D.N.R. COLLEGE (AUTONOMOUS): BHIMAVARAM

DEPARTMENT OF COMPUTER SCIENCE

OPERATING SYSTEM

IV SEMESTER PAPER-5

UNIT -1

INTRODUCTION TO OPERATING SYSTEM

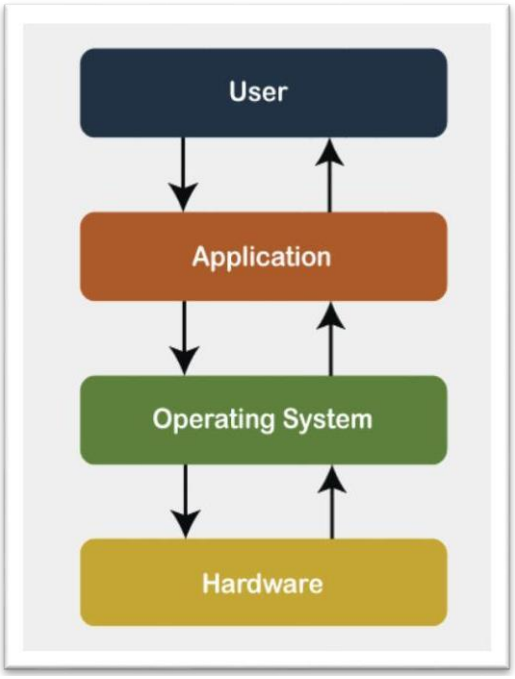
Operating System: An operating system is a program that acts as an interface between a user of a computer and the computer hardware.

History of the Operating System:

Today a very few know the inventor of Disk Operating System (DOS) **Gary Kildall.** DOS went on to transform into Operating systems that we all use today. Before his invention, every computer chip needed to have its own set of codes for users to interact with the computer. The first operating systems were developed in the 1950s,

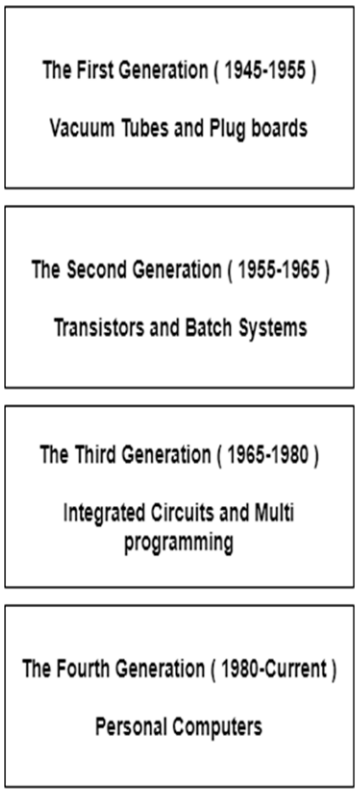
The first operating system was created by General Motors in 1956 to run a single IBM central computer. In the 1950s, IBM was the first computer manufacturer to take on the task of developing operating systems and began distributing operating systems included in its computers.

when computers could only run one program at a time. Later in the following decades, computers began to include more and more software programs, sometimes called libraries, that came together to create the start of today's operating systems.



Evolution of the Operating System (Generations of OS):

Operating Systems have evolved over the years. So, their evolution through the years can be mapped using generations of operating systems. There are four generations of operating systems. These can be described as follows −



OPERATING SYSTEM GENERATIONS

The First Generation (1945 - 1955):

1940 is the year when the first electronic computer was developed. This computer was created without any operating system. In that time, program was written for each task in absolute machine language. It was used for solving only simple mathematical calculations and this calculations didn't require an operating system

The Second Generation (1955 - 1965):

This operating system was developed for the IBM computer. GMOS was based on single stream batch processing system, because it collects all similar jobs in groups or batches and then submits the jobs to the operating system using a punch card to complete all jobs in a machine. In that time, machines were very big and not everyone could use them, but only professional operators.

The Third Generation (1965 - 1980):

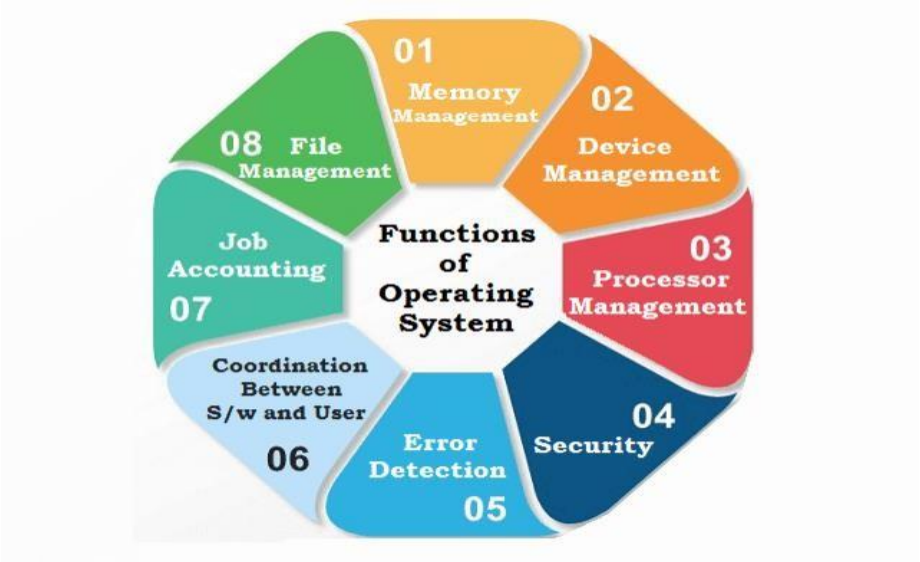
New operating system that could perform multiple tasks in a single computer program was success for this generation and this capability is called multiprogramming. Another progress which leads to developing of personal computers in fourth generation is a new development of minicomputers with DEC PDP-1.

