows the resources required for the process to run, including memory and data structures. This may involve loading the program code and initializing variables.
- Process Scheduling: The OS manages the execution of processes fairly and efficiently. The CPU executes processes in a particular order using scheduling algorithms. The scheduler chooses which process to execute next based on priority, time slice, and resource availability.
- Process Execution: After scheduling a process, the operating system hands over control of the CPU to the said process, enabling it to execute its instructions. The CPU follows a sequential process of executing instructions. The computer program retrieves directions from its memory and performs computations or actions accordingly. Throughout its execution, it may communicate with other processes, request further resources, or interact with input/output devices.
- Process Termination: A process may terminate when it has finished running or when an error or exception occurs. The operating system reclaims the resources allotted to the terminated process. Ensuring we keep all relevant data structures up-to-date, free up memory, and properly close any open files. Once a system terminates the process, it is removed from the system, freeing up its resources for other processes to use.
- Process Synchronization and Communication: To share data or coordinate activities, processes may have to synchronize their execution or communicate. The operating system has synchronization mechanisms, such as semaphores, locks, and monitors, to ensure that processes can safely access shared resources and prevent conflicts. Inter-process communication mechanisms like shared memory, message passing, and pipes enable processes to exchange data and coordinate activities.
- Process States and Control: The operating system maintains track of details about every process, such as its current state, priority, and resource utilization. A process can be in various states, such as running, waiting, or ready. The operating system controls the changes between these modes depending on circumstances like I/O

completion, timer interrupts, or resource availability. It manages process execution, switching between them and dynamically allocating resources according to their requirements.

2. Memory Management

Memory management plays a vital role in any operating system. It encompasses the allocation, tracking, and deallocation of computer memory resources. Its main objective is to effectively use currently available memory to facilitate the execution of processes and applications.

- Allocation: The operating system allocates memory to processes or applications when they load into memory. Allocating sufficient memory to each process is necessary to guarantee it can complete its duties and handle memory fragmentation problems.
- Deallocation: After a process finishes or gets terminated, the operating system will release the memory used by the process. This will allow the memory to be used for other purposes in the future.
- Address Mapping: The operating system handles address mapping, which translates the logical addresses utilized by processes into physical addresses in the memory.
- Paging and Swapping: Memory management techniques such as paging and swapping enable processes to utilize more memory than is physically available. Paging divides the memory into fixed-size pages while swapping moves inactive processes or parts of them to secondary storage, freeing up physical memory.
- Memory Protection: To ensure data integrity and security, the operating system has memory protection mechanisms that prevent one process from accessing or altering the memory of another process.
- Virtual Memory: By using virtual memory, the operating system can give each process the appearance of having its own dedicated memory space. The system utilizes physical memory and secondary storage, transferring data between the two when necessary.

3. File System Management

Managing the file system is an essential task performed by an operating system. It involves arranging, storing, and retrieving files on various storage devices. I can provide you with a detailed explanation of File System Management:

- File Organization: The file system controls how files are arranged logically and systematically on storage systems. It maintains a directory structure that makes it simple for users to navigate and locate files. This organization can incorporate directories and subdirectories for the hierarchical organization of files.
- File Creation and Deletion: Users can easily create and delete files using the file system. When creating a new file, users can specify its name, location, and initial contents. Similarly, users can delete no longer required files, freeing up storage space.
- File Access and Permissions: The file system manages file access and enforces permissions to uphold data security and protect privacy. Users or user groups are assigned permissions, such as read, write, and execute, to regulate their ability to view, modify, or execute files. This ensures the protection of sensitive data and

restricts unauthorized access.

- File Operations: The file system offers several ways to perform different file operations. These operations involve reading data from a file, writing data to a file, adding data to an existing file, changing the read/write pointer position, and resizing or truncating a file. These operations allow users and applications to manipulate the content of files efficiently.
- File Metadata Management: The file system stores and manages Metadata related to each file. File name, size, creation and modification dates, file owner, and permissions are just a few examples of the information included in the metadata.
- File System Integrity and Recovery: The file system implements mechanisms to guarantee the integrity and recoverability of files and file system data. It uses techniques like journaling or transaction logs to keep track of changes and recover from system problems or failures. This assists in avoiding data corruption and preserving the file system's consistency.

