OPERATING SYSTEMS

GUIDE FOR " U "

LEKSHMI SURYA S L
ASSISTANT PROFESSOR
ON CONTRACT
SREE NARAYANA COLLEGE
KOLLAM

FIRST EDITION

OPERATING SYSTEMS

GUIDE FOR " U "

About Author

Lekshmi Surya S L is working as Assistant Professor on Contract of Physics and Computer Applications Department at Sree Narayana College Kollam. She is a graduate in B Tech from Kerala University and post graduate from Anna University. She is NET qualified. She is having 8 years of teaching experience from various Colleges. She has presented papers in various conferences, published journals and attended various Seminars, Workshops and Conferences.

ISBN NO: 978-81-958369-5-6

MRP 200 /-

SEMESTER VI

OPERATING SYSTEMS

OPERATING SYSTEMS

A Complete Guide for
B.Sc. Computer Science, B.Sc. Computer Application, BCA,
B.Sc. Physics & Computer Application

(University of Kerala)

Revised Syllabus CBCSS 2021

LEKSHMI SURYA.S.L
(B.Tech, M.Tech, UGC-NET CS)
Assistant Professor on Contract
Department of Physics and Computer Application
Sree Narayana College Kollam

OPERATING SYSTEMS
Third Year B.Sc
Revised Syllabus CBCSS 2021
First edition

Typed by
Lekshmi Surya.S.L

Printed at
Pournami Graphics
Kappalandimukku
Near S N College
S.N.Press Compound
Karbala
Kollam
Kerala-691001

ISBN : 978-81-958369-5-6

Copies can had from:
S.N.College Library
And Physics and Computer Application Dept.
S.N.C Kollam
Mob:-6282741938
e-mail:-lekshmisnc2020@gmail.com

PREFACE

Though this book is especially written for the student of B.Sc. Physics and Computer Applications department, of Kerala University, but it is equally useful for B.Sc. (CS) BCA of Kerala University.

Operating Systems research is a vital and dynamic field. Even young computer science students know that Operating Systems are the core of any computer system and a course about Operating Systems is more than common in any Computer Science department all over the world.

Self-study almost anything without anyone's help and cultivate self-confidence to learn almost anything. · Score extra marks without additional hard work. Score more even if you have less time for preparation. This is a complete guide for degree students who can pass the subject with highest mark. This book completely based on previous year university question papers and pass through all the portions provided in the syllabus.

Thanks to Management of Sree Narayana College, Colleagues, Students. I Express My Gratitude and I am also grateful to my parents for their valuable suggestion. I am also deeply indebted to our principal and colleagues for their encouragement .I am also but not least thanks to my parents and my daughter for extending their cooperation.

I will appreciate any suggestion for improving the quality and usefulness of the book. Best of Luck dear students.

Lekshmi Surya.S.L

UNIVERSITY OF KERALA

B.Sc PHYSICS AND COMPUTER APPLICATIONS

Semester-VI

PC1672: OPERATING SYSTEMS

1. COURSE OUTCOMES

At the end of the Course, the student will be able to

CO1	Understand working of various Operating Systems
CO2	Apply constrained resource allocation, process scheduling and memory management techniques
CO3	Evaluate synchronization of processes and protection of an Operating System
CO4	Analyse salient features available to various Operating Systems

2. SYLLABUS

Module I: Operating System Overview: Introduction - Structure of Operating System, the Evolution of Operating System, Operating System Functions, System calls. Distributed Systems: introduction, Trends in Distributed System, challenges.

Module II: Process Management: The Process, Process State, PCB, Threads, Process Scheduling - Basic Concepts, Scheduling Criteria, Scheduling Algorithms. Process Coordination: Critical Section problems, Semaphores, Synchronization - Interprocess Communication Problems. Deadlock – Definition, Resource Allocation Graph, Conditions of deadlock, deadlock prevention, deadlock avoidance, deadlock detection, deadlock recovery.

Module III: Memory Management: Basic Hardware, Address binding, Logical vs. physical address space, Dynamic Loading and Linking, Swapping, Memory Allocation Methods, Paging, Structure of Page Table, Segmentation, Virtual Memory- Background Demand Paging, Page Replacement- Basic Page Replacement, FIFO Page Replacement, Optimal Page Replacement, LRU Page Replacement, Thrashing.

Module IV: Storage Management: File Concept, Access Methods, Protection, Implementation- File System Structure, Allocation Methods, Recovery, Secondary Storage- Overview, Disk Scheduling, Disk Management, RAID. I/O Systems- I/O Hardware, Application I/O Interface, Kernel I/O Subsystem.

INDEX

MODULE – I

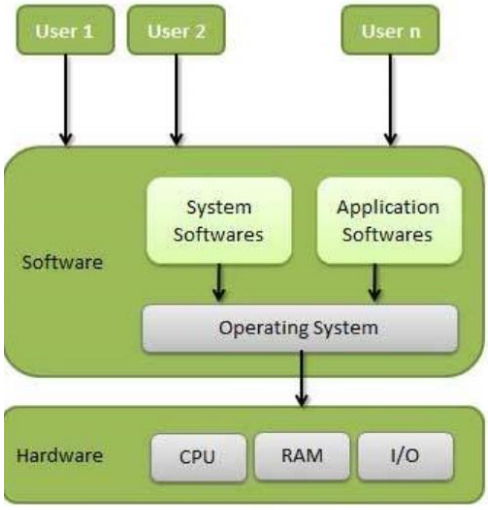
OPERATING SYSTEM OVERVIEW

2 marks

1. What is an operating system?

   An Operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

   Intechnical terms, It is a software which manages hardware. An Operating System controls the allocation of resources and services such as memory, processors, devices and information.



2. What are operating system services?

   An Operating System provides services to both the users and to the programs.

   It provides programs, an environment to execute.
   It provides users, services to execute the programs in a convenient manner.

   Following are few common services provided by operating systems.

   - Program execution
   - I/O operations

- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

1. Program Execution:

■ The Operating System is responsible for execution of all types of programs whether it be user programs or system programs.

■ The Operating System utilizes various resources available for the efficient running of all types of functionalities.

2.Handling Input/Output Operations:

■ The Operating System is responsible for handling all sorts of inputs, i.e, from keyboard, mouse, desktop, etc

- The Operating System does all interfacing in the most appropriate manner regarding all kinds of Inputs and Outputs.

3.Manipulation of File System:

■ The Operating System is responsible for making decisions regarding the storage of all types of data or files, i.e, floppy disk/hard disk/pen drive, etc.

■ The Operating System decides how the data should be manipulated and stored.

4.Error Detection and Handling:

■ The Operating System is responsible for detection of any types of error or bugs that can occur while any task. ■ The well secured OS sometimes also acts as countermeasure for preventing any sort of breach to the Computer System from any external source and probably handling them.

5.Resource Allocation:

■ The Operating System ensures the proper use of all the resources available by deciding which resource to be used by whom for how much time.

■ All the decisions are taken by the Operating System.

6.Accounting:

■ The Operating System tracks an account of all the functionalities taking place in the computer system at a time.

■ All the details such as the types of errors occurred are recorded by the Operating System.

7.Information and Resource Protection:

■ The Operating System is responsible for using all the information and resources available on the machine in the most protected way.

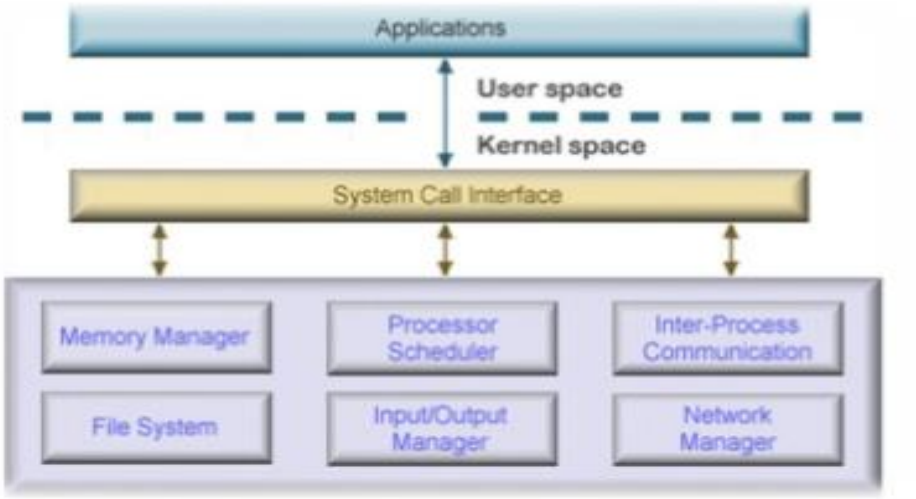
■ The Operating System must foil an attempt from any external resource to hamper any sort of data or information

3. Explain OS structure?

- Monolithic Systems
- Layered Systems
- Micro Kernels
- Modular

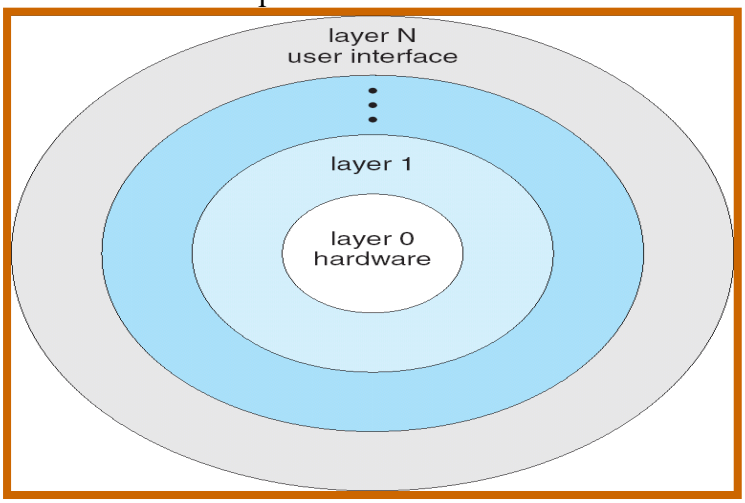
- **Monolithic Systems**

- In this model, for each system call there is one service procedure that takes care of it and executes it. The utility procedures do things that are needed by several service procedures.



● **Layered structure**:
○ An OS can be broken into pieces and retain much more control on 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 only.
○ This simplifies the debugging process as 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.
○ The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system.

○ Moreover careful planning of the layers is necessary as a layer can use only lower level layers.
○ UNIX is an example of this structure.



✱ Microkernels

- This method structures the OS by removing all known essential components from the kernel and implementing them as system and user level programs. The result is a smaller kernel.
- Microkernels provide minimal process and memory management in addition to communication facility.

Function of microkernel

+ provide communication between client program and the various services that are also running in the user space.
+ communication is provided through message passing.

Advantages

+ It makes extending the operating system easier. all new services are added to user space and consequently do not require modification of the kernel
+ easier to port from one hardware design to another.
+ microkernel also provides more security and reliability since most services are running as user rather than kernel processes if a service fails the rest of the OS remains untouched.

- Mac OS is an example of this type of OS.

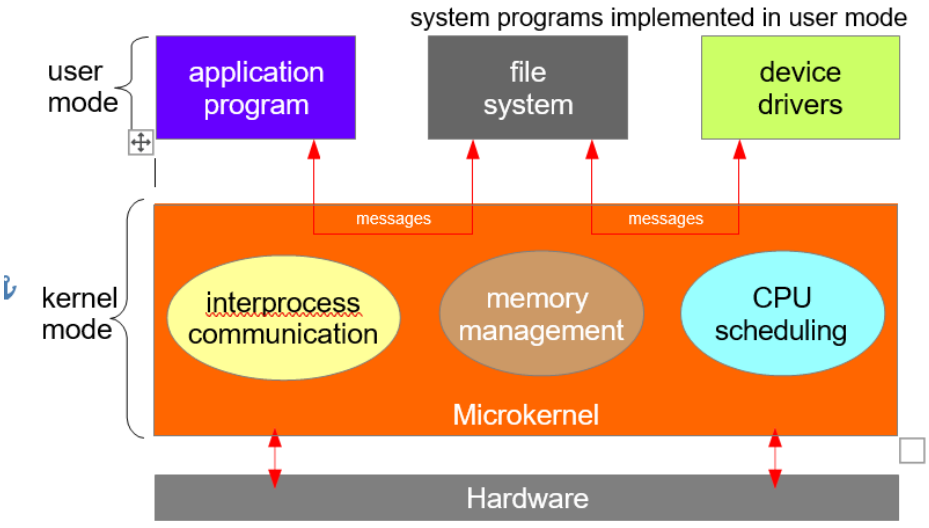


Fig : Microkernel implementation

Modular structure or approach:

○ It is considered as the best approach for an OS.
○ It involves designing a modular kernel.
○ The kernel has only a set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time.
○ It resembles layered structure due to the fact that each kernel has defined and protected interfaces

but it is more flexible than the layered structure as a module can call any other module.
- For example Solaris OS

- **Differences Between Microkernel and Monolithic Kernel**
  - The microkernel is much smaller in size as compared to the monolithic kernel.
  - The microkernel is easily extensible whereas this is quite complicated for the monolithic kernel.
  - The execution of the microkernel is slower as compared to the monolithic kernel.
  - Much more code is required to write a microkernel than the monolithic kernel.
  - Examples of Microkernel are QNX, Symbian, L4 Linux etc. Monolithic Kernel examples areLinux, BSD etc.

4. Describe the operating system functions?(2022-4 marks)

What are the various objectives and functions of Operating systems?

What does an Operating system do?

Basic Functions of Operating System:

- Managing Resources: OS will coordinate all the computer resources (Hardware and Software)
- Providing a user interface: Users interact with application programs and hardware through a user interface
- Running Applications: These programs load and run applications such as word processors and spreadsheets
- Support for built-in utility programs: Uses utility programs for maintenance and repairs
- Control to the computer hardware: OS sits between the programs and the Basic Input Output System

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

○**1.Security** –
■ The operating system uses password protection to protect user data and similar other techniques.
■ it also prevents unauthorized access to programs and user data.

○ **2.Control over system performance** –
■ Monitors overall system health to help improve performance.
■ Records the response time between service requests and system response to have a complete view of the system health.
■ This can help improve performance by providing important information needed to troubleshoot problems.

○ 3.**Job accounting –**
■ Operating system Keeps track of time and resources used by various tasks and users.
■ This information can be used to track resource usage for a particular user or group of users.

○ **4.Error detecting aids** –
■ Operating system constantly monitors the system to detect errors and avoid the malfunctioning of the computer system.

○ **5.Coordination between other software and users** –
■ Operating systems also coordinate and assign interpreters, compilers, assemblers and other software to the various users of the computer systems.

○ **6.Memory Management**–
■ The operating system manages the Primary Memory or Main Memory.
■ It keeps track of primary memory, i.e., which bytes of memory are used by which user program
■ In multiprogramming, the OS decides the order in which processes are granted access to memory, and for how long.

○**7. Processor Management** –
■ In a multiprogramming environment, the OS decides the order in which processes have access to the processor, and how much processing time each process has.
■ This function of the OS is called process scheduling.

○ **8.Device Management** –
■ An OS manages device communication via their respective drivers.
■ Keeps track of all devices connected to the system. designates a program responsible for every device known as the Input/Output controller.
■ Decides which process gets access to a certain device and for how long. Allocates devices in an effective and efficient way.

○ **9.File Management** –
■ A file system is organized into directories for efficient or easy navigation and usage.
■ OS keeps track of where information is stored, user access settings and status of every file and more…
■ These facilities are collectively known as the file system.

5. Describe the operating system operations?
   Explain dual-mode operations of OS. (2022-2marks)

Operating System Operations

- modern operating systems are interrupt driven.
- If there are no process to execute, no i/o devices to service, and no users to whom to respond, an OS will sit quietly, waiting for something to happen.
- A trap (exception) is a software generated interrupt caused either by an error (division by 0 or invalid memory access) or by a request from a user for a service.

Dual-Mode and Multi-Mode Operation

- There are 2 seperate modes of operation

Dual-Mode of Operation *(provides a means for protecting OS)* *Privileged Instructions* *:-*

- Hardware allows these instructions to be executed only in kernel mode.
- If an attempt is made to execute these instructions in user mode, the hardware treats it as illegal and traps itto the OS.

Example for privileged instruction :- I/O control, timer management, interrupt management.

Multi-Mode Operation *(concept of modes can be extended modes)* – CPU uses more than 1 bit to set and test the mode

Timer

Use

a. to ensure that the OS maintains control over the CPU.

b. not possible to allow a user program to get stuck in an infinite loop or fail to call system services .. for these reasons, a timer is used.

- a timer can be set to interrupt the computer after a specific period.

- the period may be fixed or variable. -variable timer is implemented by a counter.

- OS sets the counter.

- every time the clock ticks the counter is decremented.

- when the counter reaches zero, an interrupt occurs

6. Compare tightly coupled systems and loosely coupled systems?

Loosely coupled systems:-

Each processor has its own local memory. Each processor can communicate with other all through communication lines

Tightly coupled systems:-
Common memory is shared by many processors No need of any special communication lines.

7. Explain simple batch system?

Operators batched together jobs with similar needs and ran through the computer as a group .The operators would sort programs into batches with similar requirements and as system become available, it would run each batch.

8. Explain time sharing operating system?

Time sharing is the sharing of a computing facility by several users that want to use the same facility at the same time. The tasks are given specific time and operating system switches between different tasks. User interaction is involved in the processing.

9. List out any four process control system calls?

* Process Control
  - end, abort
  - load, execute
  - create/terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate & free memory

10. Explain virtual machines? (2022-1 mark)

Virtual Machine is a completely separate individual operating system installation on your usual operating system. It is implemented by software emulation and hardware virtualization.

Advantages:

- Multiple OS environments can exist simultaneously on the same machine, isolated from each other.
- Virtual machine can offer an instruction set architecture that differs from real computers.
- Easy maintenance, application provisioning, availability and convenient recovery.

11 List out any four information management system calls?

* Information Maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get process, file or device attributes
  - set process, file or device attributes

12 Describe distributed operating system?

.

Distributed systems use multiple central processors to serve multiple real time application and multiple users. Data processing jobs are distributed among the processors accordingly to which one can perform each job most efficiently. The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers and so on.

13.What is the Kernel? (2022-1 mark)

The term "kernel" in the context of operating systems refers to the core component of an operating system.
A more common definition is that the OS is the one program running at all times on the computer, usually called the kernel, with all else being application programs. The kernel is responsible for managing the fundamental operations of the computer's hardware and providing essential services to software applications and processes. In summary, the kernel is the heart of an operating system, responsible for managing hardware resources, providing services to applications, and ensuring the overall stability and security of the system.

4 Marks

1.What are the major activities of an operating systems with regard to process management?

The five major activities are:

a. The creation and deletion of both user and system processes

b. The suspension and resumption of processes

c. The provision of mechanisms for process synchronization

d. The provision of mechanisms for process communication

e. The provision of mechanisms for deadlock handling
(Go through the functions of OS)

2. Differentiate distributed systems from multiprocessor system?

Parallel processing is one which divided the instructions into multiple processor whereas Distributed processing is one which run the same instructions into multiple processor to provide more capability for a device. Parallel processing use shared memory whereas Distributed processing use unique memory.