The Fourth Generation (1980 - Present):

Development of personal computer represent the fourth generation. The cost of personal computer was high back then. Microsoft and the Windows operating system were related to creating personal computers. Some of the popular operating systems are Microsoft Windows, Mac OS, Linux. Linux operating system was created in early 1990s.

Basic OS Functions:

There are basic functions of operating system.



- **Process Management in Operating System:**

Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems. The OS must allocate resources that enable processes to share and exchange information.

- **Memory Management in Operating System:**

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.

- **File Management in Operating System:**

A file management system is used for file maintenance (or management) operations. It is is a type of software that manages data files in a computer system. A file management system has limited capabilities and is designed to manage individual or group files, such as special office documents and records.

- **Security Management in Operating System:**

The security management function of an operating system helps in implementing mechanisms that secure and protect the computer system internally as well as externally. Therefore an operating system is responsible for securing the system at two different levels which are internal security and external security.

- **Input/ Output Management in Operating System:**

Input/output management - it handles input to and output from attached hardware devices, such as the keyboard, mouse, monitor, hard disks, and printers. Ensuring that device drivers are present and up to date is another responsibility of the input/output management layer.

- **Device Management in Operating System:**

Device management in an operating system means controlling the Input/Output devices like disk, microphone, keyboard, printer, magnetic tape, USB ports, camcorder, scanner, other accessories, and supporting units like supporting units control channels. A process may require various resources, including main memory, file access, and access to disk drives, and others. If resources are available, they could be

allocated, and control returned to the CPU. Otherwise, the procedure would have to be postponed until adequate resources become available.

- **Error Detection in Operating System:**

OS needs to be constantly aware of possible errors May occur in the CPU and memory hardware, in I/O devices, in user program For each type of error, OS should take the appropriate action to ensure correct and consistent computing Debugging facilities can greatly enhance the user's and programmer 's abilities to efficiently use the system

- **Job Accounting in Operating System:**

Operating system performs the function of job accounting by keeping the track of time and resource used by several jobs and users.

Types of Operating System:

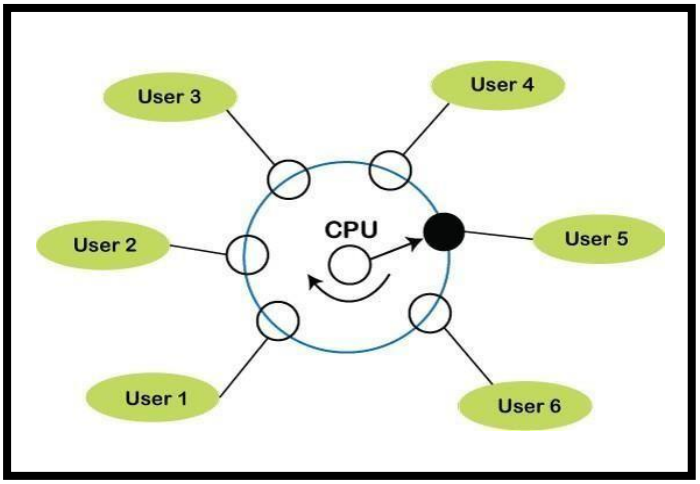
1. Batch Operating System
2. Time-Sharing Operating System
3. Embedded Operating System
4. Multiprogramming Operating System
5. Network Operating System
6. Distributed Operating System
7. Multiprocessing Operating System
8. Real-Time Operating System

1) Batch Operating System:

In Batch Operating System, there is no direct interaction between user and computer. Therefore the user needs to prepare jobs and save offline mode to punch card or paper tape or magnetic tape. After creating the jobs, hand it over to the computer operator; then the operator sort or creates the similar types of batches like B2, B3, and B4.

2) Time-Sharing Operating System:

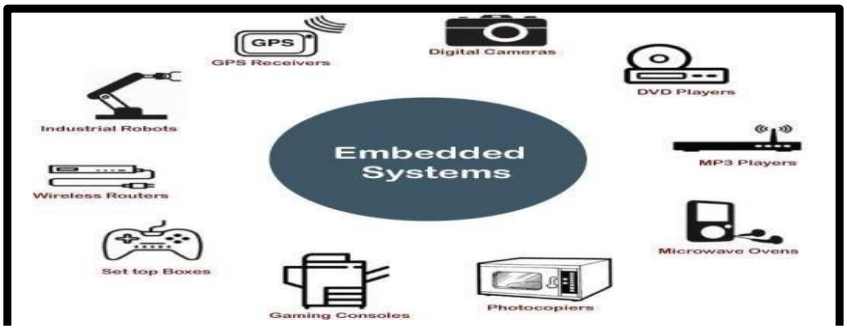
It is the type of operating system that allows us to connect many people located at different locations to share and use a specific system at a single time. The time-sharing operating system is the logical extension of the multiprogramming through which users can run multiple tasks concurrently. Furthermore, it provides each user his terminal for input or output that impacts the program or processor



currently running on the system.

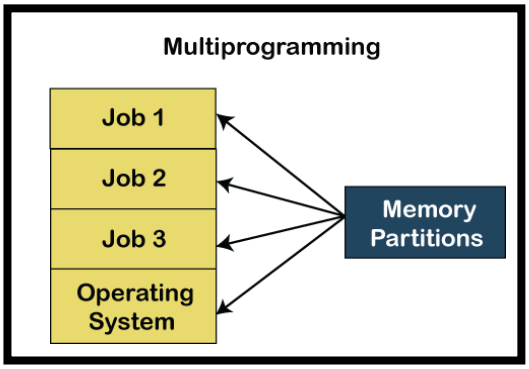
3) Embedded Operating System:

The Embedded operating system is the specific purpose operating system used in the computer system's embedded hardware configuration. These operating systems are designed to work on dedicated devices like automated teller machines (ATMs), airplane systems, digital home assistants, and the internet of things (IoT) devices.