4. Device Management

Device Management is an essential task for an operating system. It involves regulating and coordinating the communication between the computer system and its input/output (I/O) devices. Here is an explanation of Device Management:

- Device Drivers: Device drivers, which are software components that permit communication between the operating system and particular hardware devices, are made available by the operating system. It manages the low-level functions of a device, such as data transmission and receiving, resource management, and converting device-specific directives into a format that the operating system can understand.
- Device Allocation: The management of device allocation to different users or processes is handled by the operating system. It manages device access and prevents conflicts between requests for the same device. Assigning device access involves determining whether access is exclusive or shared, depending on priorities, permissions, and scheduling policies.
- Device Configuration: The operating system configures the device by identifying and detecting hardware attached to the computer system. Every device is classified based on its type, capabilities, and characteristics and configured for optimal performance. This includes creating communication channels, initializing device drivers, and setting device parameters.
- Device Input and Output: The operating system uses devices to manage input and output processes. It controls the data flow between the computer system, input devices (such as keyboards and mice), and output devices (like monitors and printers). The operating system transfers data accurately, buffered as necessary, and synchronized with the operation of processes or applications.
- Device Synchronization: The operating system offers synchronization techniques to manage how devices and processes interact. It manages the order and timing of device operations to avoid data inconsistencies or conflicts. Synchronization may involve locking, signaling, or interrupt handling to ensure proper device access and data integrity.

5. User Interface

The method through which users interact with a computer system or software application refers to as the user interface (UI). Users can enter commands, access system operations, and receive output or feedback from the system using the interface, which can be visual or tactile.

- Graphical User Interface (GUI): GUIs display system functions through icons, menus, buttons, and windows, allowing users to interact with the system by clicking or tapping. The design of these tools prioritizes user-friendliness and intuitive navigation to ensure effortless completion of tasks.
- Command-Line Interface (CLI): CLI is a text-based system where users input commands through a keyboard. Advanced users and system administrators prefer CLI interfaces for direct control and scripting abilities. It's highly efficient for executing complex commands and performing repetitive tasks.
- Menu-Driven Interface: Menu-driven interfaces present users with a list of options. They are often used in more straightforward applications or for a streamlined user experience.
- Natural Language Interface: With a natural language interface, users can communicate with a system using regular, human language. They can input commands or questions in a natural sentence, and the system will interpret and respond appropriately. These interfaces commonly use artificial intelligence and natural language processing techniques to understand and process user input.
- Touch-Based Interface: Developers create touch-based interfaces explicitly for devices with touchscreens, such as smartphones and tablets. These interfaces enable users to interact with the system through direct touch on the screen and gestures such as tapping, swiping, or pinching. Touch-based interfaces offer intuitive and tactile interactions, which help users navigate, select options, and perform actions using their fingers.

6. Security and Protection

Ensuring the safety and protection of a computer system, its resources, and the data it processes are essential functions of an operating system.

- Access Control: To protect the system resources, the operating system uses access control mechanisms that only allow authorized users or processes to access them. It enforces authentication methods, manages user accounts, and sets permissions to restrict access to sensitive data, networks, devices, and files.
- Data Encryption: The operating system has built-in features that safeguard sensitive information from being accessed or intercepted by unauthorized individuals using data encryption techniques. Encryption algorithms and cryptographic protocols utilized during transmission and storage ensure data security from unauthorized access or tampering.
- Malware Detection and Prevention: The operating system includes security measures to detect and prevent malware infections. These measures include antivirus software, firewalls, and intrusion detection systems designed to detect and mitigate security threats such as viruses, worms, trojans, and unauthorized network access attempts.
- Firewalls and Intrusion Detection Systems: The network has firewalls and intrusion detection systems that oversee and manage all incoming and outgoing traffic. They filter out any malicious or unauthorized activities. Firewalls stop unauthorized

network connections, while intrusion detection systems identify and respond to suspicious behavior or security breaches.