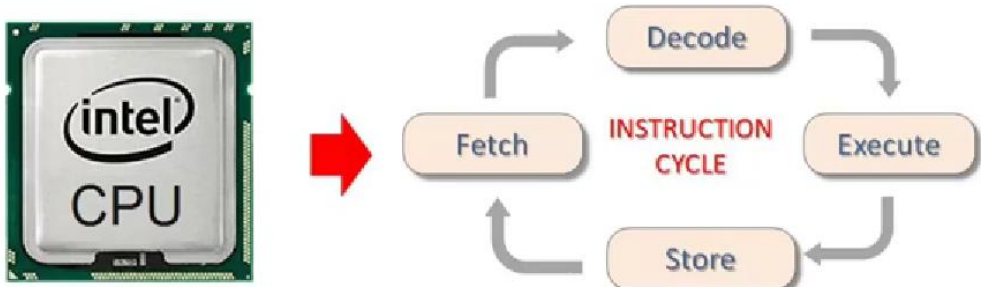
- Distributed System: A distributed system is a collection of interconnected computers or nodes that work together to solve a problem or perform a task. These nodes are typically geographically dispersed and communicate with each other over a networkTasks are performed with a less speedy process. Distributed systems are also known as loosely coupled systems. These systems communicate with one another through various communication lines, such as high-speed buses or telephone lines. These systems do not share memory or clock in contrast to parallel systems. In this there is no global clock in distributed computing, it uses various synchronization algorithms.
- Multiprocessor System: A multiprocessor system, also known as a parallel computing system, is a single computer system that contains multiple processors (CPU cores) that share a common memory and execute tasks concurrently. . Tasks are performed with a more speedy process. It is also known as a tightly coupled system. These systems have close communication with more than one processor. These systems share a memory, clock, and peripheral devices. n this, all processors share a single master clock for synchronization.

3.Explain the basic instruction cycle with appropriate diagram?

The basic instruction cycle, also known as the fetch-decode-execute cycle, is a fundamental concept in computer architecture and operating systems. It describes how a CPU (Central Processing Unit) fetches, decodes, and executes instructions from memory.

Basic Instruction Cycle Phases:

1. **Fetch**: In the fetch phase, the CPU fetches the next instruction from memory. This instruction is typically located in the memory address pointed to by the program counter (PC). The PC is a special register that keeps track of the address of the next instruction to be executed.
2. **Decode**: Once the instruction is fetched, the CPU decodes it. The decoding phase involves interpreting the instruction to determine the operation it specifies and the operands involved. This step is crucial because it sets up the CPU to perform the correct operation.
3. **Execute**: In the execute phase, the CPU performs the operation specified by the decoded instruction. This operation could involve arithmetic calculations, data movement, branching (changing the flow of execution), or other operations. The result of the execution is often stored in registers or memory.
4. **Write-back**: After executing the instruction, the CPU may write the result back to a register or memory location. This phase is optional, as not all instructions modify data or need to write back results.



4.Explain about multiprogramming and time sharing operating system?
What do you mean by degree of multiprogramming? (2022-1 mark)

Multiprogramming is an operating system technique that allows multiple programs or processes to be loaded and executed in the computer's memory simultaneously. The CPU switches between these programs, giving the illusion that they are all running at the same time.

Degree of multiprogramming refers to the number of programs that can be simultaneously loaded and executed in a computer system. It is a critical parameter in the field of operating systems and computer architecture. The degree of multiprogramming influences the system's overall performance, efficiency, and resource utilization. A higher degree of multiprogramming means that more programs can be kept in memory and executed concurrently. This can lead to better resource utilization and higher system throughput.

Resource Allocation: The primary goal of multiprogramming is to maximize CPU and memory utilization. When one program is waiting for I/O (e.g., reading from a disk), the CPU can switch to another program that is ready to execute, keeping the CPU busy.
Examples: Many traditional batch processing systems and early operating systems, such as early versions of Unix and MS-DOS, used multiprogramming to execute multiple jobs sequentially.

- A **time-sharing operating system**, also known as a multitasking or multi-user operating system, allows multiple users to interact with the computer simultaneously by sharing its resources (e.g., CPU and memory) in short time slices or time slots.
- **Resource Allocation**: In a time-sharing system, the CPU time is divided into small time intervals called time slices or quantum. Each user or process is allocated a time slice during which they can execute their commands or programs.
- **Examples**: Modern operating systems like Unix, Linux, Windows, and macOS are time-sharing systems, allowing multiple users to run applications concurrently on the same computer.

5. Explain computer system architecture?

Computer system architecture, also known as computer organization and architecture, refers to the design and structure of a computer system, including its hardware components and how they interact to perform tasks and execute instructions. It encompasses various aspects of a computer's design, including the CPU (Central Processing Unit), memory hierarchy, input/output systems, and communication between different hardware components. Here are key components and concepts within computer system architecture:

Central Processing Unit (CPU):

- The CPU is the brain of the computer and is responsible for executing instructions.
-
    - The CPU fetches, decodes, and executes instructions from memory in a repetitive cycle known as the instruction cycle.
- **Memory Hierarchy**:
    - Computer systems typically have multiple levels of memory hierarchy, including registers, cache, RAM (Random Access Memory), and secondary storage devices (e.g., hard drives and SSDs).
    - The memory hierarchy is organized to provide fast access to frequently used data and to accommodate larger storage capacities.

Input/Output (I/O) Systems:

- I/O systems enable the computer to communicate with external devices such as keyboards, mice, monitors, printers, and network interfaces.

Buses:

- Buses are communication pathways that connect different components within the computer, including the CPU, memory, and I/O devices.
- Types of buses include the address bus (for specifying memory locations), data bus (for transferring data), and control bus (for signaling various operations).
- **Instruction Set Architecture (ISA)**:
    - The ISA defines the set of instructions that a CPU can execute, including their formats and encoding.
    - It serves as the interface between software (compiled code) and hardware (the CPU).
    - Examples of ISAs include x86, ARM, and MIPS.
- **Pipelining**:
    - Pipelining is a technique used in modern CPUs to overlap the execution of multiple instructions in different stages of the instruction cycle.
    - It improves instruction throughput and CPU performance by allowing several instructions to be in different stages of execution simultaneously.
- **Caches**:
    - Caches are small, high-speed memory units that store frequently accessed data and instructions to reduce memory access times.
    - Levels of cache (L1, L2, etc.) are placed between the CPU and main memory to improve overall system performance.
- **Parallel Processing**:
    - Parallel processing involves the use of multiple CPUs or CPU cores to perform tasks concurrently, increasing computational power.
    - Symmetric Multiprocessing (SMP) and multi-core processors are common implementations of parallel processing.
- **Instruction Pipelines**:
    - Instruction pipelines break down the execution of instructions into stages, allowing for the simultaneous processing of multiple instructions.
    - Common pipeline stages include fetch, decode, execute, and write-back.
- **Memory Management**:
    - Memory management techniques, such as virtual memory and paging, are used to efficiently allocate and manage memory resources for processes and applications.

- **I/O Controllers and Peripherals**:
  - I/O controllers manage communication between the CPU and various peripheral devices, including disks, network cards, and graphics adapters.
- **System Interconnects**:
  - High-performance computers may use specialized interconnects (e.g., PCIe for expansion cards or InfiniBand for high-speed networking) to connect different components within the system.

Understanding computer system architecture is essential for computer engineers, software developers, and system administrators as it forms the foundation for designing, building, and optimizing computer systems to meet specific performance, efficiency, and functionality requirements.

6. Explain about system calls?

System calls provide the interface between a process and the operating system. When a system call is executed, it is treated as by the hardware as software interrupt.

- A system call is a mechanism that provides the interface between a process and the operating system.
- It is a programmatic method in which a computer program requests a service from the kernel of the OS.
- System call offers the services of the operating system to the user programs via API (Application Programming Interface).
- System calls are the only entry points for the kernel system.
- (Check the Qn no:1 in 15 marks section)

7. What is os user interface?

User Interface

- almost all OS have a user interface
- it can take several forms
- a.*command-line interface (CLI)*
  uses text commands and a method for entering them.

b.*batch interface*

In this, commands and directives to control those commands are entered into files, and those files are executed.

c.*graphical user interface*

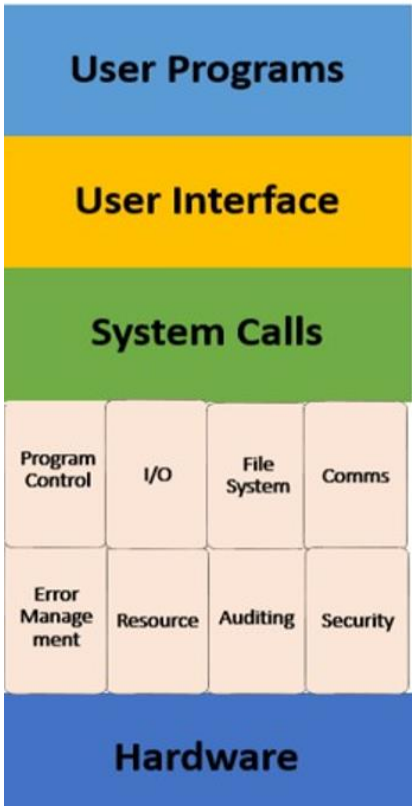
In this, the interface is a window system with a pointing device to direct I/O, choose from menu, make selections and a keyboard to enter text.

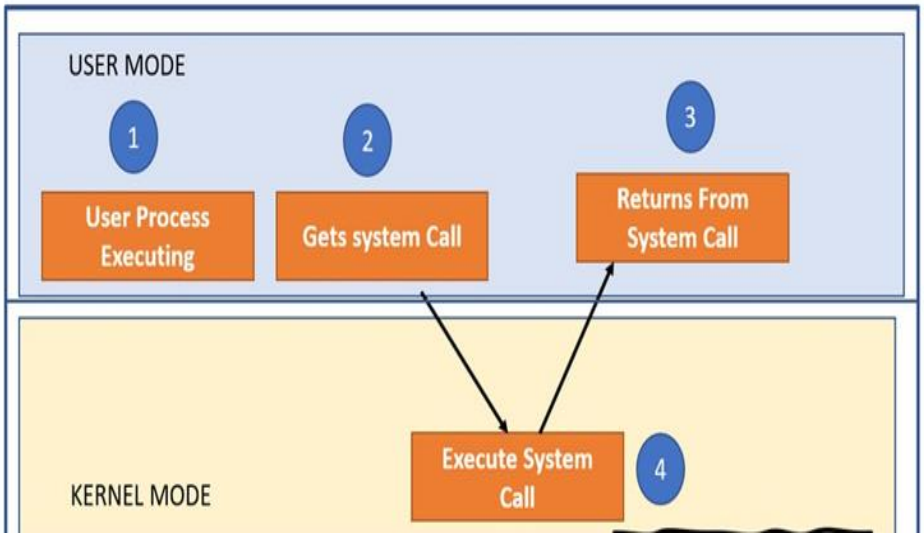
(Go through the Qn NO:4 in 15 marks section)

15 Marks

1. What is system calls in OS? Explain in detail with its types.
    - A system call is a mechanism that provides the interface between a process and the operating system.
    - It is a programmatic method in which a computer program requests a service from the kernel of the OS.
    - System call offers the services of the operating system to the user programs via API (Application Programming Interface).
    - System calls are the only entry points for the kernel system.
- How Does System Call Work?

Step 1) The processes executed in the user mode till the time a system call interrupts it.
Step 2) After that, the system call is executed in the kernel-mode on a priority basis.
Step 3) Once system call execution is over, control returns to the user mode.,
Step 4) The execution of user processes resumed in Kernel mode.





Types of System Calls

System Calls can be grouped roughly into six major categories.

* Process Control
  - end, abort
  - load, execute
  - create/terminate process
  - get process attributes, set process attributes
  - wait for time

- wait event, signal event
- allocate & free memory

* File Management
  - create/delete file
  - open, close
  - read, write, reposition
  - get file attributes/set file attributes

* Device Management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes.
  - logically attatch or detach attributes

* Information Maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get process, file or device attributes
  - set process, file or device attributes

* Communications
  - create, delete communication connection.
  - send, receive messages
  - transfer status information
  - attatch/detach remote devices

What is the use of fork and exec system calls?

Fork is a system call by which a new process is Creating. Exec is also a system call, which is used after a fork by one of the two processes to place the process memory space with a new program.

2. Discuss the Simple Operating System Structure. Describe the layered approach.

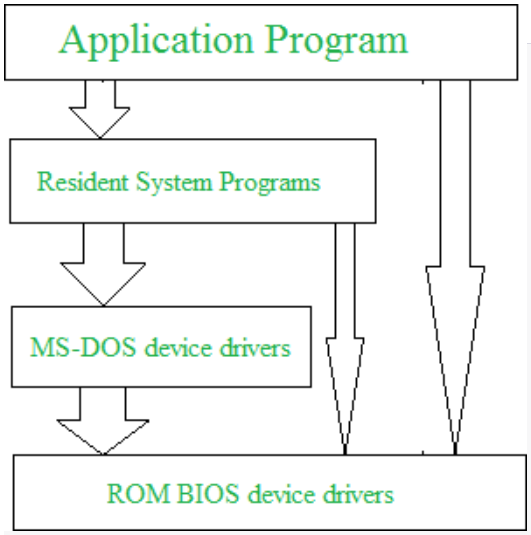
The Simple Operating System Structure and the Layered Approach are fundamental concepts in operating system design that aim to create efficient, modular, and maintainable operating systems. In summary, the Simple Operating System Structure focuses on minimalism and efficiency, while the

Layered Approach emphasizes modularity, abstraction, and flexibility.

The Simple Operating System Structure refers to a minimalistic design philosophy for building operating systems. It is characterized by the following key principles:

Monolithic Kernel: In a simple operating system structure, the entire operating system, including core services like process management, memory management, file systems, and device drivers, resides in a single monolithic kernel. This means that all kernel functions are tightly integrated into a single program or module.

1. **Limited Abstraction Layers**: Simple operating systems often have limited abstraction layers between hardware and user-level applications. They provide basic services for managing hardware resources without extensive layers of abstraction.
2. **Efficiency**: Simplicity in design typically leads to efficient code execution. Simple operating systems aim to minimize overhead and maximize the performance of critical system functions.
3. **Low Resource Requirements**: These operating systems are designed to run on hardware with limited resources. They are suitable for embedded systems, real-time systems.



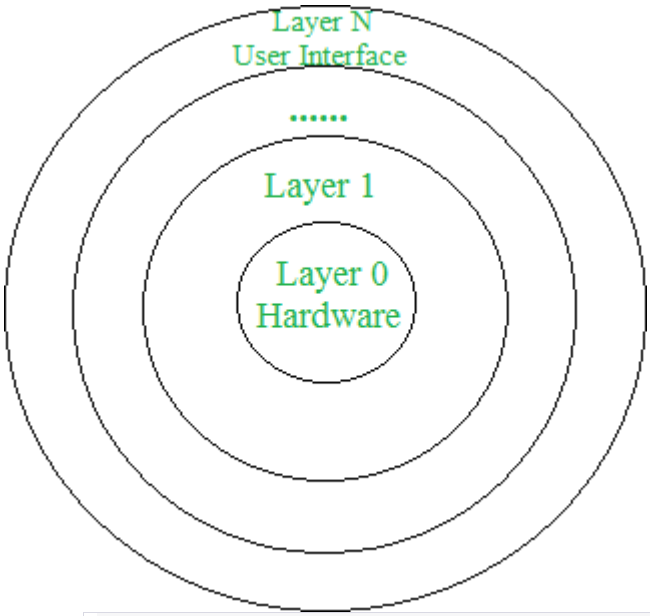
Examples of simple operating systems include early versions of MS-DOS and some embedded operating systems for microcontrollers.

Layered Approach:

The Layered Approach, on the other hand, is a design methodology used in the development of complex operating systems. It involves breaking down the operating system into multiple layers, each responsible for a specific set of functions. These layers are organized hierarchically, with higher layers relying on lower layers to provide services. Key characteristics of the layered approach include:

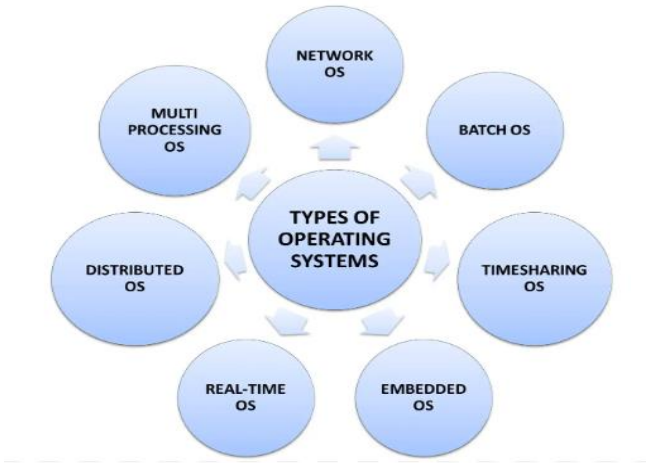
1. **Modularity**: Each layer encapsulates a specific set of functionalities, making the operating system more modular and easier to understand and maintain.
2. **Abstraction**: Layers provide abstraction to higher-level layers. For example, the file system layer abstracts away details of storage devices, allowing application programs to interact with files without needing to know the hardware details.

3. **Isolation**: Layers are typically isolated from each other. This isolation helps contain errors or issues within a particular layer, preventing them from affecting other parts of the operating system.
4. **Ease of Development**: Multiple teams or developers can work on different layers simultaneously, making development more manageable and parallelizable.
5. **Flexibility**: The layered approach allows for easy extension and modification of the operating system. New layers or functionality can be added without disrupting existing layers, assuming proper interfaces are maintained.



Modern operating systems like Unix/Linux, Windows, and macOS often adopt the layered approach

3. What are different types of operating system? Explain them in detail.
   Differentiate Real-time, Distributed and Time-Sharing operating systems. (2022-4 marks)
   Distinguish between batch systems and time-sharing systems.



Batch System	Time sharing system
Jobs or work is keep in order and jobs are run one after the other	The tasks are given specific time and operating system switches between different tasks.
there won't be any user interactions	user interaction is involved in the processing

1. **Single-User, Single-Tasking Operating System**:
    - This is the simplest type of operating system, where only one user can use the computer at a time, and the computer can run only one application or task at a time.
    - Examples include early personal computer operating systems like MS-DOS and CP/M.
2. **Single-User, Multi-Tasking Operating System**:
    - In this type of operating system, a single user can run multiple applications or tasks concurrently.
    - The operating system switches between tasks quickly, giving the appearance of simultaneous execution.
    - Examples include early versions of Windows (e.g., Windows 3.1) and macOS.
3. **Multi-User Operating System**:
    - Multi-user operating systems support multiple users who can log in and use the system simultaneously.
    - Each user has their own session and can run their own set of applications independently.
    - Examples include Unix, Linux, and modern versions of Windows and macOS.
4. **Real-Time Operating System (RTOS)**:
    - RTOSs are designed for systems that require precise and predictable timing, such as embedded systems, robotics, and industrial control systems.
    - They provide deterministic behavior, ensuring that tasks are executed within specified time constraints.
    - Examples include VxWorks and QNX.
5. **Network Operating System (NOS)**:
    - NOSs are designed for managing and sharing network resources, such as file servers and printers, in a networked environment.
    - They facilitate network communication and resource sharing among multiple users or devices.
    - Examples include Novell NetWare and Windows Server.
6. **Distributed Operating System**:
    - Distributed operating systems are designed to manage and coordinate the activities of a network of interconnected computers or nodes.
    - They provide a unified view of the network's resources and enable efficient

communication and data sharing across multiple machines.
- Examples include Amoeba and Plan 9 from Bell Labs.