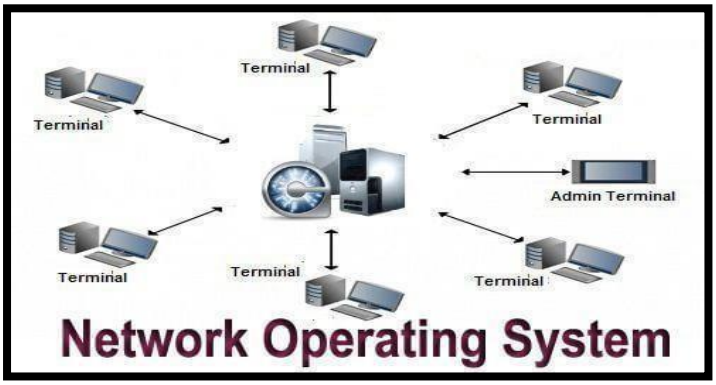
4) Multiprogramming Operating System:

A multi programming operating system refers to the concepts where in two or more processes or programs activate simultaneously to execute the processes one after another by the same computer system. When a program is in run mode and uses CPU, another program or file uses I/O resources at the same time or waiting for other system resources to become available. It improves the use of system resources, thereby increasing system throughput. Such a system is known as a multi programming operating system.



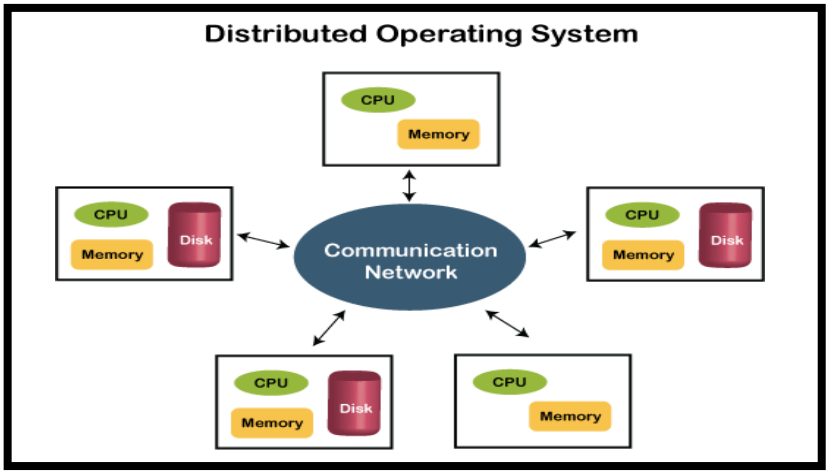
5) Network Operating System:

A network operating system is an important category of the operating system that operates on a server using network devices like a switch, router, or firewall to handle data, applications and other network resources. It provides connectivity among the autonomous operating system, called as a network operating system.



6) Distributed Operating system:

A distributed operating system provides an environment in which multiple independent CPU or processor communicates with each other through physically separate computational nodes. Each node contains specific software that communicates with the global aggregate operating system. With the ease of a distributed system, the programmer or developer can easily access any operating system and resource to execute the computational tasks and achieve a common goal.

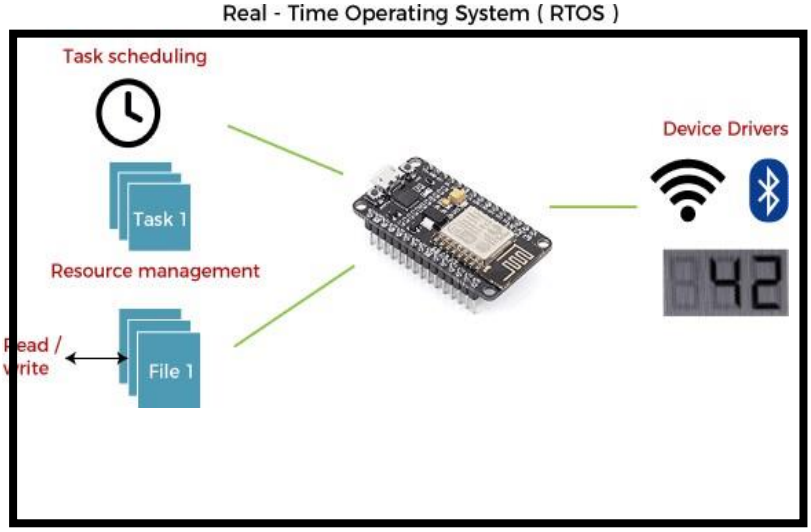


7) Multiprocessing Operating System:

It is the type of operating system that refers to using two or more central processing units (CPU) in a single computer system. However, these multiprocessor systems or parallel operating systems are used to increase the computer system's efficiency. With the use of a multiprocessor system, they share computer bus, clock, memory and input or output device for concurrent execution of process or program and resource management in the CPU.

8) Real-Time Operating System:

A real-time operating system is an important type of operating system used to provide services and data processing resources for applications in which the time interval required to process & respond to input/output should be so small without any delay real-time system. For example, real-life situations governing an automatic car, traffic signal, nuclear reactor or an aircraft require an immediate response to complete tasks within a specified time delay. Hence, a real-time operating system must be fast and responsive for an embedded system, weapon system, robots, scientific research & experiments and various real-time



Open-Source Operating Systems: An Open Source Operating System is an operating system whose copyright holders or owners enable the third parties or the user to use, see and edit the operating system's source code. We can create an operating system according to our requirements by altering the source code of an open-source operating system.

Types of Open Source Operating Systems

We can classify an open-source operating system as Linux-based and non- Linux-based open-source operating system.

- **Linux-based open source operating system** - The open source operating system which is based on the Linux kernel is known as Linux based open source operating system. (Kernel is responsible for the interface between the computer's hardware and its processes).

  Example: Ubuntu, Kali Linux, Linux mint, etc

- **Non-Linux based operating system** - An open source operating system that is not based on a Linux kernel operating system is known as a non-Linux based operating system.

  Example: FreeDOS, ReactOS, Haiku, etc.