7. Network Management

Network management refers to administering, monitoring, and optimizing computer networks to ensure efficient and secure operation. It involves processes, tools, and methodologies to maintain the network's performance, reliability, and security. Network management tasks can vary depending on the size and complexity of the network, ranging from small local area networks (LANs) to large-scale wide area networks (WANs) and data centers.

Key aspects of network management include:

- Network Monitoring: Managing a network requires ongoing monitoring of devices like routers, switches, firewalls, servers, and network links. Monitoring tools collect data on network performance, traffic patterns, and device health, allowing administrators to identify and address issues proactively.
- Network Configuration Management: Configuration management includes maintaining and updating network device configurations. Network administrators ensure proper configuration of devices, adherence to best practices, and implementation of necessary changes when required.
- Network Troubleshooting: When network issues occur, network management involves troubleshooting to identify and resolve the root causes of problems. Troubleshooting may include analyzing logs, conducting packet captures, and using network diagnostic tools.
- Network Performance Optimization: Network managers work to optimize the network's performance, ensuring efficient data flow and minimizing delays or bottlenecks. This may involve load balancing, traffic shaping, and QoS (Quality of Service) configuration.

8. Control over system performance

Controlling system performance is critical to managing computer systems and ensuring their efficient operation. It involves various techniques and tools to monitor, optimize, and maintain the performance of hardware, software, and networks to meet user expectations and requirements. Controlling system performance helps avoid bottlenecks, reduce response times, and maximize resource utilization.

9. Job Accounting

In computing, job accounting, also known as workload accounting or resource accounting, involves tracking and recording resource usage and performance metrics of individual jobs or tasks. It gathers information on resource usage by different processes, applications, or users, aiding in billing, performance analysis, capacity planning, and optimization. Job accounting is commonly used in multi-user systems, high-performance computing clusters, cloud computing environments, and shared hosting services.

10. Error Detecting aids

Mechanisms and techniques used in computing and communication systems identify and detect errors or anomalies that may occur during data transmission, processing, or storage. These are known as error-detecting aids. These aids are crucial for maintaining data integrity, ensuring information accuracy, and improving systems' reliability.

11. Co-ordination between software and users

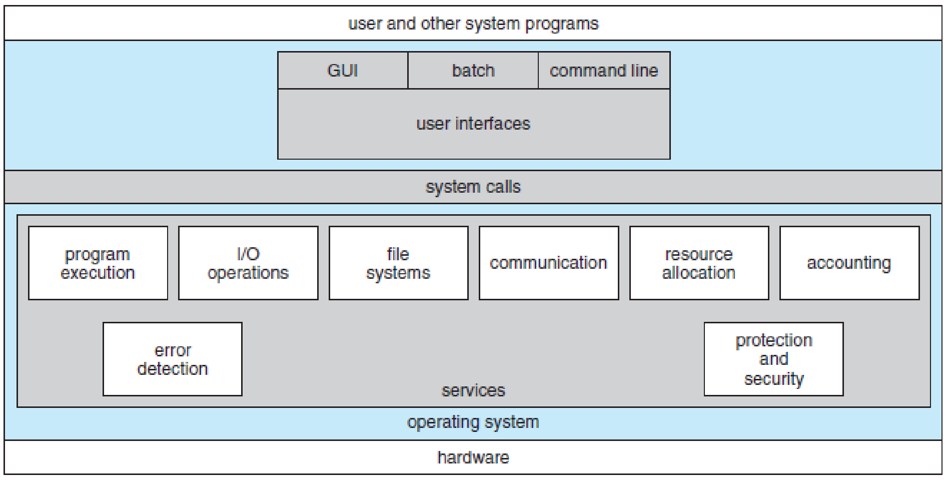
Operating systems are crucial in coordinating and managing various software tools like interpreters, compilers, assemblers, and other applications for different users in a computer system. They handle resource allocation, process management, communication, and security, enabling a seamless and efficient user experience while executing different software tools.

TOPIC 5: Interaction of O.S. & hardware architecture

Question: Explain different services provided by operating system.