7. **Mobile Operating System**:
   - Mobile operating systems are specifically designed for smartphones, tablets, and other mobile devices.
   - They offer touch-based interfaces, power management features, and support for mobile app ecosystems.
   - Examples include Android, iOS, and Windows Mobile (now deprecated).
8. **Multi-Processor Operating System**:
   - Multi-processor operating systems are optimized to run on computers with multiple processors or CPU cores.
   - They manage the efficient allocation of tasks to multiple processors, improving system performance.
   - Examples include modern versions of Windows, Linux, and macOS.
9. **Embedded Operating System**:
   - Embedded operating systems are tailored for embedded systems, which are specialized computing devices integrated into other products or systems.
   - They are often lightweight, real-time, and optimized for specific hardware.
   - Examples include FreeRTOS and Embedded Linux.
10. **Virtual Machine Monitor (VMM) or Hypervisor**:
    - Hypervisors are used for virtualization, allowing multiple virtual machines (VMs) to run on a single physical machine.
    - They provide isolation between VMs and enable efficient resource sharing.
    - Examples include VMware ESXi, Microsoft Hyper-V, and KVM.

11. Batch operating system

Users of batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. Thus, the programmers left their programs with the operator. The operator then sorts programs into batches with similar requirements.

Problems with Batch Systems are following.

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices is slower than CPU.
- Difficult to provide the desired priority.

Real-Time OS	**Distributed OS**	**Time -Sharing OS**
Real-Time Operating System (RTOS): • Purpose: Real-time operating systems are designed for applications that require precise and predictable response times to external events. These systems are used in scenarios where	Purpose: Distributed operating systems are designed to manage and coordinate a network of interconnected computers, providing a unified view of resources and services across the network.	Purpose: Time-sharing operating systems, also known as multi-user or multitasking systems, are designed to provide efficient resource sharing and task scheduling among multiple users or processes. They aim to maximize CPU utilization and ensure fair access to resources.

meeting strict deadlines is critical.		
• Scheduling: RTOSs use real-time scheduling algorithms that prioritize tasks based on their deadlines. The focus is on ensuring that tasks are completed within their specified time constraints.	Location: The components of a distributed operating system can be physically distributed across multiple machines, often connected through a network.	Scheduling: Time-sharing systems employ scheduling algorithms that allow multiple processes to run concurrently, sharing the CPU in a time-sliced manner.
Examples of Use: Embedded systems, industrial automation, avionics, automotive control systems, medical devices, and robotics, where tasks like sensor data processing, control, and monitoring need to occur in real-time.	Communication: These systems emphasize communication and coordination between distributed components and provide features for transparent access to resources located on various machines.	User Interaction: These systems prioritize interactive user tasks, enabling users to run multiple programs simultaneously while sharing system resources.
	Examples of Use: Distributed computing environments, cloud computing platforms, peer-to-peer networks, and clustered systems that require distributed processing, data sharing, and fault tolerance.	Examples of Use: General-purpose personal computers, servers, and mainframes where multiple users or processes need to run concurrently, such as in a typical office or data center environment.

4. Explain User Operating-System Interface in detail.

The User Operating-System Interface (UOSI), also known as the User Interface (UI) or simply the user interface of an operating system, is the means by which users interact with and control the operating system and its associated software and services. It serves as the bridge between human users and the underlying operating system, enabling users to perform various tasks, configure system settings, and run applications.

1. **Command-Line Interface (CLI)**:
   - In a CLI, users interact with the operating system by typing text-based commands into a terminal or command prompt.
   - Users issue commands to perform tasks such as file manipulation, process management, and system configuration.
   - Examples include the Windows Command Prompt, macOS Terminal, and Unix/Linux shells like Bash.
2. **Graphical User Interface (GUI)**:

   A GUI provides a visual and interactive way for users to interact with the operating system.
   - It uses graphical elements such as windows, icons, menus, and buttons to represent and control system functions and applications.
   - Common GUI environments include the Windows Desktop, macOS Aqua, and various Linux desktop environments like GNOME and KDE.

3. **Desktop Environment**:
    - A desktop environment is a complete GUI system that includes not only the interface but also tools and utilities for managing windows, files, and settings.
    - It often includes features like a file manager, taskbar, and system tray.
    - Examples include Windows Explorer (on Windows), Finder (on macOS), and Nautilus (on GNOME-based Linux systems).
4. **Start Menu/Dock**:
    - The Start Menu (on Windows) or Dock (on macOS) is a central hub for accessing applications, files, and system settings.
    - Users can launch programs, access recent files, and search for resources through these menus.
5. **File Manager**:
    - A file manager provides users with a graphical way to navigate, organize, and manipulate files and directories.
    - It often includes features like copy, paste, move, delete, and file property information.
    - Examples include Windows File Explorer, macOS Finder, and Nautilus.
6. **System Preferences/Control Panel**:
    - Users can configure system settings and customize their operating system through the System Preferences (on macOS) or Control Panel (on Windows).
    - Settings related to display, sound, network, security, and more can be adjusted here.
7. **Application Launchers**:
    - Application launchers provide an easy way to find and launch installed software applications.
    - They often include search functionality and categorization of applications.
    - Examples include the Start Menu search on Windows and the Spotlight search on macOS.
8. **Taskbar/Menu Bar**:
    - The taskbar (on Windows) or menu bar (on macOS) provides access to running applications, system notifications, and system status information.
    - It typically displays icons for open applications and may include a system clock, notification area
9. **Context Menus**:
    - Context menus appear when users right-click on objects such as files, folders, or desktop icons, providing access to actions specific to the selected item.
10. **Keyboard Shortcuts**:
    - Operating systems often provide keyboard shortcuts to perform various tasks quickly without using the mouse. For example, Ctrl+C to copy and Ctrl+V to paste in Windows.
11. **Accessibility Features**:
    - Operating systems offer accessibility features such as screen readers, magnification, and keyboard shortcuts to assist users with disabilities.

12**Notification System**:
    - Notifications inform users about system events, messages, updates, or application-specific alerts. Users can often interact with notifications directly.

13Help and Documentation:
    - Operating systems provide access to help resources, including user manuals, tutorials, and online documentation, to assist users in using the system effectively.

5. Explain operating system functions and services in detail.

(Please go through the Qn No:1,2,4,5 in first section and Qn no:1 in 4 marks section)

OPERATING-SYSTEM SERVICES :

An operating system provides an environment for the execution of programs. It provides certain services to programs and to the users of those programs.

i. User interface: Almost all operating systems have a user interface(UI).This interface can take several forms. One is a command-line interface(CLI), which uses text commands and a method for entering them (say, a keyboard for typing in commands in a specific format with specific options). Another is a batch interface, in which commands and directives to control those commands are entered into files, and those files are executed. Most commonly, a graphical user interface (GUI) is used. Here, the interface is a window system with a pointing device to direct I/O, choose from menus, and make selections and a keyboard to enter text. Some systems provide two or all three of these variations.

ii. Program execution: The operating system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally indicating error.

iii. I/O operations: A running program may require I/O operations. This I/O operation may be to access a file or an I/O device. For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O operations.

iv. File-system manipulation: The programs must be able to read and write files and directories. They also, need to create and delete them by name, search for a given file, list file information. Some operating systems include permissions management to allow or deny access to files or directories based on file ownership.

v. Communications: There are many circumstances in which one process needs to exchange information with another process. Such communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems tied together by a computer network. Communications may be implemented via shared memory, in which two or more processes read and write to a shared section of memory, or message passing, in which packets of information in predefined formats are moved between processes by the operating system.

vi. Error detection: The operating system needs to be detecting and correcting errors constantly. Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (such as a parity error on disk, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too-great use of CPU time). For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.

vii. 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 CPU cycles, main memory, and file storage. There may also be routines to allocate printers, USB storage drives, and other peripheral devices.

viii. Accounting: OS keeps 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) or simply for accumulating usage statistics. Usage statistics may be a valuable tool for researchers who wish to reconfigure the system to improve computing services.

ix. Protection and security: Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access to system resources.

MODULE-II

PROCESS MANAGEMENT

2 Marks

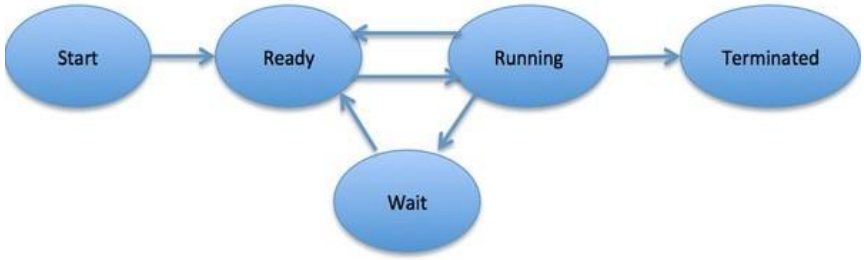
1. Define process? (2022-1mark)

- Process is the execution of a program that performs the actions specified in that program.
- It can be defined as an execution unit where a program runs.
- The OS helps you to create, schedule, and terminate the processes which are used by the CPU.
- A process created by the main process is called a child process.
- Process operations can be easily controlled with the help of PCB(Process Control Block).

2. What is meant by the state of the process?

Process States

As a process executes, it changes state. The state of a process is defined as the current activity of the process. Process can have one of the following five states at a time.



New

The process is being created.

Ready

The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.

Running

Process instructions are being executed (i.e. The process that is currently being executed).

Waiting

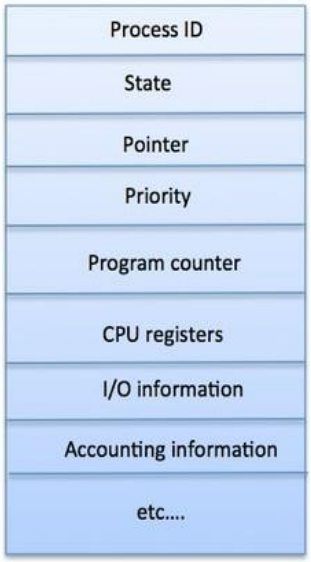
The process is waiting for some event to occur (such as the completion of an I/O operation).

Terminated

The process has finished execution.

3. What does PCB contain?
   What is PCB? (2022-2 marks)

- **Process Control Block (PCB)**
  - A Process Control Block is a data structure maintained by the Operating System for every process.
  - The PCB is identified by an integer process ID (PID).
  - There is a Process Control Block for each process, enclosing all the information about the process.
  - It is also known as the task control block. It is a data structure, which contains the following:



  - **Process ID:** Unique identification for each of the processes in the operating system.
  - **Process state:** A process can be new, ready, running, waiting, etc.
  - **Pointer :**A pointer to the parent process.
  - **Program counter:** The program counter lets you know the address of the next instruction, which should be executed for that process.
  - **CPU registers:** This component includes accumulators, index and general-purpose registers, and information of condition code.
  - **CPU scheduling information:** This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.
  - **Accounting and business information:** It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.
  - **Memory-management information:** This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.
  - **I/O status information:** This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

4.What is context switching?

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process.

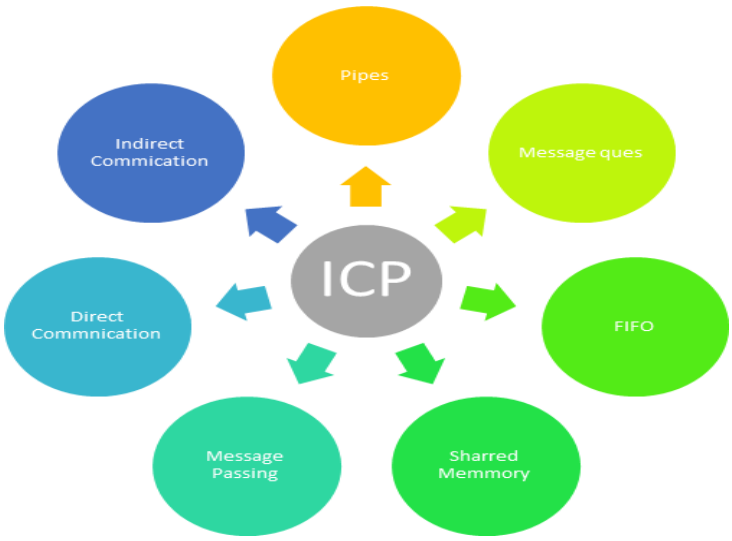
This task is known as a Context Switch. Context switching is a fundamental operation in modern operating systems that allows the operating system to efficiently manage multiple processes or threads by switching the CPU from one task to another. During context switching, the system saves the state of the currently executing process or thread and loads the state of the next process or thread that will run. This allows for multitasking, where multiple processes or threads appear to run concurrently on a single CPU.

5. Define Interprocess communication.(2022- 2 marks)

Interprocess communication provides a mechanism to allow the co-operating process to communicate with each other and synchronies their actions without sharing the same address space. It is provided a message passing system.

Inter-process communication: Shared memory systems, Message Passing

- Inter process communication (IPC) is used for exchanging data between multiple threads in one or more processes or programs.
- The Processes may be running on single or multiple computers connected by a network.



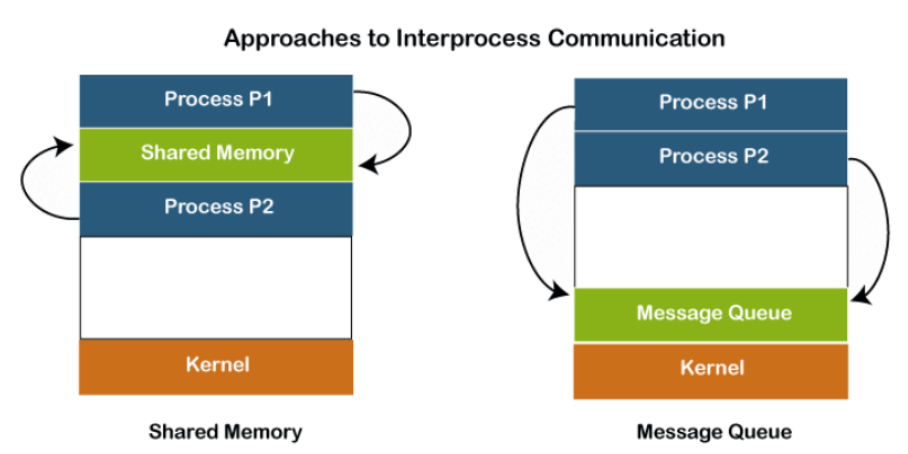
IPC (Inter-Process Communication) in an operating system refers to the mechanisms and techniques used to allow processes or threads to communicate and share data with each other. IPC is essential for coordinating and synchronizing the activities of concurrent processes and is used for various purposes, such as sharing data, signaling, and coordination. There are several methods

for IPC, and the choice of method depends on the specific requirements and constraints of the application. Here are some common methods of IPC:

There are a number of applications where processes need to communicate with each other. Processes can communicate by passing information to each other via shared memory or by message passing.
Files : Files are the most obvious way of passing information. One process writes a file, and another reads it later. It is often used for IPC.
Shared Memory: When processes communicate via shared memory they do so by entering and retrieving data from a single block of physical memory that designated as shared by all of them. Each process has direct access to this block of memory.
Message Passing : Message passing is a more indirect form of communication. Rather than having direct access to a block of memory, processes communicate by sending and receiving packets of information called messages.

These messages may be communicated indirectly or directly.
Indirect message passing is done via a mailbox.
Direct message passing is done via a link between the two communicating processes.
In both cases the messages themselves are sent via the operating system. The processes do not have direct access to any memory used in the message passing process.

Synchronization is often necessary when processes communicate. Processes are executed with unpredictable speeds. Yet to communicate one process must perform some action such as setting the value of a variable or sending a message that the other detects. This only works if the events perform an action or detect an action are constrained to happen in that order. Thus one can view synchronization as a set of constraints on the ordering of events. The programmer employs a synchronization mechanism to delay execution of a process in order to satisfy such constraints.



1.Message Passing:

- **Synchronous Messaging:** In this approach, the sender blocks until the message is received and processed by the receiver.
- **Asynchronous Messaging:** The sender sends the message and continues its execution without waiting for the receiver's response.
- **Named Pipes:** These are named channels that processes can use to send and receive messages.

2. **Shared Memory:**
    - Processes can create shared memory segments in which they can read from and write to like ordinary memory. This method is typically used for high-performance data exchange but requires careful synchronization to avoid data corruption.

3. **Sockets:**
    - Sockets are a network-based IPC method often used for communication between processes on different machines. They can also be used for IPC on the same machine (Unix domain sockets).
4. **Signals:**
    - Signals are a way for processes to notify each other about specific events or to request the termination of a process. Signals can be used for simple IPC and are often used for error and exception handling.
5. **File System:**
- Processes can use files as a means of communication. One process writes data to a file, and another process reads from it. This is a simple method of IPC, but it may not be as efficient as other methods.

6. **Semaphores:**
    - Semaphores are a synchronization primitive used to coordinate the access to shared resources. They can be used for signaling and coordination between processes.
7. **Message Queues:**
    - Message queues provide a message-based communication mechanism where processes can send and receive messages in a first-in, first-out (FIFO) order.
8. **Remote Procedure Calls (RPC):**
    - RPC allows a process to invoke a procedure or function in a remote process as if it were a local procedure call. This is often used for client-server communication and distributed systems.
9. **Memory-Mapped Files:**
    - Processes can map a file into their address space, allowing them to read and write to the file as if it were memory. This method is useful for sharing data between processes.

6. What are the 3 different types of scheduling queues?

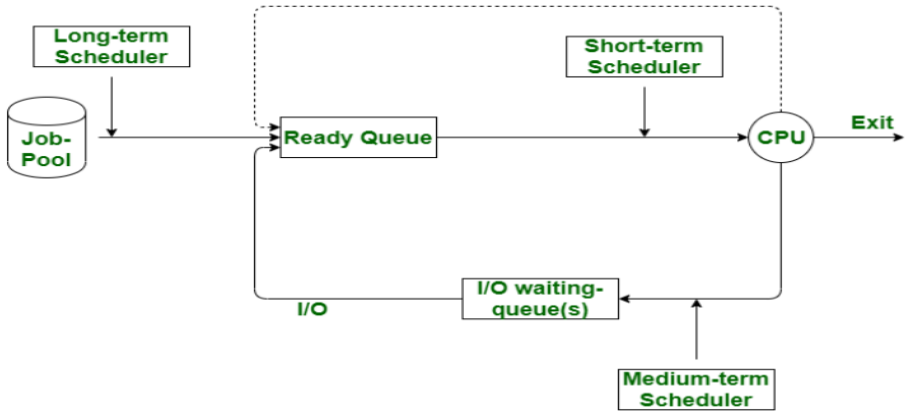
Job Queue: As process enters the system they are put into job queue.
Ready Queue: The processes that are residing in the main memory and are ready and waiting to execute are kept in the queue.
Device Queue: The list of processes waiting for particular I/O device is called a device queue.

7. Define schedulers?

Schedulers in an operating system are responsible for managing and allocating system resources, particularly CPU time, to various processes or threads in a way that maximizes system efficiency and ensures fair and timely execution of tasks. There are different types of schedulers, each serving a specific purpose in the overall process of process scheduling.

A process migrates between the various scheduling throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler.