Advantages of Open-Source Operating System

1. **Cost efficient:** Open source operating systems are usually free or sometimes less than the closed source operating system.
2. **Reliable and efficient:** As the source code of the operating system is open for all, Anyone around the globe can make changes to the source code and resolve any bugs or make some improvements to the source code. Open-source operating systems are more efficient and reliable because developers are also the users of the operating system and the bugs or issues are resolved.
3. **Flexibility:** The source code of the open-source operating system is available for everyone. The operating system can be modified or customized according to the requirement of the user.
4. **Developer Community** Open source operating systems usually have their own community, where developers around the globe gather in a place to work with the source code, improve and help other developers in the community.

Disadvantages of Open-Source Operating System

1. **Security risk:** The source code is available for everyone and anyone can analyze the source code and search for vulnerabilities and break into the operating system easily.
2. **Complicated:** Using an open-source operating system requires a little technical knowledge as compared to using a closed-source operating system such as Windows and Mac OS.
3. **No support:** As Open source operating systems are open-sourced and no company or organization is responsible for the maintenance of the operating system, there will be no help desk to address the problems encountered while using an open-source operating system. Even though open source operating systems have a developer community, unlike the community of closed source operating systems, There might be chances of not getting a proper diagnosis of the problem.

Operating System for Personal Computers:

An operating system is the most important software that runs on a computer. Personal computer operating system provides a good interface to a single user. Personal computer operating systems are widely used for word processing, spreadsheets and Internet access. The three most common operating systems for personal computers are Microsoft Windows, macOS, and Linux.

Operating System for Workstations :

Workstation is a computer which requests access to the LAN and switch services to responds the requests via switch to perform dedicated task with having enhanced features. The device which performs dedicated task with having enhanced features. In workstation devices must be installed the Graphics User Interface (GUI) to improve the performance of system. In workstation, Operations are in forms of Business, engineering etc.. Operating system used in workstation is: Unix, Linux or Windows NT.

Operating System for handheld devices:

A handheld computer is a computer that can conveniently be stored in a pocket (of sufficient size) and used while you're holding it. Today's handheld computers, which are also called Personal Digital Assistants, can be divided into those that accept handwriting as input and those with small keyboards.

Hewlett-Packard has introduced the first handheld computer with a color display. A number of companies now combine voice and data telephone service using cellular telephone or other wireless technologies with the handheld computer in a single device.

Handheld computers are typically used for personal information manager, types of applications, maintaining schedules, keeping names and phone numbers, doing simple calculations, taking notes, and, with a modem, exchanging e-mail and getting information from the Web. Nevertheless, this class of computer is widely sold and appreciated by many users.

Real Time Systems:

A real-time system means that the system is subjected to real-time, i.e., the response should be guaranteed within a specified timing constraint or the system should meet the specified deadline.
Types of real-time systems based on timing constraints are
1. Hard real-time system
2. Soft real-time system

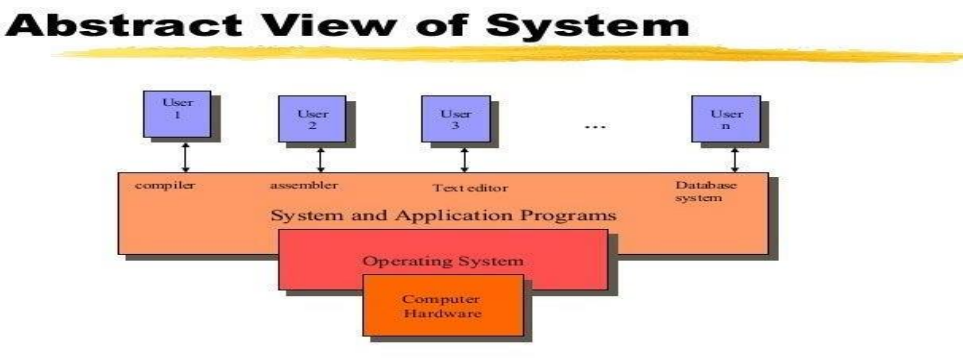
Hard real-time system

This type of system can never miss its deadline. Missing the deadline may have disastrous consequences. The usefulness of results produced by a hard real-time system decreases abruptly and may become negative if tardiness increases. Tardiness means how late a real-time system completes its task with respect to its deadline.
Example: Flight controller system.

Soft real-time system

This type of system can miss its deadline occasionally with some acceptably low probability. Missing the deadline has no disastrous consequences. The usefulness of results produced by a soft realtime system decreases gradually with an increase in tardiness.
Example: Telephone switches.

Resource Abstraction:



Resource abstraction Resource abstraction is the process of "hiding the details of how the hardware operates ,there by making computer hardware relatively easy for an application programmer to use". One way in whichthe operating system might implement resource abstraction is to provide a single abstract disk interface which will be the same for both the hard disk and floppy disk. Such an abstraction saves the programmer from needing to learn the details of both hardware interfaces. Instead, the programmer only needs to learn thedisk abstraction provided by the operating system.

UNIT – II

1. What is Process? Explain Process Control Block?

While creating a process the operating system performs several operations. To identify the processes, it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of all the processes.

A process control block (PCB) contains information about the process, i.e. registers, quantum, priority, etc. The process table is an array of PCB's that means logically contains a PCB for all of the current processes in the system.

Process Control Block

[

Pointer: It is a stack pointer which is required to be saved when the process is switched from one state to another to retain the current position of the process.

Process state: It stores the respective state of the process.

Process number: Every process is assigned with a unique id known as process ID or PID which stores the process identifier.

Program counter: It stores the counter which contains the address of the next instruction that is to be executed for the process.

Register: These are the CPU registers which includes: accumulator, base, registers and general purpose registers.

Memory limits: This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.

Open files list: This information includes the list of files opened for a process.