Solution:

- An operating system is an interface between hardware and software.
- It provides certain services to programs and users using those programs.
- Operating system services makes of the programming task easier for programmer



Services for user:

- User interface: All operating systems have user interface. There are two major types of user interface: 1) Command line Interface which uses text commands and 2) Graphical user Interface which uses menus for I/O and for selecting options.
- Program execution: Operating systems makes sure that the system must be able to load a program into memory and to run that program.
- I/O operations: A running programme require I/O such as a file or a device. So for efficiently managing entire system operating system must provide a means to do I/O.
- File-system manipulation: Programs need to create, read, write, and delete files. So operating system must manage file system by including permissions. Such as deciding which user can read or write file.
- Communications: Communication is required between processes to exchange information. Operating systems implements communication services by either: 1) Shared memory: Here two or more processes read and write to a shared section of memory2) Message passing: Message Passing takes packets of information in specified formats and moves the packet between processes.

- Error detection: The operating system needs to detect and correct errors constantly. Different errors such as memory error, power failure, connection failure etc can occur which needs to be corrected by operating system.

Other Services:

- Resource allocation: When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them. The operating system manages many different types of resources such as main memory, file storage, CPU cycles etc.
- Accounting: Operating System should keep track of which users use how much and what kinds of computer resources. This record keeping may be used for accounting (so that users can be billed).
- Protection and security: Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important.

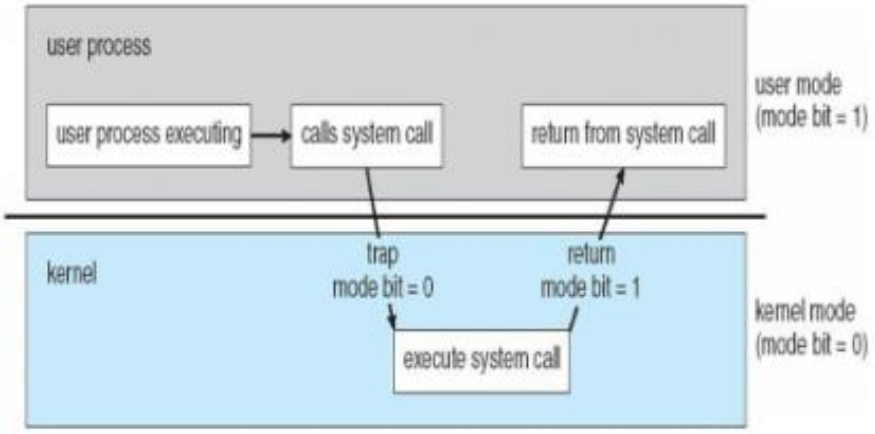
TOPIC 6: System calls

Question:What is a system call?
Solution:
DEFINATION: A system call is a way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request.

- A system call can be written in assembly language or a high-level language like C or Pascal. System calls are predefined functions that the operating system may directly invoke if a high-level language is used.
- A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.
- The Application Program Interface (API) connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services. The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.

Question:Explain steps for system call execution.
Solution:

A figure representing the execution of the system call is given as follows −

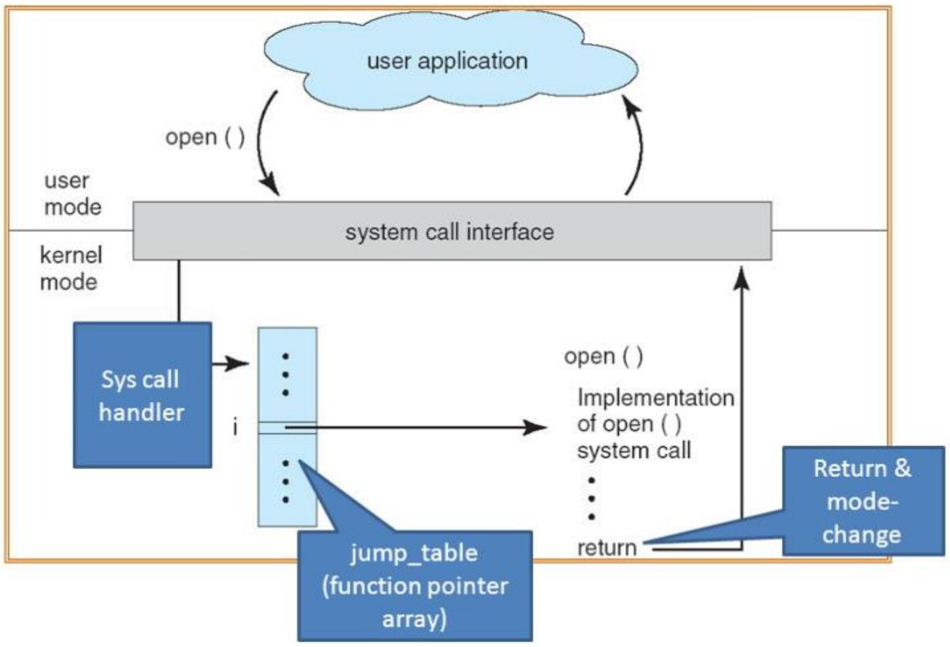


- As can be seen from this diagram, the processes execute normally in the user mode until a system call interrupts this.
- Then the system call is executed on a priority basis in the kernel mode.
- After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed.
- In general, system calls are required in the following situations −
    1. If a file system requires the creation or deletion of files. Reading and writing from files also require a system call.
    2. Creation and management of new processes.
    3. Network connections also require system calls. This includes sending and receiving packets.

    4. Access to a hardware devices such as a printer, scanner etc. requires a system call.
    5. The Applications run in an area of memory known as user space.
    6. A system call connects to the operating system's kernel, which executes in kernel space. When an application creates a system call, it must first obtain permission from the kernel.
    7. It achieves this using an interrupt request, which pauses the current process and transfers control to the kernel.
    8. If the request is permitted, the kernel performs the requested action, like creating or deleting a file.
    9. As input, the application receives the kernel's output. The application resumes the procedure after the input is received.
    10. When the operation is finished, the kernel returns the results to the application and then moves data from kernel space to user space in memory.

A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time. A more complicated system call, such as connecting to a network device, may take a few seconds. Most operating systems launch a distinct kernel thread for each system call to avoid bottlenecks. Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.

Working of System call



Steps involved in making a system call are:

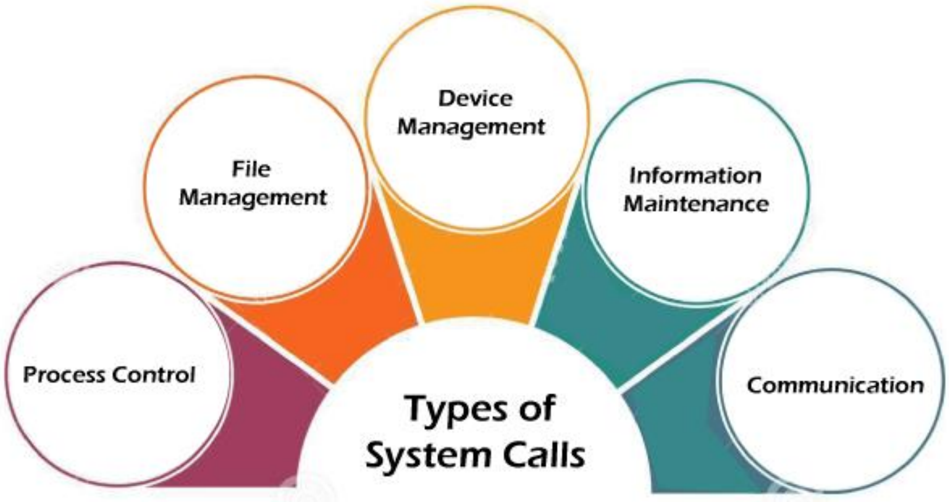
- User applications first set up the arguments to make a system call.
- The program then executes the "system call" instruction once all arguments have been set up.
- A processor exception is generated when this instruction causes it to jump to a new address and start executing the code.
- By saving your program's state at the new address, finding the system call you want, calling the kernel function that translates that call, and restoring your program's state, the instructions at the new address allow you to move control back to the user program.