8.Differentiate different types of schedulers. (2022-2 marks)



Long-Term Scheduler (Job Scheduler):

- The long-term scheduler selects processes from a pool of submitted processes and loads them into memory for execution. It decides which processes should be admitted to the system based on various criteria, such as system load and memory availability.
- This scheduler controls the degree of multi-programming, determining how many processes can be in memory at the same time. Its goal is to maintain a balance between system performance and resource utilization.

Short-Term Scheduler (CPU Scheduler):

- The short-term scheduler, also known as the CPU scheduler, is responsible for selecting which process or thread from the pool of ready processes should be executed next. It decides which process should get the CPU time slice.
- This scheduler operates at a high frequency, making rapid decisions on which process should be given access to the CPU to achieve fair and efficient utilization of CPU resources.

Medium-Term Scheduler:

- The medium-term scheduler is responsible for managing processes that are in a blocked or suspended state. It can decide to swap a process out of memory (onto disk) to free up memory resources for other processes. Later, it can bring the swapped-out process back into memory when needed.
- This scheduler helps prevent situations where the system runs out of available memory, ensuring that the system remains responsive.

9.Name some classic problem of synchronization?

Synchronization problems in operating systems refer to issues and challenges that arise when multiple processes or threads access shared resources concurrently and need to coordinate their activities to prevent conflicts and maintain data integrity. These problems can lead to issues like race conditions, deadlock, and priority inversion.

- The Bounded – Buffer Problem
- The Reader – Writer Problem
- The Dining –Philosophers Problem
- Producer-Consumer Problem

10.What is the use of cooperating processes?

Cooperating processes in an operating system are processes that can interact or communicate with each other to achieve a common goal or to exchange information. This cooperation can be done through various inter-process communication (IPC) mechanisms and synchronization methods.
Cooperating processes are essential for many applications, including operating systems, server-client communication, parallel computing, and distributed systems.
Concurrent process executing in the operating system may be either independent process or cooperating process. A process is cooperating if it can affect or be affected by the process executing in the system. Any process that shares data with other process is a cooperating process.
Advantage of Cooperating process
- Information sharing: Several users may be interested in the same piece of information.
- Computation speedup: If we want particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with others.
- Modularity: Dividing the system functions into separate process or threads.
- Convenience: Even an individual user may have many task on which to work at one time.

11.Define race condition.

When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

A race condition is a situation in which the behavior of a computer program depends on the relative timing of events, such as the order in which threads or processes execute. In a race condition, the final outcome of a program is unpredictable and can lead to incorrect results or system instability. Race conditions typically occur when multiple threads or processes access shared resources concurrently without proper synchronization.

12.What is critical section? What are the requirements that a solution to the critical section problem must satisfy?

A critical section is a part of a program or code segment in which multiple concurrent processes or threads access shared resources or variables. In a critical section, only one process or thread can execute at a time, ensuring that the shared resources are accessed in a controlled and mutually exclusive manner. The concept of a critical section is essential for managing concurrent access to shared resources to avoid data corruption and race conditions.
Processes that are working together often share some common storage that one can read and write. The shared storage may be in main memory or it may be a shared file. Each process has segment of code, called a critical section, which accesses shared memory or files. The key issue involving shared memory or shared files is to find way to prohibit more than one process from reading and writing the shared data at the same time.
Two processes are executing in their critical sections at the same time. The criticalsection problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of code implementing this request is the entry section. The critical section may be followed by an exit section. The remaining code is the remainder section. The general structure of a typical process P is shown below:

```
do
{
ENTRY SECTION
        Critical section
EXIT SECTION
        Remainder section
}while(TRUE);
```

A solution to the critical section problem must satisfy the following three requirements
1.**Mutual Exclusion:** If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.
2. **Progress:** If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.
3. **Bounded Waiting :**A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

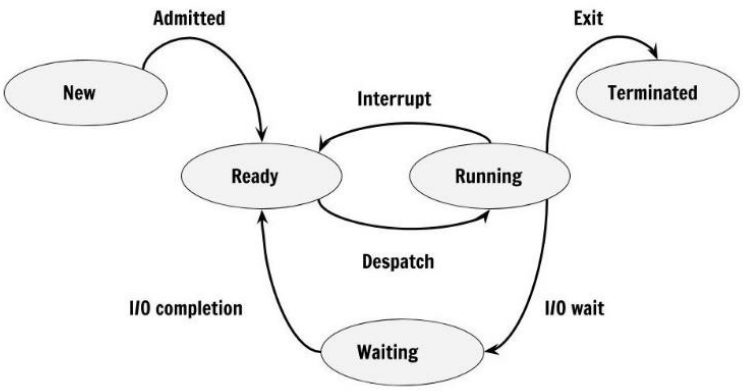
13.Define entry section and exit section.

Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section.

4 Marks

1. Explain different process states(4 marks)

In an operating system, a process goes through several different states during its execution. These process states help the operating system manage and control the execution of multiple processes efficiently. The exact number and names of these states can vary depending on the operating system, but the following are the common process states:



1. **New:** This is the initial state of a process. In this state, the operating system creates a process control block (PCB) to store information about the process and assigns it a process ID (PID). The process is then ready to be loaded into memory but has not yet started execution.
2. **Ready:** After a process is created, it enters the ready state. In this state, the process is waiting for the CPU to be allocated so that it can start executing. Multiple processes in the ready state can compete for the CPU.
3. **Running:** When a process is assigned the CPU for execution, it enters the running state. In this state, the process's instructions are being executed by the CPU.
4. **Blocked (Waiting):** A process can enter the blocked state when it is waiting for a particular event to occur, such as I/O operations or the completion of a child process. While in this state, the process cannot execute further until the event it is waiting for occurs.
5. **Terminated:** A process enters the terminated state when it has finished its execution. The operating system releases the resources associated with the process, and its PCB is removed.

2.Explain about process scheduling?

1.1 Process Scheduling - Basic concepts, Scheduling Criteria

- **Process Scheduling**
  - The act of determining which process is in the ready state, and should be moved to the running state is known as Process Scheduling.
  - The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs.
  - For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of the CPU.
  - Scheduling fell into one of the two general categories:

Differentiate pre-emptive and non pre-emptive scheduling. (2022- 1 mark)

Non Preemptive Scheduling: When the currently executing process gives up the CPU voluntarily. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time.

Preemptive Scheduling: When the operating system decides to favour another process, pre-empting the currently executing process. Preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

The process scheduling is the activity of the process manager that handles the removal of the running process from and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

Scheduling Queues

Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. The operating system

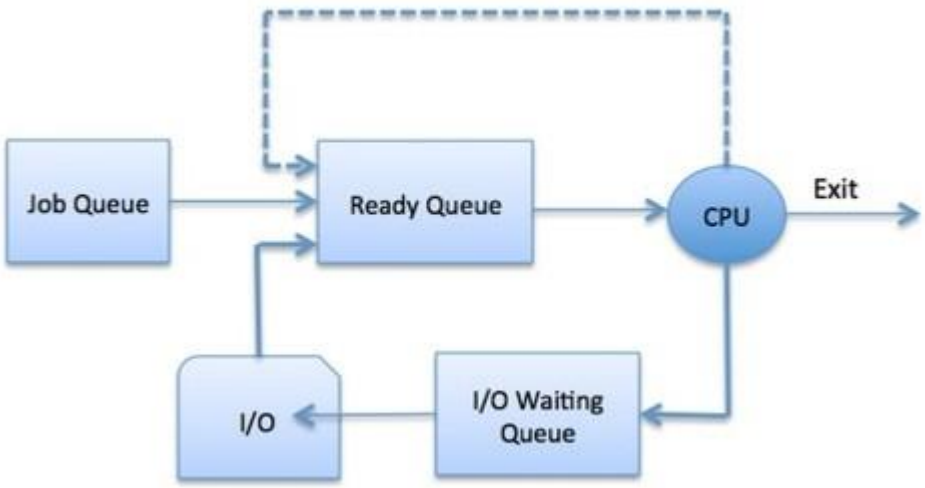
also maintains other queues such as device queue. Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has its own device queue.

This figure shows the queuing diagram of process scheduling.

Queue is represented by rectangular box.

The circles represent the resources that serve the queues. The arrows indicate

the process flow in the system.



3.Define semaphores. Explain the working of Semaphore. (2022-4 marks)

Semaphore A non-computer meaning of the word semaphore is a system or code for sending signals. To overcome the difficulty in critical section we use a synchronization tool called as semaphore. A semaphore is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: wait and signal.

wait(s) { while (S<=0) { /*do nothing*/ S= S-1; } signal(S) { S = S + 1; }

Semaphore Usage:

We can use semaphore with the n process critical section problem. The n process share a semaphore, mutex (standing for mutual exclusion), initialized to one. We can also use semaphore to solve various synchronization problems,. For eg: consider two concurrently running processes: p1 with a statement s1 and p2 with a statement s2. Suppose we require that s2 be executed only after s1 has completed.

Semaphore mutex;

Initially

mutex = 1

do

{

wait (mutex);

critical section

signal (mutex);

remainder section

} while(true);

4.Briefly explain virtual machines?

Virtual Machine is a completely separate individual operating system installation on your usual operating system. It is implemented by software emulation and hardware virtualization.

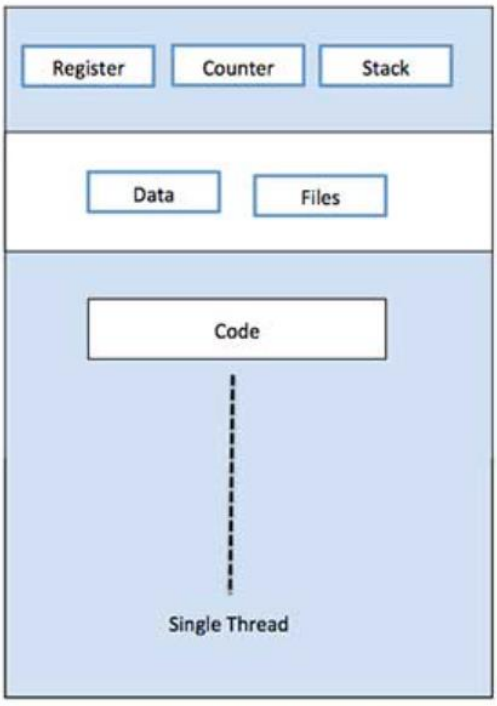
5. Define Thread and explain advantages of threads?

What is a Thread?

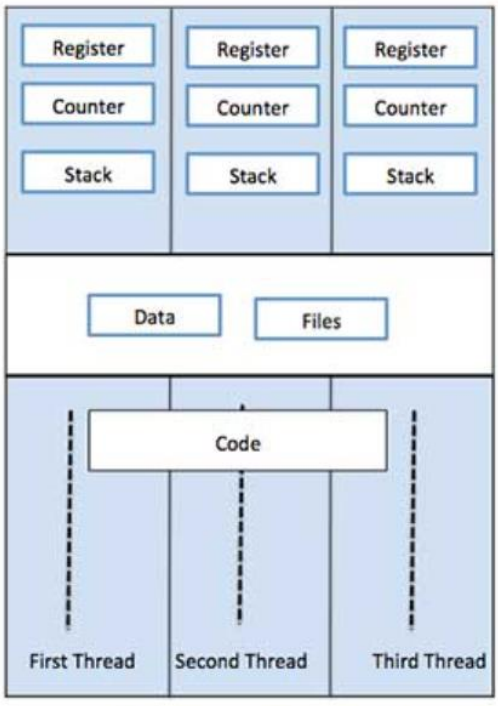
- A thread is a path of execution within a process.
- A process can contain multiple threads.

Threads are lightweight processes(LWP), is a basic unitof CPU utilization. It comparises a thread ID, a program mcounter, a register set, and a stack. They improve performance by weakening the process abstraction. A process(heavy weight) is one thread of control executing one program in one address space. A thread may have multiple threads of control running different parts of a program in one address space.

- Why Multithreading?
  - A thread is also known as a lightweight process.
  - The idea is to achieve parallelism by dividing a process into multiple threads.
  - For example, in a browser, multiple tabs can be different threads.
  - MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc.



Single Process P with single thread



Single Process P with three threads

Single-threading is the processing of one command/ process at a time. Whereas multi threading is a widespread programming and execution model that allows multiple threads to exist within the context of one process. These threads share the process's resources, but are able to execute independently.

THE BENEFITS OF MULTITHREADED PROGRAMMING

Responsiveness: Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.This quality is especially useful in designing user interfaces.For instance, consider what happens when a user clicks a button that results in the performance of a time-consuming operation.

Resource sharing: Processes can only share resources through techniques such as shared memory and message passing. Such techniques must be explicitly arranged by the programmer. However, threads share the memory and there sources of the process to which they belong by default. The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.

Economy: Allocating memory and resources for process creation is costly. Because threads share the resources of the process to which they belong, it is more economical to create and context-switch threads.

Scalability: The benefits of multithreading can be even greater in a multiprocessor architecture

- Types of Threads: There are two types of threads.
  - User Level Thread
  - Kernel Level Thread

Differentiate between process and threads

Process	Thread
Process is heavy weight or resource intensive.	Thread is light weight taking lesser resources than a process.
Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
In multiple processing environments each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
If one process is blocked then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, second thread in the same task can run.
Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
In multiple processes each process operates independently of theothers.	One thread can read, write or change another thread's data.

6.Explain the scheduling criteria

Many criteria have been suggested for comparing CPU-scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

• **CPU utilization:** We want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily loaded system).
• **Throughput:** One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be ten processes per second.
• **Turnaround time:** The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
• **Waiting time:** The amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.
• **Response time:** Response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

7.Explain about different multithreading models.

Multithreading model are of three types.

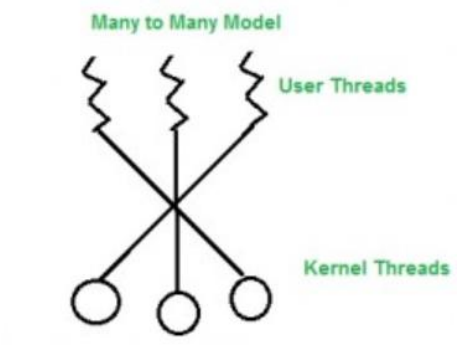
Many to many model.

Many to one model.

one to one model.

Many to Many Model
In this model, we have multiple user threads multiplex to same or lesser number of kernel level threads. Number of kernel level threads are specific to the machine, advantage of this model is if a user thread is blocked we can schedule others user thread to other kernel thread. Thus, System doesn't block if a particular thread is blocked.

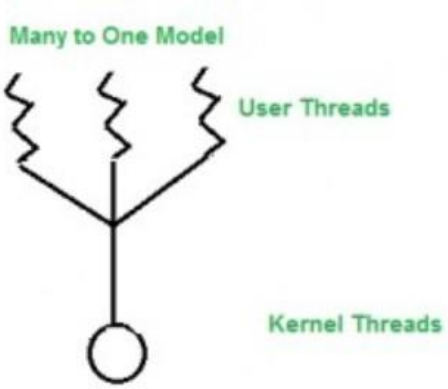
It is the best multi threading model.



Many to One Model

In this model, we have multiple user threads mapped to one kernel thread. In this model when a user thread makes a blocking system call entire process blocks. As we have only one kernel thread and only one user thread can access kernel at a time, so multiple threads are not able access multiprocessor at the same time.

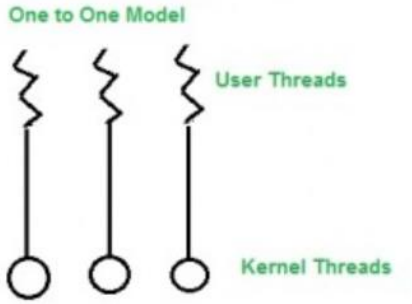
The thread management is done on the user level so it is more efficient.



One to One Model

In this model, one to one relationship between kernel and user thread. In this model multiple thread can run on multiple processor. Problem with this model is that creating a user thread requires the corresponding kernel thread.

As each user thread is connected to different kernel , if any user thread makes a blocking system call, the other user threads won't be blocked.



15 marks

1.What are the Classical problems of Synchronization? (2022- 15 marks)
What is readers-writers problem? (2022-4 marks)

Synchronization problems in operating systems refer to issues and challenges that arise when multiple processes or threads access shared resources concurrently and need to coordinate their activities to prevent conflicts and maintain data integrity. These problems can lead to issues like race conditions, deadlock, and priority inversion.

The Bounded – Buffer Problem: Producer-Consumer Problem
The Reader – Writer Problem
The Dining –Philosophers Problem

1.The Bounded-Buffer Problem

The bounded-buffer problem is a classic example of concurrent access to shared resources. A bounded buffer lets multiple producers and multiple consumers to share a single buffer. Producer writes data to the buffer and consumer reads data from the buffer.

- Producer must block if the buffer is full
- Consumer must block if the buffer is empty

Two counting semaphores are used for this. Use one semaphore empty to count the empty slots in the buffer.

¬ Initialize the semaphore to N
¬ A producer must wait on this semaphore before writing to the buffer
¬ A consumer will signal this semaphore after reading from the buffer Semaphore called full to count the number of data items in the buffer:
¬ Initialize the semaphore to 0
¬ A consumer must wait on this semaphore before reading from the buffer
¬ A producer will signal this semaphore after writing to the buffer
In our problem, the producer and consumer processes share the following data structures:

```
int n;
semaphore mutex = 1;
semaphore empty = n;
semaphore full = 0;
```

The pool consists of n buffers, each capable of holding one item. The mutex semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to the value 1. The empty and full semaphores count the number of empty and full buffers. The semaphore empty is initialized to the value n; the semaphore full is initialized to the value 0.

Producer Consumer Problem

Producer Process

```
do
{
// produce an item
wait(empty);
wait(mutex);
//add to buffer
Signal(mutex);
Signal(full);
} while (1);
```

Consumer Process

```
do
{
Wait(full);
Wait(mutex);
//remove an item from buffer
Signal(mutex);
Signal(empty);
} while (1);
```

2.The Reader – Writer Problem

Suppose that a database is to be shared among several concurrent processes. Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database. To distinguish between these two types of processes by referring to the former as readers and to the latter as writers.

There is a shared resource which should be accessed by multiple processes. There are two types of processes. They are reader and writer. Any number of readers can read from the shared resources simultaneously, but only one writer can write to the shared resources. When a writer is writing data to the resource, no other process can access the resource. A writer cannot write to the resource if there is non zero number of readers accessing the resource at that time. The reader writer problem is used to manage synchronization so that there is no problem with the shared resource.

The semaphore mutex and wrt are initialized to 1;
read count (rc) is initialized to 0.
The semaphore wrt is common to both the reader and writer processes.
The mutex semaphore is used to ensure mutual exclusion when the variable rc is updated .The rc variabl keeps track of how many processes are currently reading the object. The semaphore wrt functions as a mutua exclusion semaphore for the writers. It is also used by the first or last reader that enters or exit the critic section.

```
semaphore mutex = 1;
semaphore wrt = 1;
int rc = 0;
```

Reader process

```
do
{
wait( mutex);
rc++;
 if(rc= =1)
wait(wrt) ;
signal(mutex);
 //read object
wait(mutex);
rc- -;
 if(rc= =0)
signal(wrt) ;
signal(mutex) ;
}
while(1);
```

writer process

```
while(1)
{
wait(wrt);
 //write object
signal(wrt);
}
```

3.Dining Philosophers Problem:

The dining philosophers problem is another classic synchronization problem which is used to evaluate situations where there is a need of allocating multiple resources to multiple processes. At any instant, a philosopher is either eating or thinking. When a philosopher wants to eat, he uses two chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place. From the problem statement, it is clear that a philosopher can think for an indefinite amount of time. But when a philosopher starts eating, he has to stop at some point of time. The philosopher is in an endless cycle of thinking and eating. When a philosopher wants to eat the rice, he will wait for the chopstick at his left and picks up that chopstick. Then he waits for the right chopstick to be available, and then picks it too. After eating, he puts both the chopsticks down. But if all five philosophers are hungry simultaneously, and each of them pickup one chopstick, then a deadlock situation occurs because they will be waiting for another chopstick forever. The possible solutions for this are:

• A philosopher must be allowed to pick up the chopsticks only if both the left and right chopsticks are available. • Allow only four philosophers to sit at the table.