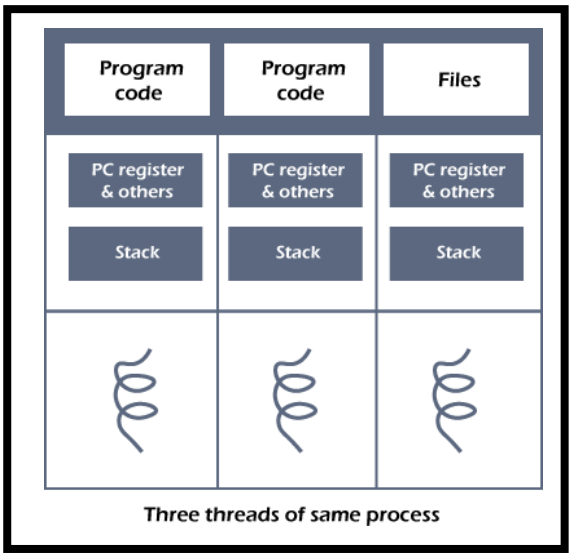
Miscellaneous accounting and status data: This field includes information about the amount of CPU used, time constraints, jobs or process number, etc. The process control block stores the register content also known as execution content of the processor when it was blocked from running.

When the process makes a transition from one state to another, the operating system updates its information in the process's PCB. The operating system maintains Pointers to each process's PCB in a process table so that it can access the PCB quickly.

2. What are threads? Explain about types of threads?

A thread is a sequential flow of tasks within a process. Each thread has its own set of registers and stack space. There can be multiple threads in a single process having the same or different functionality.

Eg: While playing a movie on a device the audio and video are controlled by different threads in the background.



The process can be split down into so many threads. For example, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

User-level thread

The operating system does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes.

Examples: Java thread, POSIX threads, etc.

Advantages of User-level threads

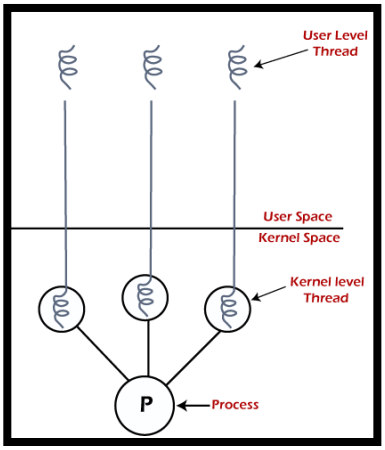
1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

Disadvantages of User-level threads

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.

Kernel level thread

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue.



Advantages of Kernel-level threads

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

Disadvantages of Kernel-level threads

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

4. Explain the Preemptive Scheduling Algorithms?

Preemptive Scheduling is defined as the scheduling which is done when the process changes from running state to ready state or from waiting for the state to ready state. In this, the resources are allocated to execute the process for a certain period. After this, the process is taken away in the middle and is placed in the ready queue its bursts time is left and this process will stay in ready line until it gets its turn to execute.

Suppose if a process which has the highest priority arrives, then this process does not wait for the complete execution of the current process. Instead of it, the ongoing process is interrupted in between and is placed in the ready queue until the process which has the highest priority does its execution. Thus in this way, all the processes which are in the available line get some time to run.

Example of preemptive scheduling: Round robin scheduling, priority scheduling, and shortest job first (SJF) scheduling

Explanation

Suppose there are four processes P1, P2, P3 and P4 whose arrival time and burst time are given in the table below

Process	Arrival Time	CPU Burst Time
P1	3	2
P2	2	4
P3	0	6
P4	1	4

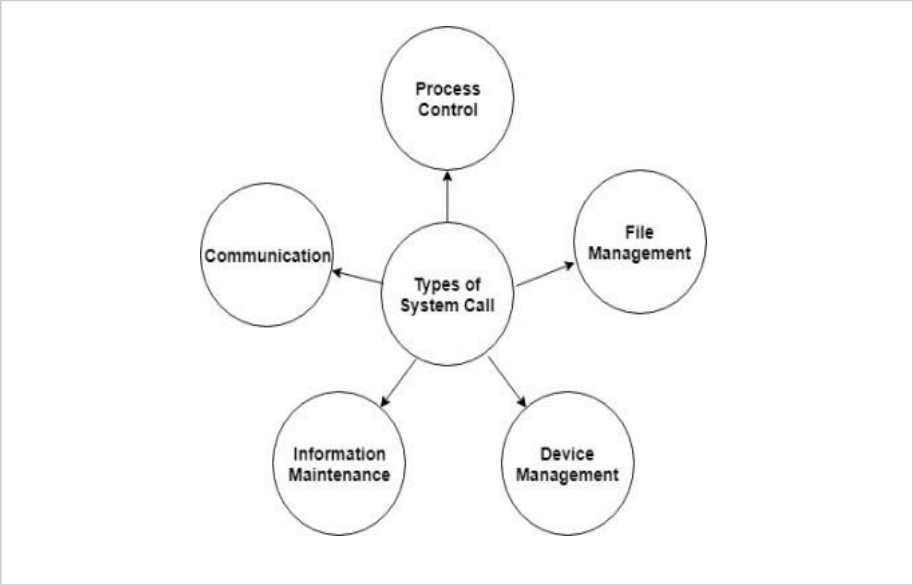
- Here, the process P3 arrives first i.e at time 0. As we know there is no process in the queue. So, the CPU is allocated to the process P3.
- When the process P3 was executing, the process P4 interrupts in-between as it arrives at time 1. Now the time left for the complete execution of the process P3 is 5 ms. This time is more than the time required by the process P4 for its implementation. So, process P4 is allocated to the CPU.
- When the process P4 was executing, the process P2 arrival time is reached. Now the time left for the execution of P4 is 3 ms. This time is less than the time required by process P2 and process P3. So, now P4 will continue.
- When P4 is ongoing with its execution, process P1 arrives. Now the time left for the implementation of the process P4 is 2ms. This time is equal to the time required by the process P1 which is 2ms. So, P4 will continue its execution.
- When the process P4 completes its execution, CPU will be allocated to the process P1 as its burst time is less than the other processes.
- After the completion of process P1, the process P2 will be executed, and in the end process, P3 will be executed.

5. Explain about System calls.

System calls are usually made when a process in user mode requires access to a resource. Then it requeststhe kernel to provide the resource via a system call.