Question: Write different types of System Calls.

Solution:

Types of System Calls

There are commonly five types of system calls. These are as follows:



1. **Process Control**
2. **File Management**
3. **Device Management**
4. **Information Maintenance**
5. **Communication**

Process Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

Examples of Windows and Unix system calls

There are various examples of Windows and Unix system calls. These are as listed below in the table:

Process	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		Fork()
		Exit()
		Wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		Open()
		Read()
		Write()
		Close()

Device Management	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		Ioctl()
		Read()
		Write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		Getpid()
		Alarm()
		Sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		Pipe()
		Shmget()
		Mmap()

Protection	SetFileSecurity()
	InitializeSecurityDescriptor()
	SetSecurityDescriptorgroup()
		Chmod()
		Umask()
		Chown()

TOPIC 7: Batch, multiprogramming. Multitasking, time sharing, parallel, distributed & realtime O.S.

Question: What is an operating system? List the types of operating system and explain in brief.

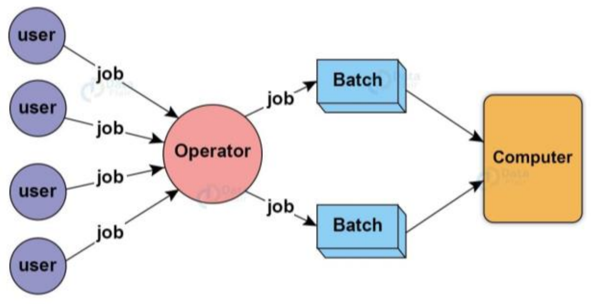
Solution:

- An operating system is a program that manages computer hardware. It provides interface between user and computer hardware.
- Operating System helps user to communicate with the computer. Eg. Windows, unix , mac os etc.

Types of operating system:

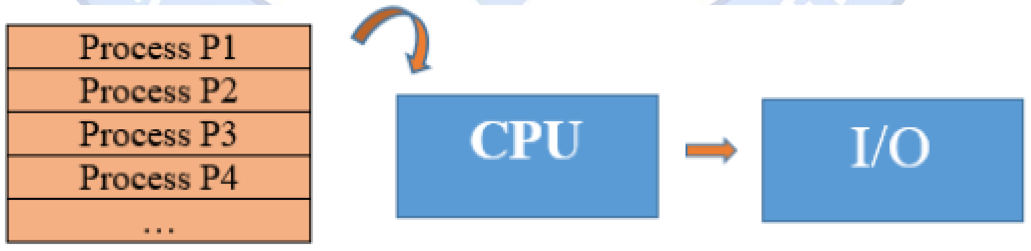
- **Batch operating system**
- **Multiprogramming operating system**
- **Multitasking/Time Sharing operating system**
- **Distributed operating system**
- **Network operating system**
- **Real Time operating system**
- **Embedded operating system**

Batch operating system



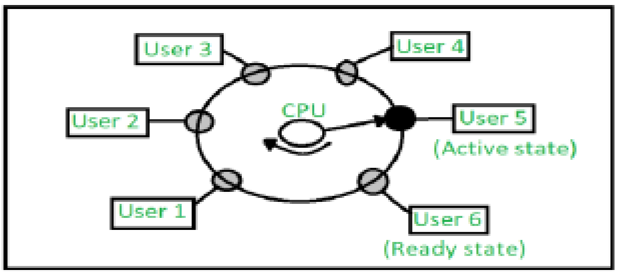
- Batch operating system does not communicate or interact with the computer directly
- Each user uploads its task to punch cards and give it to operator.
- There is an operator in this kind of operating system
- Operator clubs similar jobs having the same requirement and give to CPU in form of batches.
- Advantages of Batch Operating System
- It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in the queue.
- Multiple users can share the batch systems.
- The idle time for the batch system is very less.
- It is easy to manage large work repeatedly in batch systems.
- Disadvantages of Batch Operating System
- The computer operators should be well known with batch systems.
- Batch systems are hard to debug.
- It is sometimes costly.
- The other jobs will have to wait for an unknown time if any job fails.
- Examples of Batch Operating Systems: Payroll Systems, Bank Statements, etc.