That way, if all the four philosophers pick up four chopsticks, there will be one chopstick left on the table.

So, one philosopher can start eating and eventually, two chopsticks will be available.

In this way, deadlocks can be avoided.

An array of five semaphores, stick[5], for each of the five chopsticks.
The code for each philosopher looks like:

```
while(true)
{
wait(stick[i]) ;
//Mod is used because if i=5,next chopstick is 1(circular dining table)
 wait(stick[(i+1)%5]);
//eat
signal(stick[i]);
signal (stick[(i+1)%5]);
 //think
 }
```

2. Write about the various CPU scheduling algorithms. Explain with suitable examples. (2022-15 marks)
Explain FCFS scheduling algorithm with example.
Explain SJF scheduling algorithm with example.
Explain Priority scheduling algorithm with example.
Explain Round Robin scheduling algorithm with example.
Define Starvation and aging.(4 marks)

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

CPU scheduling is an essential part of the operating system's task management. It's responsible for deciding which processes or threads get access to the CPU and for how long. Various scheduling algorithms exist, each with its own characteristics, advantages, and disadvantages. Here are some common CPU scheduling algorithms:

1. **First-Come, First-Served (FCFS)**:
    1. Processes are executed in the order they arrive in the ready queue.
    2. Simple to implement but may lead to poor average turnaround time, especially if long processes arrive early.
    3. With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. The code for FCFS scheduling is simple to write and understand.
    4. Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P1	24
P2	3
P3	3

If the processes arrive in the order P1, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart, which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes:

The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2, and 27 milliseconds for process P3. Thus, the average waiting time is (0 + 24 + 27)/3 = 17 milliseconds. If the processes arrive in the order P2, P3, P1, however, the results will be as shown in the following Gantt chart:

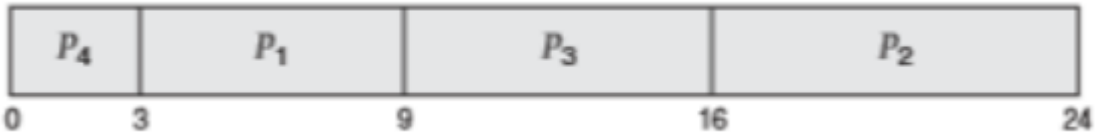
The average waiting time is now (6+0+3)/3 = 3 milliseconds. This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the processes CPU burst times vary greatly.

2. **Shortest Job Next (SJN) or Shortest Job First (SJF)**:
    1. The CPU is allocated to the process with the smallest execution time remaining.
    2. Minimizes waiting time and provides optimal turnaround time when all information is known in advance.
    3. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

4. As an example of SJF scheduling, consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Using SJF scheduling, we would schedule these processes according to the following Gantt chart:



The waiting time is 3 milliseconds for process P1, 16 milliseconds for process P2, 9 milliseconds for process P3, and 0 milliseconds for process P4. Thus, the average waiting time is (3 + 16 + 9 + 0)/4 = 7 milliseconds.

The SJF algorithm can be either preemptive or non preemptive.

3. **Round Robin (RR)**:

- Each process is assigned a fixed time slice (quantum) to execute, after which it's placed at the end of the queue.
- Provides fair access to the CPU but may result in high context switching.
- The round-robin (RR) scheduling algorithm is designed especially for time- sharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length. The ready queue is treated as a circular queue.
- The average waiting time under the RR policy is often long. Consider the following set of processes that arrive at time0, with the length of the CPU burst given in milliseconds:
- Process Burst Time
- P1 24
- P2 3
- P3 3

If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds. Since it requires another 20 milliseconds, it is preempted after the first time quantum, and the CPU is given to the next process in the queue, process P2. Process P2 does not need 4 milliseconds, so it quits before its time quantum expires. The CPU is then given to the next process, process P3. Once each process has received 1 time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is as follows:

Let’s calculate the average waiting time for this schedule. P1 waits for 6 milliseconds (10-4), P2 waits for 4milliseconds, and P3 waits for 7milliseconds. Thus, the average waiting time is 17/3 = 5.66 milliseconds.

4.Priority Scheduling:

Each process is assigned a priority, and the CPU is allocated to the highest priority process.

Can lead to starvation if lower-priority processes never get CPU time.

The SJF algorithm is a special case of the general priority-scheduling algorithm. A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order.

The larger the CPU burst, the lower the priority, and vice versa. As an example, consider the following set of processes, assumed to have arrived at time 0 in the order P1, P2, ⋯ , P5, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Using priority scheduling, we would schedule these processes according to the following Gantt chart:

The average waiting time is 8.2 milliseconds.

Starvation and aging are two important concepts in the context of priority scheduling algorithms in operating systems.

Starvation:

- Starvation occurs in priority scheduling when a process with a low priority never gets a chance to execute. This happens because higher-priority processes continually occupy the CPU, leaving lower-priority processes waiting indefinitely.
- For example, if a high-priority process is frequently arriving, lower-priority processes may be delayed or starved because the scheduler always selects the higher-priority tasks.

Aging:

- Aging is a technique used to mitigate the problem of starvation in priority scheduling algorithms. It involves increasing the priority of a process as it waits in the queue.
- The idea is that as a process waits for a long time without being scheduled, its priority gradually increases. This helps to ensure that even lower-priority processes eventually get a chance to execute.
- Aging is implemented by periodically incrementing the priority of processes in the queue.

5. Multilevel Queue Scheduling:

- The ready queue is divided into multiple priority queues, and each queue follows a different scheduling algorithm.
- Useful for systems where processes have different characteristics and requirements.

6. Multilevel Feedback Queue Scheduling:

Similar to multilevel queue scheduling, but processes can move between queues based on their behavior (e.g., if a process frequently uses the CPU, it moves to a lower-priority queue).

7.Highest Response Ratio Next (HRRN):

- Considers both the priority and the time a process has spent waiting when making scheduling decisions.
- Aims to reduce both waiting time and priority inversion problems.

8.Earliest Deadline First (EDF):

- Processes are scheduled based on their absolute deadlines.
- Commonly used in real-time operating systems to ensure timely execution of tasks.

Each scheduling algorithm has its strengths and weaknesses, and the choice of algorithm depends on the specific requirements of the system and the types of processes it needs to manage. Real-world operating systems often use a combination of these algorithms to achieve a balance between fairness, efficiency, and responsiveness.

2 Marks

1.What is deadlock?

A deadlock is a situation in which two or more processes are unable to proceed because each is waiting for the other to release a resource. Deadlocks can occur in a multi-threaded or multi-process environment, where processes compete for limited resources, such as CPU time, memory, or I/O devices. Deadlocks are a challenging problem in operating systems and concurrent programming and can lead to a system becoming unresponsive or frozen.

2.What are different methods for handling deadlocks?

The deadlock problem can be dealt with in one of the three ways:

a. Use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.

b. Allow the system to enter the deadlock state, detect it and then recover.

C. Ignore the problem all together, and pretend that deadlocks never occur in the system.

3.What is starvation? (2022-2 marks)

A problem related to deadlock is indefinite blocking or starvation, a situation where processes wait indefinitely within a semaphore. Indefinite blocking may occur if we add and remove processes from the list associated with a semaphore in LIFO order.

4.What are necessary conditions for deadlocks?

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

- Mutual exclusion
- Hold and wait
- No pre-emption
- Circular wait

5.What is race condition?

A race condition is a situation in which the behavior of a computer program depends on the relative timing of events, such as the order in which threads or processes execute. In a race condition, the final outcome of a program is unpredictable and can lead to incorrect results or system instability. Race conditions typically occur when multiple threads or processes access shared resources concurrently without proper synchronization.

4 Marks

1.What are necessary conditions for deadlocks?

The conditions necessary for a deadlock to occur are often referred to as the "Four Necessary C Deadlock":

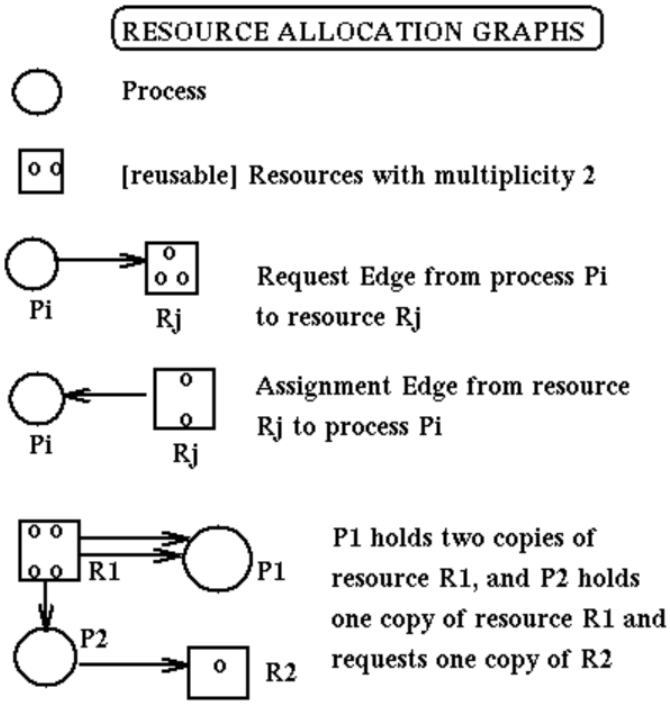
1. **Mutual Exclusion:** At least one resource must be non-shareable, meaning that only one process can use it at a time. This condition ensures that when a process acquires a resource, it has exclusive access to it.
2. **Hold and Wait:** Processes must hold at least one resource and be waiting to acquire additional resources. This condition implies that a process can request new resources while still holding onto the resources it already possesses.
3. **No Preemption:** Resources cannot be preempted, meaning that once a process holds a resource, it cannot be forcibly taken away from it until the process voluntarily releases it.
4. **Circular Wait:** There must be a circular chain of two or more processes, each waiting for a resource held by the next process in the chain. This condition creates a circular dependency, where no process can make progress.

2.Explain about resource allocation graph(RAG)?
Write the resource allocation algorithm for dead lock?

Deadlocks can be described more precisely in terms of a directed graph called a system resource allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes; P the set consisting of all active processes in the system and R the set consisting of all resource types in the system.

1. **Resource Allocation Graph:** The system can maintain a resource allocation graph that helps detect and resolve deadlocks. The graph is used to visualize resource allocation and request relationships among processes.

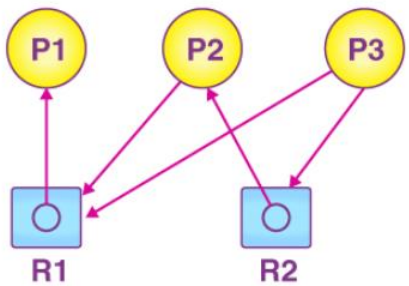
2. **Resource Allocation Policies:** Operating systems may implement policies to allocate resources in a way that prevents or reduces the likelihood of deadlocks, such as requiring processes to request all the resources they need at once or releasing all held resources if additional resources cannot be granted.



Example

Consider three processes, P1, P2, and P3, as well as two resource types, R1 and R2. Each resource has a single instance. According to the graph, P1 is using R1, P2 is holding R2 while waiting for R1, and P3 is waiting for both R1 and R2.

Because no cycle is generated in the graph, there is no deadlock.



15 Marks

1.Discuss the various Deadlock handling Techniques. (2022- 15 marks)
What is deadlock? Explain deadlock prevention in detail.(4 marks)

What is deadlock? Explain deadlock Avoidance in detail.(4 marks)

What is deadlock? Explain deadlock recovery in detail.(4 marks)
Explain Banker's algorithm with example. (2022-4 marks)
Explain Deadlock Detection.(4 marks)

Deadlock handling methods are strategies and techniques used in computer science and operating systems to address and resolve deadlock situations, which occur when two or more processes or threads are unable to proceed because they are each waiting for a resource that the other holds. Deadlocks can lead to system instability and reduced efficiency. There are several methods to handle deadlocks:

1.Deadlock Prevention:

1. **Mutual Exclusion:** Ensure that resources can only be used by one process at a time, reducing the possibility of resource contention.
2. **Hold and Wait:** A process must request and obtain all the resources it needs before it begins execution.
3. **No Preemption:** Resources cannot be forcibly taken away from a process. If a process holds a resource and needs another, it must release its current resources and request all of them again.

2.Deadlock Avoidance:

The following algorithms define the deadlock avoidance approach.

I. Resource Allocation Graph Algorithm

II. II. Banker's Algorithm

I.Banker's Algorithm: Safety Algorithm.

A mathematical approach to determine whether a resource allocation request will lead to a safe state (a state where no deadlock can occur). If the request doesn't lead to a safe state, it is denied.

The Banker's Algorithm is a resource allocation and deadlock avoidance algorithm used in operating systems. It was developed by Edsger Dijkstra in 1965. The primary purpose of the Banker's Algorithm is to determine whether allocating resources to processes in a multi-process, multi-resource system will result in a safe state (a state in which no deadlock can occur).

Here's how the Banker's Algorithm works:

(a)Initialization:

- The system starts in a known initial state where it has a fixed number of resources of each type.
- It also keeps track of the maximum demand for each resource type for each process (the maximum number of resources each process might need) and the currently allocated resources for each process.

(b)Request and Allocation:

When a process requests additional resources, the system checks if the requested resources are available.

If the requested resources are available, the system temporarily allocates them to the process and checks if the new state remains safe. To determine if the state is safe, it employs a hypothetical execution of processes to see if they can complete without causing a deadlock.

(c)Safety Check algorithm:(1 mark)

The system maintains a work vector that represents the available resources after considering the current allocations and the resources requested by the processes.

It then tries to find an execution sequence of processes in such a way that every process can finish its execution (if it is allowed to proceed) without encountering a deadlock.

If a valid sequence is found, the requested resources are granted; otherwise, the request is denied.

(d)Resource Allocation:

- If the system grants the resources, it updates the allocation and available resource vectors accordingly.

The algorithm is based on the following principles:

1.A process can request resources one at a time.

2.A process will not make a request for resources if it has not released all the resources it currently holds.

3.The algorithm keeps track of available resources and the maximum resources a process may need.

4.It checks whether allocating the requested resources will still allow for at least one safe sequence of process execution. If so, the resources are allocated; otherwise, the request is denied.

The Banker's Algorithm is an important tool for managing resources in a way that minimizes the risk of deadlocks in multi-process systems.

II.Resource Allocation Graph: A graphical representation of resources and processes, used to determine if a deadlock is possible. If a cycle is found in the graph, a deadlock may exist.

(Please go through Qn No:2 in 4 marks section)

3.Deadlock Detection and Recovery:

4. **Deadlock Detection:** Periodically check the system for deadlocks. This can be done using algorithms like the Banker's Algorithm or by maintaining a wait-for graph.
5. **Deadlock Recovery:** After detecting a deadlock, there are several ways to recover:
    - **Process Termination:** Terminate one or more processes involved in the deadlock to release resources.
    - **Resource Preemption:** Temporarily take resources away from one or more processes to break the deadlock. This requires careful planning to avoid data corruption.

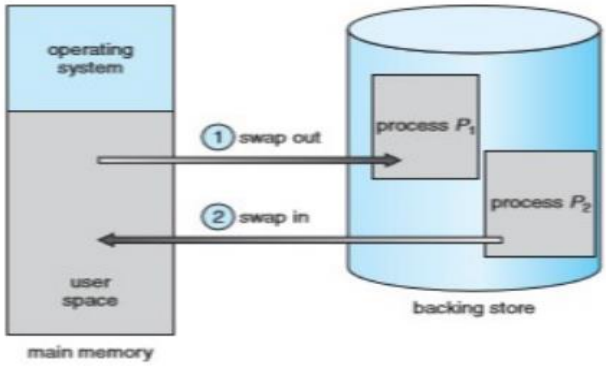
MODULE -III

MEMORY MANAGEMENT

2 Marks

1.Define Swapping. (2022-1mark)

Swapping is the act of moving processes between memory and a backing store. This is done to free up available memory. Swapping is necessary when there are more processes than available memory. A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution. A process needs to be in memory to be executed. However a process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution. This process is called swapping.



Swapping is used to implement multiprogramming in system with little hardware support for memory management. Swapping is helpful in improving processor utilization in partitioned memory environment.
When the scheduler decides to admit a new process for which no suitable free partition can be found, the swapper is invoked to vacate such a partition. The swapper is an operating system process whose major responsibility include:

- Selection of processes to swap out
- Selection of process to swap in
- Allocation and management of swap space

2.What is External Fragmentation?

As processes are loaded and removed from the memory, the free memory space is broken into little pieces. No other processes can be loaded to this free space because of its smaller size and hence, it remains unused. This problem is known as Fragmentation
Both the first-fit and best-fit strategies for memory allocation suffer from external fragmentation. As processes are loaded and removed from memory, the free memory space is broken into little pieces.

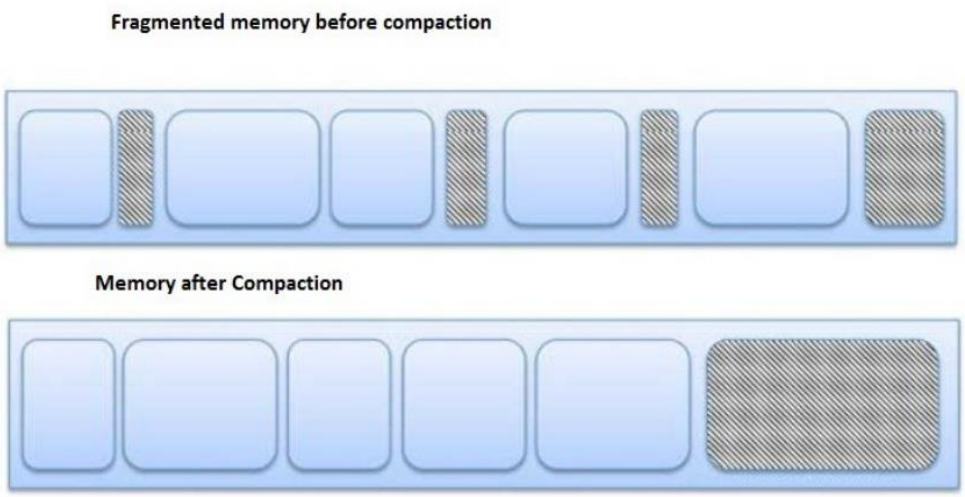
External fragmentation exists when there is enough total memory space to satisfy are quest but the available spaces are not contiguous: storage is fragmented into a large number of small holes. This fragmentation problem can be severe. Wasting of memory between partitions, due to scattering of free space into a number of discontinuous areas, is called External Fragmentation.