Types of System Calls

There are mainly five types of system calls. These are explained in detail as follows



Process Control

These system calls deal with processes such as process creation, process termination etc.

File Management

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

Device Management

These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance

These system calls handle information and its transfer between the operating system and the user program.

Communication

These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

Some of the examples of all the above types of system calls in Windows and Unix are given as follows –

Types of System Calls	Windows	Linux
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Management	CreateFile() ReadFile() WriteFile() CloseHandle())	open() read() write() close()
Device Management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl()read()write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()

6. Write about different types of Schedulers.

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

1. **Long-Term Scheduler**
2. **Short-Term Scheduler**
3. **Medium-Term Scheduler**

- **Long Term Scheduler:**

It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

- **Short Term Scheduler:**

It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

- **Medium Term Scheduler:**

Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in- charge of handling the swapped out-processes.

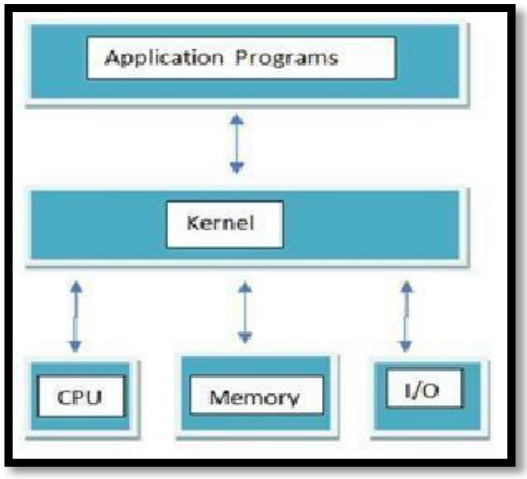
A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix

7. Explain about kernels?

Kernel is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again. It is responsible for various tasks such as disk management, task management, and memory management.

It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute. It basically acts as an interface between user applications and hardware. The major aim of kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.



What are the process scheduling algorithms in Operating Systems?

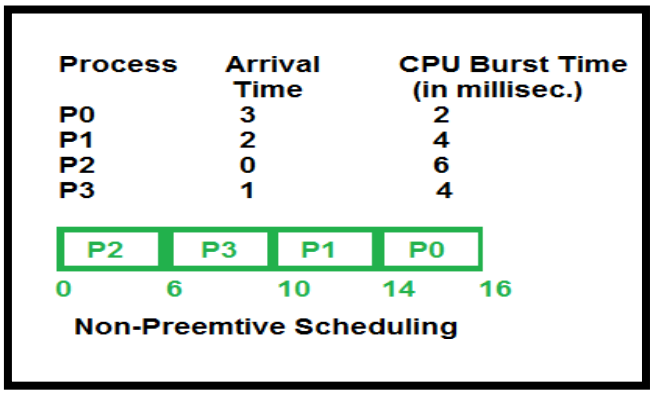
Process scheduling is an essential part of Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms.

1. First-Come, First-Served (FCFS) Scheduling
2. Shortest-Job-Next (SJN) Scheduling
3. Priority Scheduling
4. Shortest Remaining Time
5. Round Robin(RR) Scheduling
6. Multiple-Level Queues Scheduling

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the 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.

Explain about non- preemptive process scheduling policies.

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process. Algorithms based on non-preemptive scheduling are: Shortest Job First (SJF basically nonpreemptive) and Priority (non preemptive version), etc.

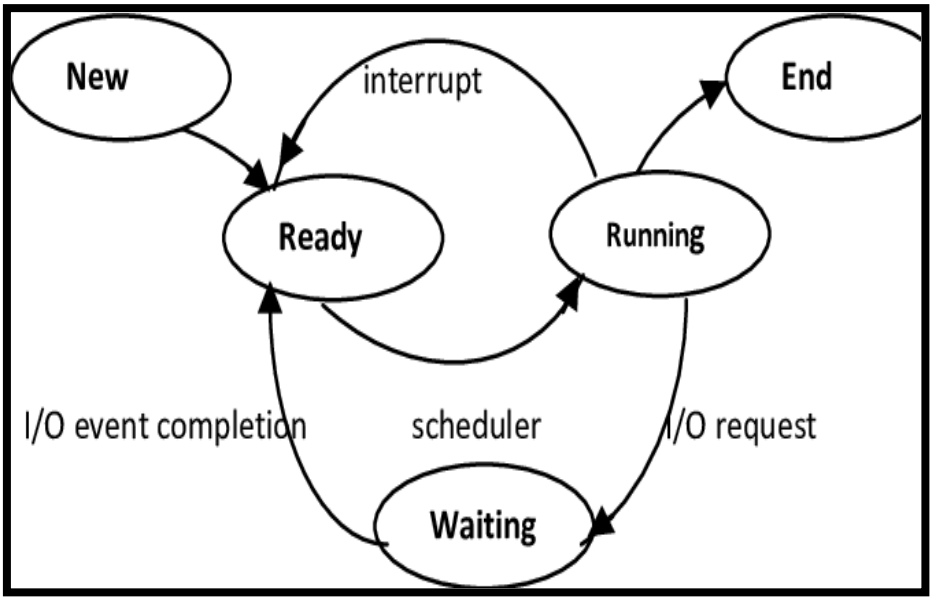


Difference between preemptive and non-preemptive scheduling

Preemptive Scheduling	Non-Preemptive Scheduling
In preemptive scheduling, the processes are allocated for a short period.	In non-preemptive scheduling, the process is allocated to the CPU, and the resource will hold the process until it completes its execution or changes its state to waiting for the state from ready state.
In this, we can interrupt the process in between of execution.	In this, we cannot interrupt until the process completes its execution or switches its state.
This type of schedule is flexible as each process in the ready queue gets some time to run CPU.	This type of scheduling is rigid.
Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling has not overheads of scheduling the processes.
There is a cost associated with the preemptive scheduling.	There is no cost associated with non-preemptive scheduling.
In preemptive scheduling, if a process which has high priority arrives in the ready queue, then the process which has low priority may starve.	In non-preemptive scheduling, if the process which has long burst time is executing, then the other method which has less burst time may starve.