Multiprogramming Operating System



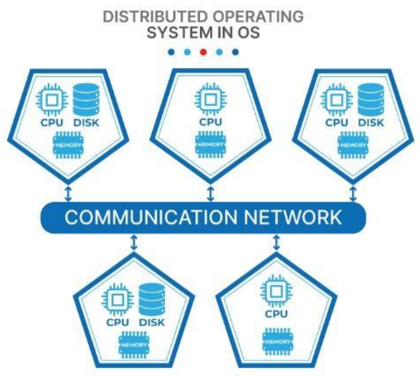
- In multiprogramming operating system, more than one programs can execute on single processor.
- This is achieved through uploading more than one process into main memory
- Multiple jobs are kept main memory. If one job is busy with IO, CPU can be assigned to other job.
- Adv: CPU will not be idle and increases degree of multiprogramming
- Disadv: CPU scheduling must be done

Multitasking/Time Sharing operating system



- Here multiple processes are executed by CPU, simultaneously by switching between them.
- CPU works based on time quantum. E.g. If time quantum is 2s, then each process will get CPU for two seconds.
- Because of fast switching, user feels like multiple processes are running at the same time.
- Also this will increase response time of process.
- Advantages of Multi-Tasking Operating System
- Multiple Programs can be executed simultaneously in Multi-Tasking Operating System.
- It comes with proper memory management.
- Disadvantages of Multi-Tasking Operating System
- The system gets heated in case of heavy programs multiple times.

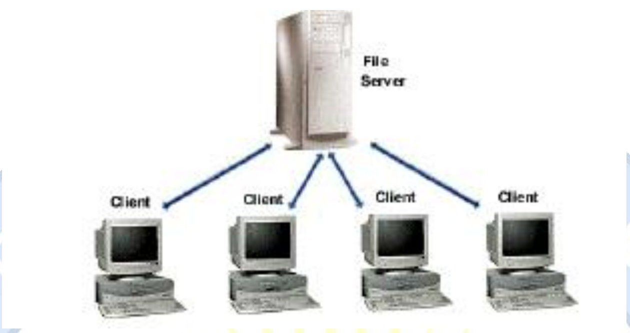
Distributed Operating System



- In this type of system, multiple systems are connected in communication networks
- These systems have their own resources
- Here user will take it as single system even though there are multiple system even though there are multiple systems in the network.
- User does not know which systems are in distributed network
- Advantages of Distributed Operating System
- Failure of one will not affect the other network communication, as all systems are independent of each other.
- Electronic mail increases the data exchange speed.
- Since resources are being shared, computation is highly fast and durable.
- Load on host computer reduces.
- These systems are easily scalable as many systems can be easily added to the network.
- Delay in data processing reduces.
- Disadvantages of Distributed Operating System
- Failure of the main network will stop the entire communication.

- To establish distributed systems the language is used not well-defined yet.
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.
- Examples of Distributed Operating Systems are LOCUS, etc.
-

Network Operating System



- Network operating system runs on a server and gives the server the capability of managing data, users, groups, security, applications and other networking functions.
- The basic purpose of the network operating system is to allow shared files and printer access among multiple computers in the network.
- Here security is managed by server
- Adv: File sharing and resource sharing is possible in network operating system.
- Disadv: Cost of buying and running the server is high.