3.What is Internal Fragmentation?

When partitioning is static, memory is wasted in each partition where a process of smaller size than the partition is loaded .Wasting of memory within a partition, due to a difference in size of a partition and the process within it is called Internal Fragmentation. One solution to these problems is compaction.

4.What do you mean by Compaction?

One solution to internal fragmentation is compaction. It is possible to combine all the holes (free spaces) into a large block by pushing all the processes downward as far as possible. Compaction is possible only if relocation is dynamic, and is done at execution time.



Compaction is usually not done because it consumes a lot of CPU time. It is usually done on a large machine like mainframe or supercomputer because they are supported with a special hardware to perform this task (compaction).

5. How the problem of external fragmentation can be solved?

Solution to external fragmentation :

1) Compaction : shuffling the fragmented memory into one contiguous location.
2) Virtual memory addressing by using paging and segmentation.

6. What are Pages and Frames?

Paging is a memory management scheme that permits the physical -address space of a process to be non-contiguous. In the case of paging, physical memory is broken

into fixed-sized blocks called frames and logical memory is broken into blocks of the same size called pages.
A non-contiguous policy with a fixed size partition is called paging. Paging is a process whereby Physical memory is broken down into fixed size blocks called page frame. Logical memory is broken down into blocks of the same size as physical memory blocks and is called pages. When a process is to be executed, its pages are loaded from the backing store into any available memory frames.

7. What is the use of Valid-Invalid Bits in Paging?

When the bit is set to valid, this value indicates that the
associated page is in the process's logical address space, and is thus a legal page. If the bit is said to invalid, this value indicates that the page is not in the process's logical address space. Using the valid-invalid bit traps illegal addresses.

8.What is Segmentation? (2022-2 marks)

In the paging scheme, pages are of a fixed size an alternate approach, called segmentation, divides the process's address space into a number of segments - each of variable size. A program is a collection of segments. A segment is a logical unit such as: main program, procedure, function, method, object, local variables, global variables, common block, stack, symbol table, arrays etc.

9.Define Belady's Anomaly.

More number of page faults occur when more frames are allocated to a process. This most unexpected result is termed as Belady's Anomaly. Belady's anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.

10.What is Virtual Memory? List advantages.

Virtual memory is a memory management technique that allows the execution of processes that may not be completely in main memory and do not require contiguous memory allocation. The address space of virtual memory can be larger than that physical memory.
Advantages:
• Programs are no longer constrained by the amount of physical memory that is available.
• Increased degree of multiprogramming.
• Less overhead due to swapping.
Virtual memory can be implemented via:
- Demand paging
- Demand segmentation

11.What is Demand Paging? (2022-2 marks)

Demand paging is similar to paging with swapping. In demand paging a page of the process is loaded in main memory only when it is demanded. In pure demand paging initially the main memory is considered as a set of free frames. We start the execution of the process with no pages of the process in main memory. Each page

of the process is loaded only when it is demanded i.e., never bring a page until it is required.
Virtual memory is commonly implemented by demand paging. In demand paging, the pager brings only those necessary pages into memory instead of swapping in a whole process. Thus it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.
The two major problems to implement demand paging is:
Frame allocation algorithm
Page replacement algorithm

12.What is Thrashing? (2022-2 marks)

Too much or over allocation of memory can lead to a serious performance problem known as thrashing. Thus, when a page fault occurs, the page that is removed from memory will soon give rise to a new fault, which in turn removes a page that will soon give rise to a new fault . In a system that is thrashing, a high percentage of the system's resources is devoted to paging, and overall CPU utilization and throughput drop dramatically.
In a virtual memory system, thrashing is a situation when there is excessive swapping of pages between memory and the hard disk, causing the application to respond more slowly. The operating system often warns users of low virtual memory when thrashing is occurring.

13.What is Context Switching? (2022-1 mark)

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as context switch. It is the process of saving the state of a running process and loading the state of another process so that multiple processes can share a single CPU (Central Processing Unit) or core, giving the appearance of concurrent execution. Context switching is crucial for multitasking and multiprogramming in operating systems, enabling them to efficiently manage multiple tasks or processes.

14.What is spooling?

A spool is a buffer that holds output for a device, such as printer, that cannot accept interleaved data streams. When an application finishes printing, the spooling system queues the corresponding spool file for output to the printer. The spooling system copies the queued spool files to the printer one at a time.

15.What is Address Translation or Address Binding schemes?

The binding of instructions and data to memory can be done in any of the following ways.
1.At compile time The Compiler generates physical addresses at compile time. It requires knowledge of where the process resides. If location changes, then it will be necessary to recompile the code. This scheme is rarely used in MSDOS .COM files.
2.At link-edit time The compiler generates relocatable addresses for each process unit. Linkage editor converts the relocatable address to absolute address. A program can be loaded only where specified and cannot move once loaded.
3.At load time Similar to at link-edit time, but do not fix the starting address. Program can be loaded anywhere, but cannot be split.

4**.At execution time** Address translation is done dynamically during execution. Hardware is needed to perform the virtual to physical address translation quickly.

16. Define Security.

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. • Authentication • One Time passwords • Program Threats • System Threats • Computer Security Classifications.

17. One Time passwords(2022-1 mark)

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways such as:

• **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
• **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
• **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

18. Define Page Faults .

The logical address is in the range of valid addresses, but the corresponding page is not currently present in memory, but rather is stored on disk. The operating system must bring it into memory before the process can continue to execute this condition a page fault.

19.What is a Reference String?

An algorithm is evaluated by running it on a particular string of
memory references and computing the number of page faults. The string of memory reference is called a reference string.

20.What is locality of reference?

Locality of reference refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period. In other words, **Locality of Reference** refers to the tendency of the computer program to access instructions whose addresses are near one another.

21.What is the basic approach of Page Replacement?

If no frame is free is available, find one that is not currently
being used and free it. A frame can be freed by writing its contents to swap space, and changing the page table to indicate that the page is no longer in memory. Now the freed frame can be used to hold the page for which the process faulted.

22.Define Overlays.

To enable a process to be larger than the amount of memory allocated to it, overlays are used. The idea of overlays is to keep in memory only those instructions and data that are needed at a given time. When other instructions are needed, they are loaded into space occupied previously by instructions that are no longer needed.

23. What do you mean by Best Fit, First fit and Worst fit?

The processes are allocated to the partitions based on the allocation policy of the system. The allocation policies are: • First Fit • Best Fit • Worst Fit In first fit policy, the memory manager will choose the first available partition that can accommodate the process even though its size is more than that of the process. In worst fit policy, the memory manager will choose the largest available partition that can accommodate the process. In best-fit policy, the memory manager will choose the partition that is just big enough to accommodate the process.

4 Marks

1.Explain dynamic loading.

To obtain better memory-space utilization dynamic loading is used. With dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a relocatable load format. The main program is loaded into memory and executed. If the routine needs another routine, the calling routine checks whether the routine has been loaded. If not, the relocatable linking loader is called to load the desired program into memory.

2. Explain dynamic Linking.

Dynamic linking is similar to dynamic loading, rather that loading being postponed until execution time, linking is postponed. This feature is usually used with system libraries, such as language subroutine libraries. A stub is included in the image for each library-routine reference. The stub is a small piece of code that indicates how to locate the appropriate memory-resident library routine, or how to load the library if the routine is not already present.

3.How does Paging work? (2022- 4 marks)
Explain paging scheme for memory management, discuss the paging hardware and Paging.

Memory allocation methods in an operating system (OS) are techniques used to manage and allocate system memory to different processes and tasks. These methods ensure that processes run efficiently, without interfering with each other or causing memory-related issues. There are several memory allocation methods, and the choice of method depends on the specific needs and constraints of the system. Here are some common memory allocation methods:

1. **Contiguous Memory Allocation:**
    - In this method, the entire process is allocated a single contiguous block of memory.
    - It's simple and efficient but can lead to **memory fragmentation.**
2. **Partitioned Memory Allocation:**
    - Memory is divided into fixed or variable-sized partitions, and each partition can hold a single process.
    - Fixed partitions are simple but can lead to internal fragmentation, while variable partitions are more flexible but can still suffer from fragmentation issues.
3. **Paging:**
    - The physical memory is divided into fixed-sized blocks called frames, and processes are divided into fixed-sized blocks called pages.
    - Processes don't need to be stored in contiguous memory, which reduces external fragmentation.
    - The OS maintains a page table to map logical pages to physical frames.
4. **Segmentation:**
    - Memory is divided into logical segments, such as code, data, and stack.
    - Segments can grow or shrink dynamically, making this method more flexible than paging.
5. **Demand Paging:**
    - Processes don't load into memory entirely when they start but are loaded on demand, as needed.
    - This minimizes memory wastage and allows for more concurrent processes.
6. **Virtual Memory:**
    - Combines paging and demand paging to allow processes to use more memory than is physically available.
    - Part of the process remains in physical memory, while the rest can be swapped in and out from secondary storage (e.g., disk) as needed.
7. **Best-Fit, Worst-Fit, First-Fit:**
    - These are allocation algorithms used to allocate available memory blocks to processes based on size and other criteria.
    - Best-fit looks for the smallest block that fits the process, worst-fit looks for the largest block, and first-fit takes the first available block that fits the process.

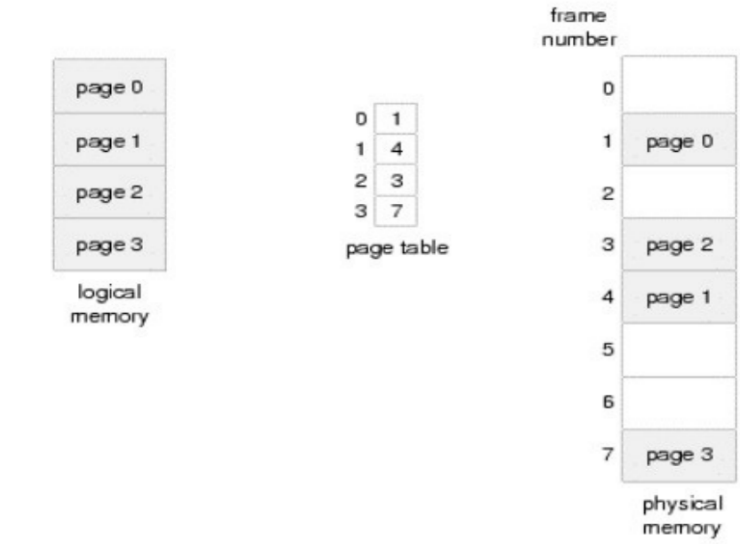
8. **NON-CONTIGUOUS ALLOCATION** Non-contiguous memory means that the available memory is not contiguous but is distributed. This scheme has the benefit of minimizing external fragmentation. The logical address space of the process is allowed to be non- contiguous, thus allowing a process to be allocated physical memory wherever the later is available Non-contiguous memory allocation is of different types,

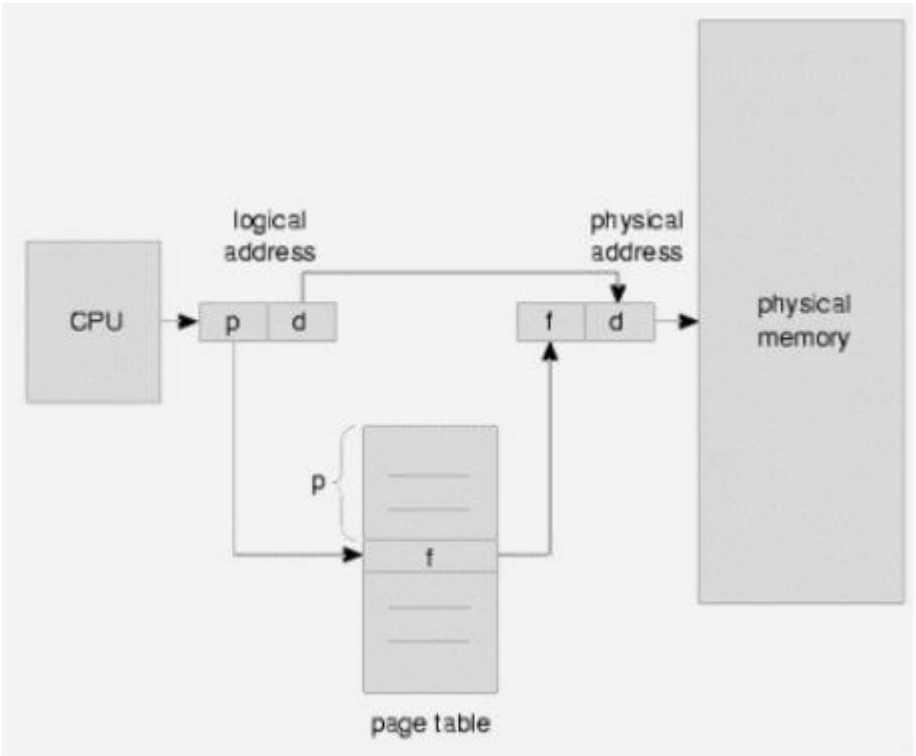
1. Paging
2. Segmentation
3. Segmentation with paging

Each of these memory allocation methods has its advantages and disadvantages, and the choice of method depends on the design goals and constraints of the operating system. Modern operating systems often use a combination of these methods to achieve optimal memory management and performance.

Paging:-

A non-contiguous policy with a fixed size partition is called paging. Paging is a process whereby Physical memory is broken down into fixed size blocks called page frame. Logical memory is broken down into blocks of the same size as physical memory blocks and is called pages. When a process is to be executed, its pages are loaded from the backing store into any available memory frames.





Address translation Logical address=Page number + Page Offset
Physical address=Frame Number + Page offset

4.Explain about contiguous memory allocation?

Contiguous Allocation Contiguous literally means adjacent. Here it means that the program is loaded into a series of adjacent (contiguous) memory locations. In contiguous memory allocation, the memory is usually divided into two partitions, one for the OS and the other for the user process. At any time, only one user process is in memory and it is run to completion and then the next process is brought into the memory. This scheme is sometimes referred to as the Single Contiguous Memory Management.
In contiguous memory allocation, each process is contained in a single section of memory that is contiguous to the section containing the next process

5.Explain about advantages and disadvantages of paging? And Explain difference between paging and segmentation?

Advantages and disadvantages of paging

- It reduces external fragmentation but still it suffer from internal fragmentation.
- It is simple to implement but assumed as an efficient memory management technique.
- Due to equal size of pages and frames swapping, becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

S.NO	Paging	Segmentation
1.	In paging, the program is divided into fixed or mounted size pages.	In segmentation, the program is divided into variable size sections.
2.	For the paging operating system is accountable.	For segmentation compiler is accountable.
3.	Page size is determined by hardware.	Here, the section size is given by the user.

4.	It is faster in comparison to segmentation.	Segmentation is slow.
5.	Paging could result in internal fragmentation.	Segmentation could result in external fragmentation.
6.	In paging, the logical address is split into a page number and page offset.	Here, the logical address is split into section number and section offset.
7.	Paging comprises a page table that encloses the base address of every page.	While segmentation also comprises the segment table which encloses segment number and segment offset.
8.	The page table is employed to keep up the page data.	Section Table maintains the section data.
9.	In paging, the operating system must maintain a free frame list.	In segmentation, the operating system maintains a list of holes in the main memory.
10.	Paging is invisible to the user.	Segmentation is visible to the user.
11.	In paging, the processor needs the page number, and offset to calculate the absolute address.	In segmentation, the processor uses segment number, and offset to calculate the full address.
12.	It is hard to allow sharing of procedures between processes.	Facilitates sharing of procedures between the processes.

13	In paging, a programmer cannot efficiently handle data structure.	It can efficiently handle data structures.
14.	This protection is hard to apply.	Easy to apply for protection in segmentation.
15.	The size of the page needs always be equal to the size of frames.	There is no constraint on the size of segments.
16.	A page is referred to as a physical unit of information.	A segment is referred to as a logical unit of information.
17.	Paging results in a less efficient system.	Segmentation results in a more efficient system.

6.Explain about memory management?

Memory Management Unit(MMU) An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit—that is, the one loaded into the memoryaddress register of the memory—is commonly referred to as a physical address

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Instructions and data to memory addresses can be done in following ways

- **Compile time** -- When it is known at compile time where the process will reside, compile time binding is used to generate the absolute code.
- **Load time** -- When it is not known at compile time where the process will reside in memory, then the compiler generates re-locatablecode.
- **Execution time** -- If the process can be moved during its execution from one memory segment to another, then binding must be delayed to be done at run time.

Dynamic Loading

dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Dynamic Linking

Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed. Operating system can link system level libraries to a program. When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamiclinking.

In static linking, libraries linked at compile time, so program code size becomes bigger whereas in dynamic linking libraries linked at execution time so program code size remains smaller.

Logical versus Physical Address Space (2022-1 mark)

Logical Address	**Physical Address**
1. An address generated by CPU is referred to us a logical address.	1. An address seen by memory unit that is, the one loaded into the memory address register of the memory is referred to as physical address.
2. The set of all logical address generated by a program is a logical address space.	2. The set of all physical address corresponding to these logical addresses is a physical address.
3. For user view.	3. For system view.
4. The user program deals with logical address or these are generated by user (program).	4. These are generated by memory management unit (MMU).

An address generated by the CPU is a logical address whereas address actually available on memory unit is a physical address. Logical address is also known **a Virtual address.**

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

15 Marks

1.Explain with the help of examples FIFO and LRU, optical page replacement algorithms with example reference string. Mention the merits and demerits of each of the above.

Explain about the following page replacement algorithms a)FIFO b)OPR, c)LRU LRU (Least Recently Used)

Explain Optimal Page Replacement algorithm with example. OPT or OPR (Optimal) (2022-4 marks)

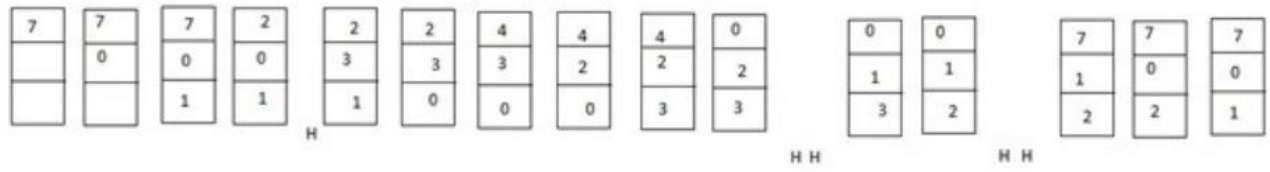
What is the various Page Replacement Algorithms used for Page Replacement?

If there are no free memory frame to accommodate a newly demanded page, then one of the existing pages in memory needs to be swapped out and the new page loaded. To decide "Which page should I swap out?, we have page replacement policies or algorithms.

1. FIFO PAGE REPLACEMENT ALGORITHM: Replace the page which is in memory for the longest time. When a page must be replaced, the oldest one is identified and removed from the main memory. In order to implement the FIFO replacement algorithm, the memory manager must keep track of the relative order of the loading of pages into the main memory. To accomplish this is to maintain a FIFO queue.

To illustrate the problems that are possible with a FIFO replacement algorithm, we consider the reference string.