Explain Process States.

The process, from its creation to completion, passes through various states. The minimum number of states is five.



The names of the states are not standardized although the process may be in one of the following states during execution.

- **New:** A program which is going to be picked up by the OS into the main memory is called a new process.
- **Ready:** Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory. The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.
- **Running/ Execution:** One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one. If we have n processors in the system then we can have n processes running simultaneously.
- **Block or waiting:** From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.
- When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.
- **Completion / Termination / End:** When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.

What are the responsible scheduling criteria?

The scheduling criterion is responsible for helping in the design of the good scheduler. These criteria are as follows −

- **CPU Utilization:** The scheduling algorithm should be designed in such a way that the usage of the CPU should be as efficient as possible.
- **Throughput:** It can be defined as the number of processes executed by the CPU in a given amount of time. It is used to find the efficiency of a CPU.
- **Response Time:** The Response time is the time taken to start the job when the job enters the queues so that the scheduler should be able to minimize the response time.

  Response time = Time at which the process gets the CPU for the first time - Arrival time.
- **Turnaround time:** Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to its completion.

  Turnaround time = Burst time + Waiting time

  or

  Turnaround time = Exit time - Arrival time
- **Waiting time:** The Waiting time is nothing but where there are many jobs that are competing for the execution, so that the Waiting time should be minimized.

  Waiting time = Turnaround time - Burst time
- **Fairness:** For schedulers there should be fairness for making sure that the processes get the fair shareof chances to be executed.

UNIT – III
PROCESS MANAGEMENT

1. Explain Deadlock Prevention techniques?

Deadlock Prevention: Deadlock prevention is eliminating one of the necessary conditions of deadlock so that only safe requests are made to OS and the possibility of deadlock is excluded before making requests.

Deadlock prevention techniques:

Deadlock prevention techniques refer to violating any one of the four necessary conditions.

- Mutual Exclusion
- Hold and Wait
- No preemption
- Circular Wait

Mutual Exclusion

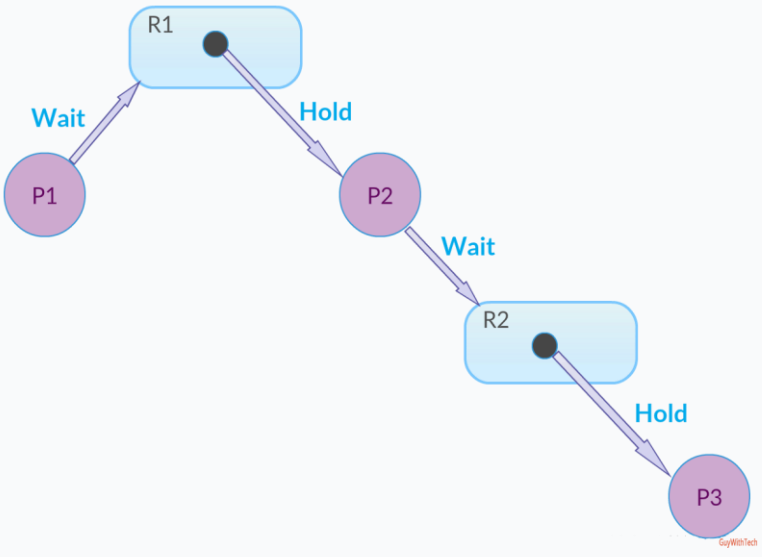
A mutual exclusion means that unshareable resources cannot be accessed simultaneously by processes. Shared resources do not cause deadlock but some resources can't be shared among processes, leading to a deadlock.

For Example: read operation on a file can be done simultaneously by multiple processes, but write operation cannot write. Write operation requires sequential access. so, some processes have to wait while another process is doing a write operation.

Hold and Wait

Hold and wait is a condition in which a process is holding one resource while simultaneously waiting for another resource that is being held by another process. The process cannot continue till it gets all the required resources.

In the diagram given below:

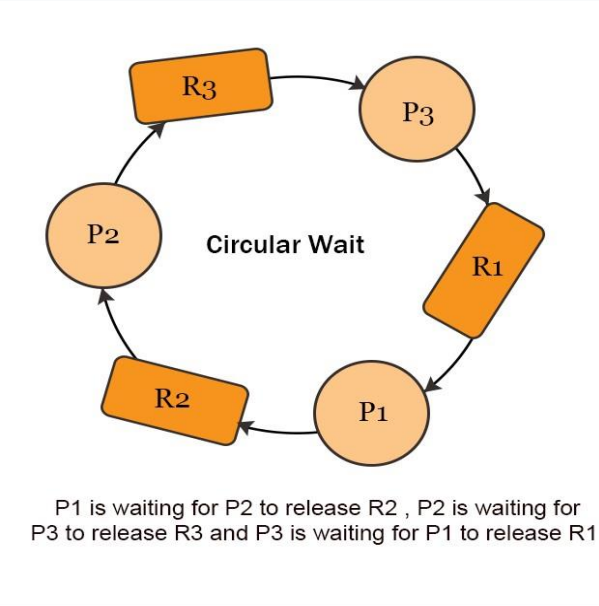


No preemption

A process P1 is waiting for some resource, and there is another process P2 that is holding that resource and is blocked waiting for some other resource. Then the resource is taken from P2 and allocated to P1. This way process P2 is preempted and it requests again for its required resources to resume the task.

Circular Wait

In circular wait, two or more processes wait for resources in a circular order. We can understand this better by the diagram given below:



P1 is waiting for P2 to release R2 , P2 is waiting for P3 to release R3 and P3 is waiting for P1 to release R1

2. Explain methods for handling Deadlocks?

Methods of handling deadlocks are mainly three approaches to deals.