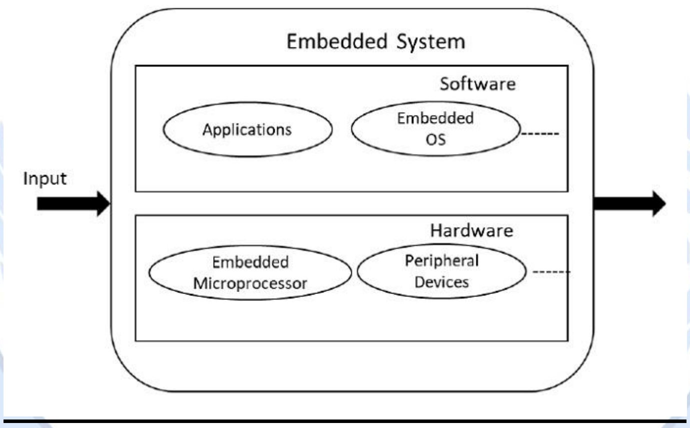
Real Time operating system

- Time constraints is the key parameter in the Real Time operating system
- When user gives an input an input to the system it must response within specified time
- **Hard real time system**: Hard real time systems guarantees that critical task completes on time. E.g. Missile Launching
- **Soft real time system:** Soft real time systems are less sensitive and cannot guarantee that it will be able to meet deadline in all conditions E.g. Gaming
- Advantages of RTOS
- Maximum Consumption: Maximum utilization of devices and systems, thus more output from all the resources.
- Task Shifting: The time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds in shifting from one task to another, and in the latest systems, it takes 3 microseconds.
- Focus on Application: Focus on running applications and less importance on applications that are in the queue.
- Real-time operating system in the embedded system: Since the size of programs is small,

RTOS can also be used in embedded systems like in transport and others.

- Error Free: These types of systems are error-free.
- Memory Allocation: Memory allocation is best managed in these types of systems.
- Disadvantages of RTOS
- Limited Tasks: Very few tasks run at the same time and their concentration is very less on a few applications to avoid errors.
- Use heavy system resources: Sometimes the system resources are not so good and they are expensive as well.
- Complex Algorithms: The algorithms are very complex and difficult for the designer to write on.
- Device driver and interrupt signals: It needs specific device drivers and interrupts signal to respond earliest to interrupts.
- Thread Priority: It is not good to set thread priority as these systems are very less prone to switching tasks.
- Examples of Real-Time Operating Systems are Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.
-
-

Embedded operating system



- This operating system are not used in regular computers
- Embedded operating systems are used in small devices like robots, DVDs, microwave ovens.
- They have limited features and specific task.
- All Embedded Systems are task specific. They mostly do a particular task on loop/repeatedly for their entire lifetime. These systems are designed to execute their task within a particular time interval, and thus they have to be fast enough to be up to their time limit.
- They have little or no user interface like a fully automatic washing machine does its task fully once its programmed is set and stops after its work is finished with almost no user interface.
- They are built to achieve a particularly good efficiency level. They are very small in size operating system, need little power
- These systems can't at all be upgraded or updated. Thus, they must be really high on efficiency and reliability as they can't be updated.
- Advantages
- The advantages of embedded operating system are as follows −
- Portable
- Much faster than other operating systems
- Less Hardware requirement
- Highly Predictable

- Disadvantages
- The disadvantages of embedded operating system are as follows −
- Less optimization
- High modification is required
- Customization is time taking process

Question: Describe in detail about variety of techniques used to improve the efficiency and performance of secondary storage.

some techniques used to enhance the efficiency and performance of secondary storage:

1. Disk Caching: Disk caching involves storing frequently accessed data in a faster storage medium (such as RAM) to reduce read/write latency from the slower secondary storage (like hard disks). This technique improves overall system responsiveness.
2. Redundant Arrays of Inexpensive Disks (RAID): RAID configurations combine multiple physical disks into a single logical unit. Different RAID levels (e.g., RAID 0, RAID 1, RAID 5) provide various benefits such as improved performance, fault tolerance, or both.
3. File Compression and Decompression: Compressing files reduces their size, allowing more data to fit on storage devices. Decompression occurs when accessing the data, ensuring efficient storage utilization.
4. Deduplication: Deduplication identifies and eliminates duplicate data blocks within storage. By storing only unique data, it reduces storage requirements and enhances efficiency.
5. Tiered Storage: Tiered storage involves organizing data into different tiers based on access frequency. Frequently accessed data resides on faster storage tiers (e.g., SSDs), while less frequently accessed data is moved to slower, cost-effective tiers (e.g., traditional hard drives).
6. Object Storage: Object storage systems (like Amazon S3 or OpenStack Swift) provide scalable and efficient storage for unstructured data. They use unique identifiers (object keys) to manage data, making retrieval and distribution more efficient.