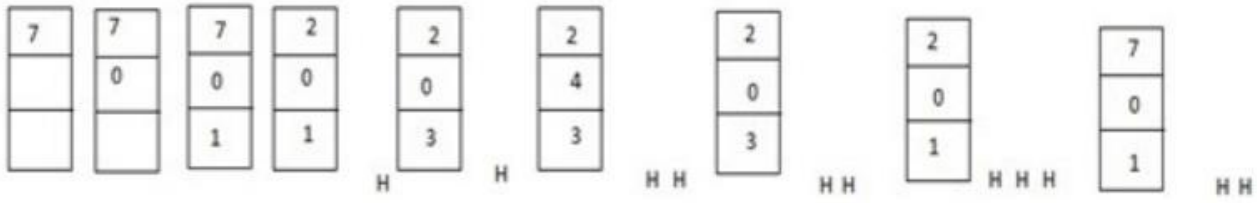
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



For our example reference string, our three frames are initially empty. The first three references (7,0,1) cause page faults and are brought into these empty frames. The next reference (2) replaces page 7, because page 7 was brought in first. Since 0 is the next reference and 0 is already in memory, we have no fault for this reference. The first reference to 3 results in replacement of page 0, since it is now first in line. Because of this replacement, the next reference, to 0, will fault. Page 1 is then replaced by page 0. The first three references (7, 0, 1) cause page faults and get loaded into three empty frames. The next reference (2)causes a page fault and page replacement. The page replaced is page 7 ,because it was brought in the first. H represents a 'HIT' means the referenced page is already in the memory (no page fault). . The FIFO page-replacement algorithm is easy to understand and program. However, its performance is not always good. Generally, on increasing the number of frames to a process' virtual memory, its execution becomes faster as less number of page faults occur. Sometimes the reverse happens, i.e. more number of page faults occur when more frames are allocated to a process. This most unexpected result is termed as **Belady's Anomaly.**

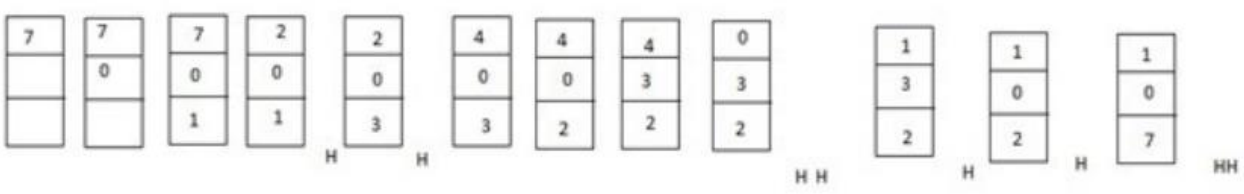
Belady's anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.
Performance Hit ratio = 5/20 = 25%
Advantages: Easy to understand/ implement.
Disadvantages: It has a poor performance.

2. OPTIMAL PAGE REPLACEMENT ALGORITHM: One result of the discovery of Belady's anomaly was the search for an optimal pagereplacement algorithm—the algorithm that has the lowest page-fault rate of all algorithms and will never suffer from Belady's anomaly. Such an algorithm does exist and has been called OPT or MIN. Replace the page that will not be used for the longest period of time. Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames. Consider the reference string.
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



The first three references (7, 0, 1) cause page and get loaded into three empty frames. The reference to page 2 replaces page 7, because 7 will not be used until reference 18, whereas page 0 will be used at reference 5, and page 1 at reference 14.The reference to page 3 replaces page 1 as it would be the last of the three pages in memory to be referenced again.
Performance: Hit ratio = 11/20 = 55%
Advantages: Has the lowest page fault rate for a fixed number of frames.
Disadvantages: Difficult to implement, because it requires future knowledge of reference string.

3. LRU PAGE REPLACEMENT ALGORITHM : Replace the page that has not been used for the longest period of time. When a page has to be replaced, LRU chooses that page that has not been used for the longest period of time.
Consider the reference string.
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



LRU algorithm sees that, of the three frames in memory, page 2 was used least recently. The most recently used page was page 0 and just before that, page 3 was used. Thus LRU replaces page 2 not knowing that page 2 is about to be used again. When page 2 is again referenced, LRU replaces page 3 since it is the least recently used.

Performance Hit ratio = 8/20 = 40%

Advantages: Reasonable approximation of Optimal Algorithm.

Disadvantages: Requires substantial hardware assistance.

MODULE -IV

STORAGE MANAGEMENT

2 Marks

1. What is a File?

A file is a named collection of related information that is recorded on secondary storage. A file contains either programs or data. A file has certain "structure" based on its type.

a "file" refers to a collection of related data or information that is stored on a computer's storage device, such as a hard drive, solid-state drive, or networked file system. Files can contain a wide range of data, including text, images, videos, programs, configuration settings, and more.

File Systems: Files are organized and managed by the file system of the operating system. The file system determines how files are stored, named, accessed, and organized on the storage medium. Common file systems include NTFS and FAT for Windows, ext4 for Linux, and HFS+ for macOS.

2.What are the File attributes? List the various File Attributes. (2022-2 marks)

File Attributes: Each file has associated attributes that provide information about the file, such as its name, size, location on the storage device, and timestamps (e.g., creation time, last modification time). File attributes can also include permissions and ownership information.

A file has certain other attributes, which vary from one operating system to another, but typically consist of these: Name, identifier, type, location, size, protection, time, and date and user identification.

File Attributes

•Name

The symbolic file name is the only information kept in human- readable form.

•Identifier

This unique tag, usually a number, identifies the file within the file system; it is the non-humanreadable
name for the file.

•Type

This information is needed for systems that support different types of files.

•Location

This information is a pointer to a device and to the location of the file on that device.

•Size

The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.

•Protection

Access-control information determines who can do reading, writing, executing, and so on.

•Time, date, and user identification

This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

3.What are the various File Operations? (2022-4 marks)

The six basic file operations are

- Creating a file
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file
- Truncating a file

Files are meant to store information and allow it to be retrieved later. Different systems provide different operations to allow storage and retrieval. The most common File Operations are follows.

1. Create: The file is created with no data. The purpose of the call is to announce that the file is coming and to set

2. Delete:When the file is no longer needed, it has to be deleted to free up disk space.

3. Open: Before using a file, a process must open, the purpose of the open call is to allow the system to fetch the attributes and list of disk addresses into main memory for rapid access on later calls.

4. Close: When all the accesses are finished, the attributes and disk addresses are no longer needed, so the file should be closed to free up internal table space.

5. Read: To read data from file. Usually, the bytes a come from the current position. The caller must specify how much data are needed and must also provide a buffer to put them in.

6. Write: Data are written to the file, again, usually at the current position. If the current position is end of the file then the file size get increased.

7. Append: This call is a restricted from of write. It can only add data to the end of the file.

8. Seek: For random access files, a method is needed to specify from where to take the data.

9. Get attributes: Processes often need to read file attributes to do their work.

10 .Set attributes: Some of the attributes are user settable and can be changed after the file has been created. This system call makes that possible. The protection mode information is an obvious example.

11.Rename: It frequently happens that a user needs to change the name of an existing file.

12.Truncate: To erase the content of the file but keep its attributes.

4.What is the information associated with an Open File?

Several pieces of information are associated with an open file which may be:

- File pointer

✱ File open count

✱ Disk location of the file
✱ Access rights

5.What are the different Accessing Methods of a File?

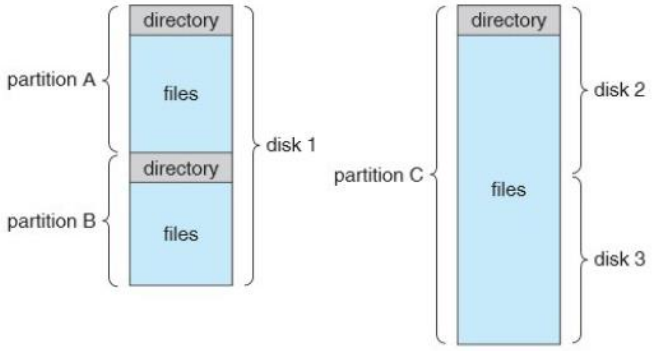
The different types of accessing a file are:

Sequential access: Information in the file is accessed sequentially.

Direct access: Information in the file can be accessed without any particular order.
Other access methods: Creating index for the file, indexed sequential access method (ISAM) etc.

6.What is Directory?

A file is any kind of computer document whereas a directory is a collection of files and folders.
The device directory or simply known as directory records information-such as name, location, size, and type for all files on that particular partition. The directory can be viewed as a symbol table that translates file names into their directory entries.



7.What is RAID technique? (2022-1 mark)

RAID, which stands for Redundant Array of Independent Disks, is a technology used in data storage to improve data redundancy, performance, and reliability. It involves using multiple hard drives or solid-state drives (SSDs) in various configurations to achieve these goals.
Examples:

RAID 0 (Striping):
Data is striped across two or more drives.
No redundancy; if one drive fails, all data is lost.

RAID 1 (Mirroring):

- Data is duplicated on two or more drives.
- Provides redundancy; if one drive fails, data is still accessible from the mirrored drive(s).

(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

8. What are the operations that can be performed on a Directory?

Operations that can be performed on a directory*:*

•Search for a file

To search a directory structure to find the entry for a particular file. Since files have symbolic names, and similar names may indicate a relationship among files, it is able to find all files whose names match a particular pattern.

•Create a file

New files need to be created and added to the directory.

•Delete a file

When a file is no longer needed, we want to be able to remove it from the directory.

•List a directory

We need to be able to list the files in a directory and the contents of the directory entry for each file in the list.

•Rename a file

Because the name of a file represents its contents to its users, we must be able to change the name when the contents or use of the file changes. Renaming a file may also allow its position within the directory structure to be changed.

• Traverse the file system

To access every directory and every file within a directory structure. For reliability, it is a good idea to save the contents and structure of the entire file system at regular intervals. This technique provides a backup copy in case of system failure.

What are the most common schemes for defining the Logical Structure of a Directory?

9. Define UFD and MFD.

1. MFD (Master File Directory):

The MFD, or Master File Directory, is the top-level directory that contains information about all the users or entities on a computer system. It serves as a master directory for the entire system and keeps track of all the user directories. Each user or entity typically has their own entry or record within the MFD, which points to their UFD (User File Directory).

2. UFD (User File Directory):

The UFD, or User File Directory, is a directory specific to a particular user or entity on the system. Each user's UFD contains information about the files and directories owned or accessible by that user. It's a subdirectory under the MFD and provides a user-specific file organization structure.

10. What is Disk Scheduling? (2022-1 mark)

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

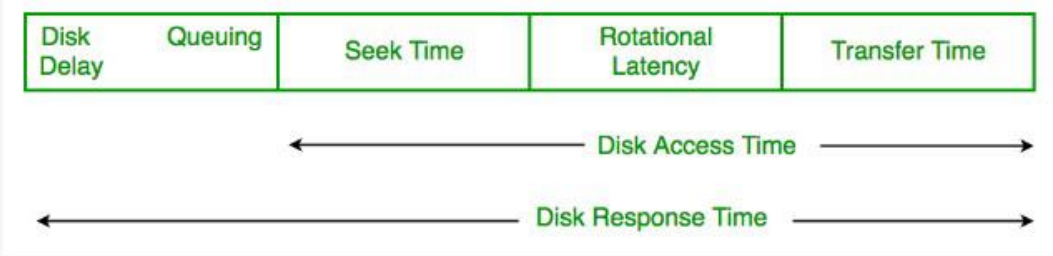
 Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.

 Two or more request may be far from each other so can result in greater disk arm movement.
 Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

Some of the important terms:

 Transfer Time: Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

 Disk Access Time: Disk Access Time is:



Disk Response Time: Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests.

11.What is a Path Name?

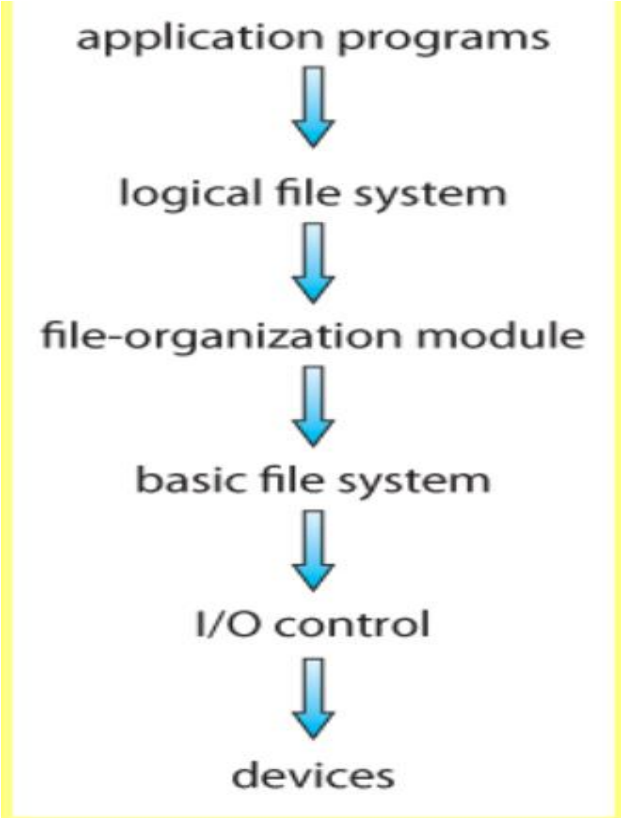
A pathname is the path from the root through all subdirectories
to a specified file. In a two-level directory structure a user name and a file name define a path name.
Path names can be of two types.
Absolute path name: Begins at the root and follows a path down to the specified file, giving the directory names on the path.
Relative path name: Defines a path from the current directory.

12.File system Structure.

File systems define the structure and rules for how data is stored and accessed on storage devices, such as hard drives, solid-state drives, and network-attached storage (NAS).

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

A typical file-control block.



13.Distinguish file from dictionary.

A file is any kind of computer document whereas a directory is a collection of files and folders.

14. Define Seek Time and Latency Time.

The time taken by the head to move to the appropriate cylinder or track is called seek time. Once the head is at right track, it must wait until the desired block rotates under the read-write head. This delay is latency time.

 Seek Time : Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
 Rotational Latency: Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

15.What are the various Disk-Scheduling Algorithms?

First Come First Served Scheduling , Shortest Seek Time First Scheduling

SCAN Scheduling, C-SCAN Scheduling, LOOK scheduling

16. Different file types.

File types, also known as file formats, refer to the specific structure and encoding used to store and organize data within a computer file. Different file types are designed for various purposes and applications, and they dictate how the data within the file should be interpreted. Here are some common file types:

1. **Text Files**:
   - **Plain Text (.txt)**: Contains unformatted text, often in ASCII or Unicode encoding.
   - **Rich Text Format (.rtf)**: Supports text formatting, such as fonts, colors, and styles.
2. **Document Files**:
   - **Microsoft Word (.doc, .docx)**: Contains text, images, and formatting for word processing.
   - **PDF (.pdf)**: Portable Document Format, used for fixed-layout documents.
3. **Spreadsheet Files**:
   - **Microsoft Excel (.xls, .xlsx)**: Stores tabular data and formulas for calculations.
   - **CSV (.csv)**: Comma-Separated Values file, used for data exchange in a plain text format.
4. **Image Files**:
   - **JPEG (.jpg, .jpeg)**: Lossy compression format for digital images.
   - **PNG (.png)**: Lossless compression format with support for transparency.
   - **GIF (.gif)**: Supports animations and simple graphics.
   - **BMP (.bmp)**: Windows Bitmap format, uncompressed and of high quality.
5. **Audio Files**:
   - **MP3 (.mp3)**: Compressed audio format for music and audio files.
   - **WAV (.wav)**: Uncompressed audio format with high quality.
   - **MIDI (.midi, .mid)**: Musical Instrument Digital Interface for storing musical compositions.
6. **Video Files**:
   - **MP4 (.mp4)**: A widely used video format with compression.
   - **AVI (.avi)**: Audio Video Interleave format, developed by Microsoft.
   - **MKV (.mkv)**: Matroska multimedia container format, known for flexibility.
7. **Archive Files**:
   - **ZIP (.zip)**: Compressed archive format for multiple files.
   - **RAR (.rar)**: Proprietary archive format with higher compression ratios.
   - **TAR (.tar)**: Tape Archive format, often used in Unix-like systems.
8. **Executable Files**:
   - **EXE (.exe)**: Windows executable files.
9. **Database Files**:
   - **SQL Database (.sql)**: Contains structured data and SQL commands.
   - **Microsoft Access (.mdb, .accdb)**: Proprietary database format.
10. **Web Files**:
    - **HTML (.html)**: Hypertext Markup Language for web pages.
    - **CSS (.css)**: Cascading Style Sheets for web page styling.
    - **JavaScript (.js)**: Scripting language for web development.

11. **System Files**:
    - **INI (.ini)**: Configuration files often used in Windows.
    - **DLL (.dll)**: Dynamic Link Library files in Windows.
    - **SYS (.sys)**: System files in Windows and DOS.

17.What is access matrix?

- General model of protection can be viewedabstractly as a matrix, called an access matrix.
- The rows of the access matrix represent domains,and the columns represent objects.
- Each entry in the matrix consists of a set of accessrights.

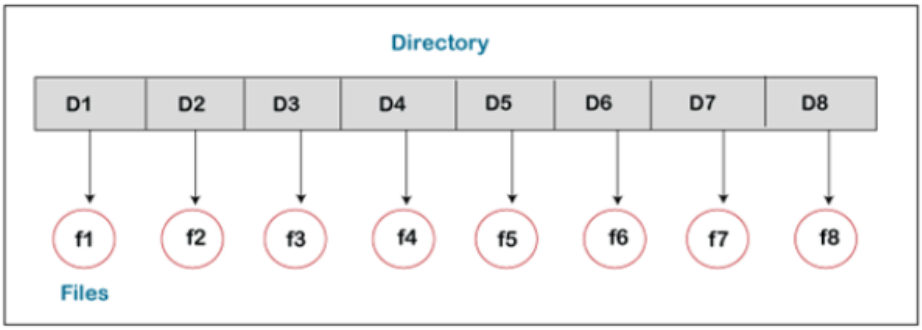
4 Marks

1. Explain about single-level, two-level directory structure?

Single – level directory

The simplest directory structure is the single-level directory. All files are contained in the same directory, which is easy to support and understand.

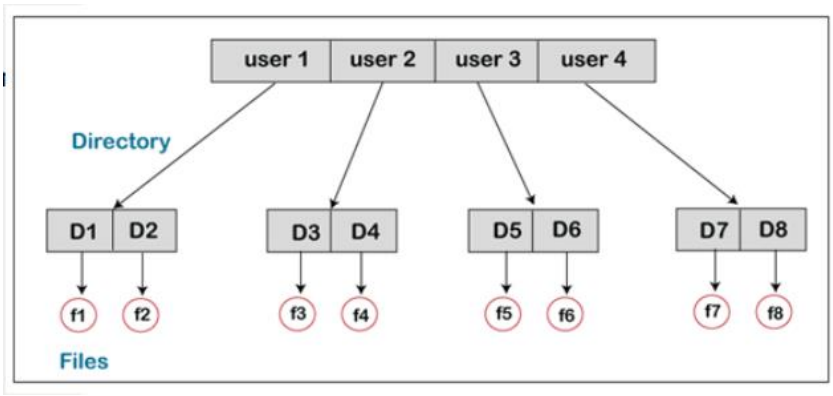
A single-level directory has limitations, when the number of files increases or when the system has more than one user. Since all files are in the same directory, they must have unique names.



Two-Level Directory

A single-level directory often leads to confusion of file names among different users to avoid this is to create a separate directory for each user. In the two-level directory structure, each user has his own user file directory (UFD. The UFDs have similar structures, but each lists only the files of a single user. When a user logs in, the system' s master file directory(MFD) is searched. The MFD is indexed by user name or account number, and each entry points to the UFD for that user.

When a user refers to a particular file, only his own UFD is searched. Thus, different users may have files with the same name, as long as all the file names within each UFD are unique.



The two-level directory structure solves the name-collision problem, it still has disadvantages. This structure effectively isolates one user from another. Isolation is an advantage but is a disadvantage when the users want to cooperate on some task or to access one another's files which is not allowed.

2.What are the several ways to access information from the file? (2022-4 marks)
Which are the file access methods?

Files store information. When it is used, this information must be accessed and read into computer memory. The information in the file can be accessed in several ways. Some systems provide only one access method for files. while others support many access methods, and choosing the right one for a particular application is a major design problem.

1.Sequential Access

The simplest access method is sequential access. Information in the file is processed in order, one record after the other.

For example, editors and compilers usually access files in this fashion.

Reads and writes make up the bulk of the operations on a file. A read operation —read next()— reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, the write operation —write next() — appends to the end of the file and advances to the end of the newly written material (the new end of file).

2.Direct Access

Another method is direct access(or relative access). Here, a file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order. The direct-access method is based on a disk model of a file, since disks allow random access to an file block. For direct access, the file is viewed as a numbered sequence of blocks or records.

Thus, we may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct-access file. Direct-access files are of great use for immediate access to large amounts of information.

Eg : Databases are often of this type. When a query concerning a particular subject arrives, we compute which block contains the answer and then read that block directly to provide the desired information.

3.Other Access Methods

These methods generally involve the construction of an index for the file. The Index contains pointers to the various blocks. To find a record in the file, we first search the index and then use the pointer to access the file directly and to find the desired record.

3.What are goals or principles of system protection?

System protection in an operating system refers to the mechanisms implemented by the operating system to ensure the security and integrity of the system. System protection involves various techniques to prevent unauthorized access, misuse, or modification of the operating system and its resources.
There are several ways in which an operating system can provide system protection:

User authentication: The operating system requires users to authenticate themselves before accessing the system. Usernames and passwords are commonly used for this purpose.

Access control: The operating system uses access control lists (ACLs) to determine which users or processes have permission to access specific resources or perform specific actions.

Need for Protection:

- To prevent the access of unauthorized users
- To ensure that each active programs or processes in the system uses resources only as the stated policy

4.Explain language based protection.

Language-based protection, also known as language-based security, is a concept in computer science and information security that involves using programming languages, their features, and associated mechanisms to enhance the security of software systems. The primary goal of language-based protection is to mitigate vulnerabilities and reduce the risk of security breaches by leveraging the properties of programming languages to prevent or detect common security issues.

1. **Access Control:** Programming languages often define access control mechanisms for objects and data. By enforcing proper access control through language constructs, unauthorized access to sensitive resources can be prevented.
2. **Security Policies:** Language-based protection may involve defining and enforcing security policies at the language level. This can include specifying which operations are allowed or restricted for specific objects or data.
3. **Secure Coding Practices:** Language-based protection often goes hand in hand with secure coding practices, which include writing code that adheres to security best practices and follows the language's security features and guidelines.

5.What do you mean by Threats in system security? (2022-2 marks)

System Threats

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system resources/ user files are misused.
Following is the list of some wellknown system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.

• **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings

15 Marks

1. Explain different I/O Systems in detail.

I/O Systems in computing are responsible for managing input and output operations between a computer's central processing unit (CPU) and various external devices, such as disks, displays, keyboards, and network interfaces. These systems are divided into three main components: I/O hardware, Application I/O interface, and Kernel I/O subsystem.

I/O Hardware:

I/O hardware refers to the physical components and interfaces that enable the exchange of data between the computer and external devices. This includes devices such as hard drives, keyboards, monitors, USB ports, network cards, and more. I/O hardware provides the necessary interfaces and protocols for data transfer. It also includes controllers and adapters that manage the communication between the CPU and these devices. The I/O hardware is responsible for generating and detecting signals, managing data buffers, and handling interrupts.

Application I/O Interface:

The Application I/O Interface is the layer of the I/O system that interacts with the application software. It provides a higher-level abstraction of I/O operations, making it easier for application programs to perform I/O tasks without having to worry about the low-level details of hardware and device-specific intricacies. It includes system calls and libraries that allow applications to read from and write to devices. For example, functions like **open()**, **read()**, and **write()** in Unix.

Kernel I/O Subsystem:

The Kernel I/O Subsystem is the intermediary layer between the Application I/O interface and the I/O hardware. It is responsible for managing and coordinating I/O operations across different devices and application programs.

Key functions of the Kernel I/O Subsystem include:

- Handling device-specific issues and managing device drivers.
- Managing I/O queues to optimize I/O operations.
- Implementing policies for I/O scheduling, including prioritizing I/O requests.
- Caching frequently accessed data for performance improvements.
- Handling interrupts from devices and notifying application programs of I/O completion.

2.Explain different Disk scheduling algorithms SCAN,CSCAN.CLOOK(DEC 2015)

Several disk-scheduling algorithms can be used.

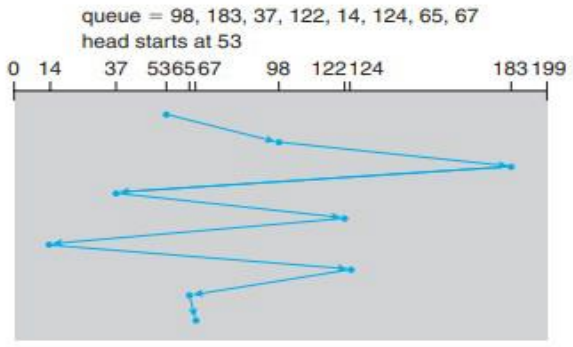
1.FCFS Scheduling

The simplest form of disk scheduling is, the first-come, first-served (FCFS) algorithm. This algorithm is fair, but it generally does not provide the fastest service.

Consider, for example, a disk queue with requests for I/O to blocks on cylinders

98, 183, 37, 122, 14, 124, 65, 67

If the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37, 122, 14,124, 65, and finally to 67, for a total head movement of 640 cylinders



FCFS disk scheduling.

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time
- May not provide the best possible service

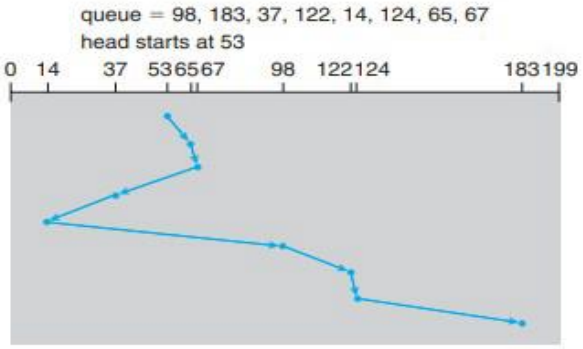
2. SSTF Scheduling

It service all the requests close to the current head position before moving the head far away to service other requests. The shortest-seek-time-first (SSTF) algorithm selects the request with the least seek time from the current head position.

Consider the request queue,

98, 183, 37, 122, 14, 124, 65, 67

the closest request to the initial head position (53) is at cylinder 65. Once we are at cylinder 65, the next closest request is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next. Continuing, we service the request at cylinder 14, then 98, 122,124, and finally 183 .This scheduling method results in a total head movement of only 236 cylinders .(65 67 37 14 98 122 124 183)



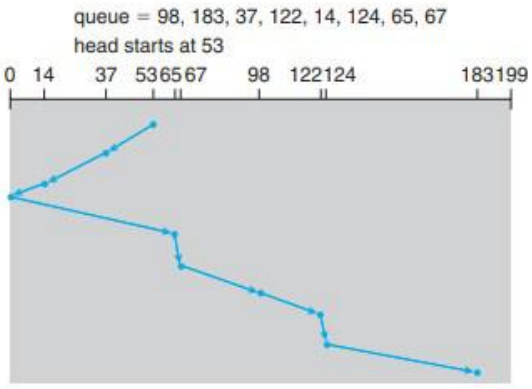
SSTF disk scheduling.

Advantages:
 Average Response Time decreases
 Throughput increases
Disadvantages:
 Overhead to calculate seek time in advance
 Can cause Starvation for a request if it has higher seek time as compared to incoming requests
 High variance of response time as SSTF favours only some requests

3.SCAN Scheduling

In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk. The SCAN algorithm is sometimes called the **elevator algorithm** , since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

Before applying SCAN to schedule the requests on cylinders 98, 183, 37, 122, 14, 124, 65, and 67. Assuming that the disk arm is moving toward 0 and that the initial head position is again 53, the head will next service 37 and then 14. At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 18.



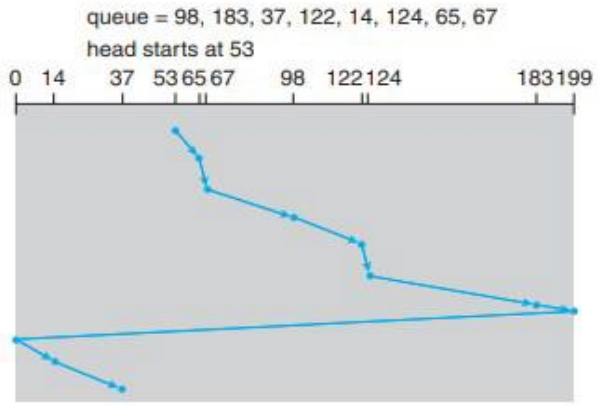
SCAN disk scheduling.

Advantages:
 High throughput
 Low variance of response time
 Average response time
Disadvantages:
 Long waiting time for requests for locations just visited by disk arm

4.C-SCAN Scheduling

Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN,C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip .The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.

Consider the request queue,
98, 183, 37, 122, 14, 124, 65, 67
Assume that the initial head position is at cylinder 53 ,the head moves as 65, 67, 98, 122, 124,183,14, 37.



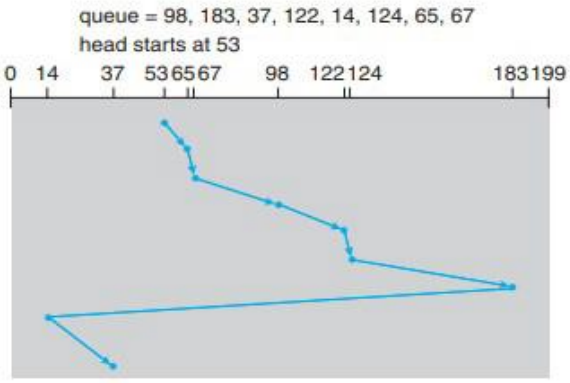
C-SCAN disk scheduling.

5. LOOK scheduling

It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

6.CLOOK scheduling

As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.



C-LOOK disk scheduling.

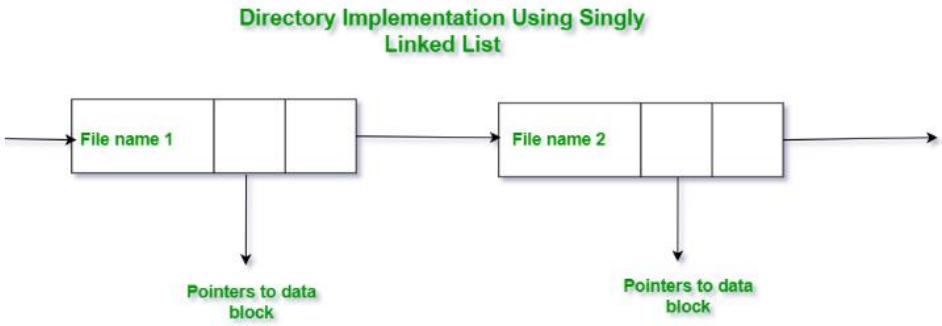
3.Explain Directory Implementation and Allocation methods. (2022-15 marks)

The selection of directory-allocation and directory-management algorithms significantly affects the efficiency, performance, and reliability of the file system.

Implementation methods:-

1. Linear List

The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks. This method is simple to program but time-consuming to execute. To create a new file, we must first search the directory to be sure that no existing file has the same name.



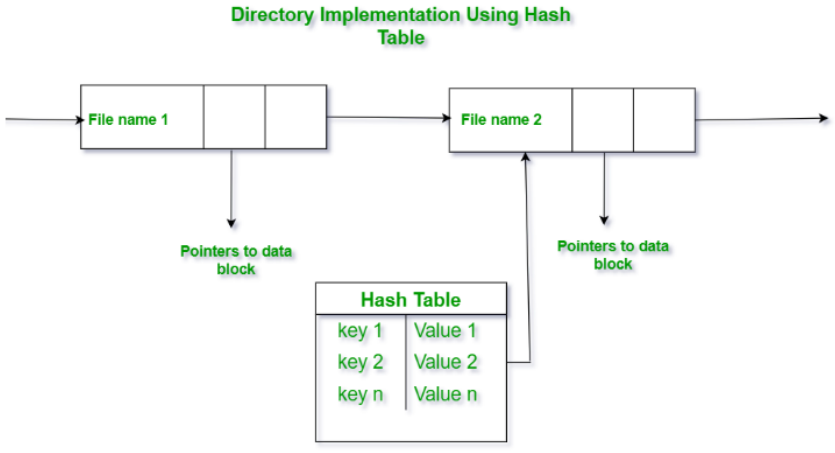
Then, we add a new entry at the end of the directory. To delete a file, we search the directory for the named file and then release the space allocated to it. To reuse the directory entry, we can do one of several things. We can mark the entry as unused, or we can attach it to a list of free directory entries or to copy the last entry in the directory into the freed location and to decrease the length of the directory

Disadvantage

For finding a file requires a linear search

2. Hash Table

The hash table takes a value computed from the file name and returns a pointer to the filename in the linear list. Therefore, it can greatly decrease the directory search time. Insertion and deletion are easy and provision must be made for name collisions .



Disadvantage

The major difficulties with a hash table are its generally fixed size and the dependence of the hash function on that size.

What are the Allocation Methods of a Disk Space?
What are the advantages of Contiguous Allocation?
What are the drawbacks of Contiguous Allocation of Disk Space?
What are the advantages and disadvantages of Linked Allocation?

Allocation methods:-

Three major methods of allocating disk space are in wide use: contiguous, linked, and indexed.

1.Contiguous Allocation

Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk. Disk addresses define a linear ordering on the disk. Accessing a file that has been allocated contiguously is easy.

Disadvantages

 Difficulty in finding space for a new file.

 Suffer from the problem of external fragmentation. As files are allocated and deleted, the free disk space is broken into little pieces.

 The total amount of space needed for a file is known in advance, preallocation may be inefficient.

2.Linked Allocation

Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. The size of a file need not be declared when the file is created. A file can continue to grow as long as free blocks are available.

Disadvantages

 It can be used effectively only for sequential-access files.

 A pointer requires 4 bytes out of a 512-byte block, then 0.78 percent of the disk is being used for pointers, rather than for information.

Another problem of linked allocation is reliability. If a pointer were lost or damaged ,entire file is can't be accessed.

3 .Indexed Allocation

Linked allocation solves the external-fragmentation and size-declaration problems of contiguous allocation. Linked allocation cannot support efficient direct access, since the pointers to the blocks are scattered with the blocks themselves all over the disk and must be retrieved in order.

Indexed allocation solves this problem by bringing all the pointers together into one location the **index block** .Each file has its own index block, which is an array of disk-block addresses .The ith entry in the index block points to the i th block of the file.

REFERENCES:

- 1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne-Operating System Concepts,
- 10th Edition.
- 2. P. Balakrishna Prasad- Operating Systems and Systems Programming, 5th Edition.
- 3. Achyut S Godbole and Atul Kahate - Operating systems, McGrawhill

4. George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair – Distributed Systems, Concepts and Designs, 5th Edition

OS Model question Pool

PART-A (1 mark)

1. What is degree of multiprogramming?
2. What do you mean by process?
3. What is FCB?
4. What is PCB?
5. What is hit ratio?
6. What is dirty bit?
7. What is mutual exclusion?
8. What is semaphore?
9. Which process scheduling is used in time sharing operating systems?
10. What is Turn around time?
11. What you mean by throughput?
12. What you mean by RAID?
13. What is an operating system?
14. What is spooling?
15. What do you mean by a process state?
16. What is starvation?
17. What is swapping?
18. List any two strategies used to select a free hole from the set of available holes?
19. What is access matrix?
20. Expand SJF?
21. What is critical sction?
22. What is demand paging?
23. Which algorithm is used for the process synchronization of N process?
24. What are cooperating processes?
25. Define CPU scheduler?

PART-B (2 marks)

1. Why we need process scheduling?
2. Explain various schedulers?
3. What is the use of fork() and exec() system call?
4. Explain IPC.
5. What is context switching?
6. Explain the tasks involved in the memory management function of an operating system.
7. What are the two important factors of operating system design?
8. Differentiate logical address vs physical address.
9. Differentiate between process and thread.
10. What is critical section? What is its important?
11. Define wait and signal operations of a semaphore.
12. State any one classical synchronization problem with solution.
13. What are the necessary conditions of a dead lock?
14. Differentiate external and internal fragmentation.
15. What is demand paging?
16. Differentiate execution time binding and load time binding.
17. What are overlays? What are its advantages?
18. Explain about real time operating systems.
19. Explain multiprocessor system.
20. Explain message passing model of communication.
21. What do you mean by authorization?
22. What are the various functions of an operating system?
23. Explain about multilevel feedback queue.
24. Explain various process states.
25. Write a note on free space management.
26. Explain thrashing.
27. Distinguish authorization and authentication.
28. Explain file attributes.
29. What are TLBs?

PART-C (4 marks)

1. Explain process states with diagram.
2. Explain the technique of multilevel queue scheduling.
3. Describe peterson solution for critical section.
4. Explain the terms: Turn around time, burst time, Response time, Waiting time.
5. What are the causes of thrashing?
6. Explain the file operations.
7. Explain the continuous file allocation technique.
8. Differentiate polling and interrupts.
9. What are the advantages of DMA transfer?
10. What all services supervised by I/O subsystem?
11. Explain SCAN algorithm used for disk scheduling.
12. Explain Optimal page replacement algoritm.

13. List the services of operating system.
14. List and discuss the various methods for implementing a directory.
15. Discuss I/O bound process and CPU bound process.
16. How is deadlock detection done?
17. Discuss the solution to dining philosopher's problem using semaphore.
18. Discuss about the address translation method used in paging with example.
19. Write various steps to handle a page fault by considering a page replacement policy.
20. Explain about various mechanisms available for implementing security in an operating system.
21. What is the difference between preemptive and non-preemptive scheduling?

PART-D (15 marks)

1. What are the different types of operating systems?
2. Explain various CPU scheduling algorithms in detail.
3. Explain with appropriate example Banker's algorithm.
4. Discuss various Non-continuous memory allocation techniques.
5. Explain directory structure in detail.
6. Explain page replacement algorithm with example.
7. Explain Disk scheduling alogorithms with example.
8. What is RAID? Explain different RAID levels.
9. What are the various free space management techniques available in an operating system.
10. Explain various protection and security measures needed in an operating system.
11. Explain in detail about deadlock detection and preventions.
12. Describe file structures, file attributes and file operations in detail.
13. Explain the architecture of Operating system.
14. Explain the concept of virtual memory.