1. Deadlock Prevention
2. Deadlock Avoidance
3. Deadlock Detection

1. Deadlock Prevention: Deadlock prevention is eliminating one of the necessary conditions of deadlock so that only safe requests are made to OS and the possibility of deadlock is excluded before making requests.

Deadlock prevention techniques:

Deadlock prevention techniques refer to violating any one of the four necessary conditions.

- Mutual Exclusion
- Hold and Wait
- No preemption
- Circular Wait

2. Deadlock Avoidance: Deadlock Avoidance is a process used by the Operating System to avoid Deadlock. Operating System avoids Deadlock by knowing the maximum resources requirements of the processes initially, and also Operating System knows the free resources available at that time. Operating System tries to allocate the resources according to the process requirements and checks if the allocation can lead to a safe state or an unsafe state. If the resource allocation leads to an unsafe state, then Operating System does not proceed further with the allocation sequence.

Two techniques to avoid deadlock:

- Process initiation denial
- Resource allocation denial

3. Deadlock Detection: Deadlock detection is used by employing an algorithm that tracks the circular waiting and killing one or more processes so that deadlock is removed. The system state is examined periodically to determine if a set of processes is deadlocked.

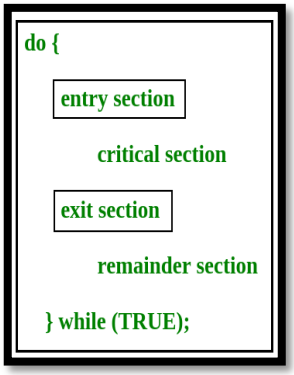
A deadlock is resolved by aborting and restarting a process, relinquishing all the resources that the process held.

- This technique does not limit resources access or restrict process action.
- Requested resources are granted to processes whenever possible.
- It never delays the process initiation and facilitates online handling.
- The disadvantage is the inherent pre-emption losses.

3. Explain about Critical Section Problems

Process synchronization: A situation, where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a race condition. To guard against the race condition, we need to ensure that only one process at a time can be manipulating the variable counter. To make this, we require some form of synchronization of the processes.

Critical Section Problem



Consider a system consisting of n processes {P0,P1, ,Pn-1}. Each process has a segment of code, called a critical section, in which the process may be changing common variables, updating a table, writing a file, and so on. When one process is executing in its critical section, no other process is to be allowed to execute in its critical section. Thus, the execution of critical sections by the processes is mutually exclusive in time. 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. Apicture(right side) showing general structure of a process Pi.

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 some processes wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

3) **Bounded Waiting:** There exists a bound 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.

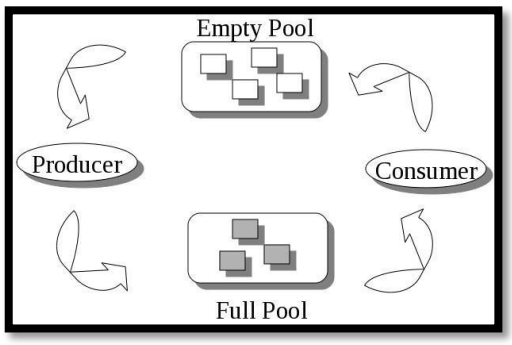
3. Explain about Classical problems of Synchronization

The classical problems of synchronization are as follows:

1. **Bound-Buffer problem**
2. **Sleeping barber problem**
3. **Dining Philosophers problem**
4. **Readers and writers problem**

1. Bound-Buffer problem

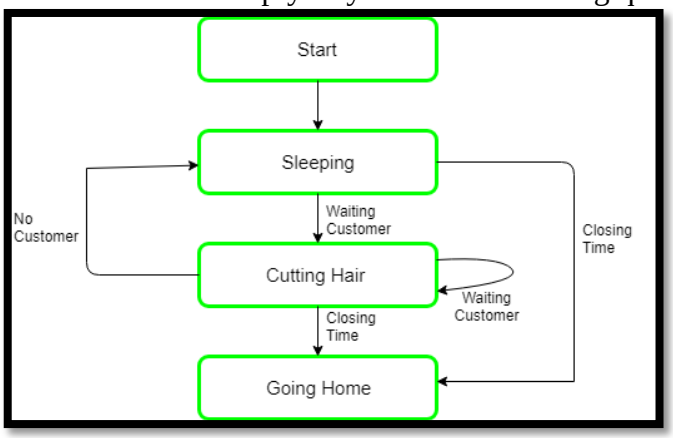
It is also known as the Producer-Consumer problem. In this problem, there is a buffer of n slots, and each buffer is capable of storing one unit of data. There are two processes that are operating on the buffer – Producer and Consumer. The producer tries to insert data and the consumer tries to remove data.



If the processes are run simultaneously they will not yield the expected output. The solution to this problem is creating two semaphores, one full and the other empty to keep a track of the concurrent processes.

2. Sleeping Barber Problem

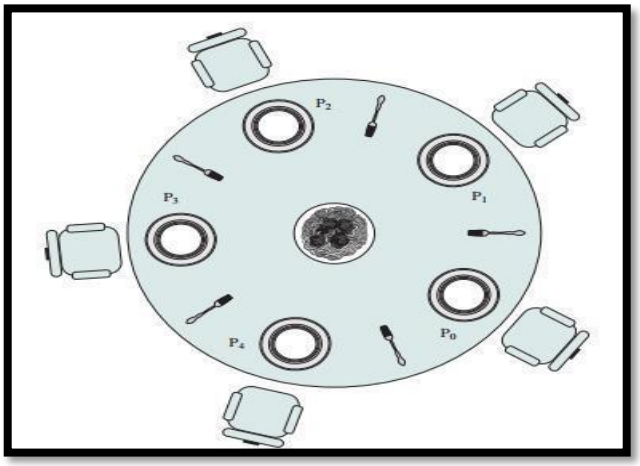
This problem is based on a hypothetical barbershop with one barber. When there are no customers the barber sleeps in his chair. If any customer enters he will wake up the barber and sit in the customer chair. If there are no chairs empty they wait in the waiting queue.



3. Dining Philosopher's problem

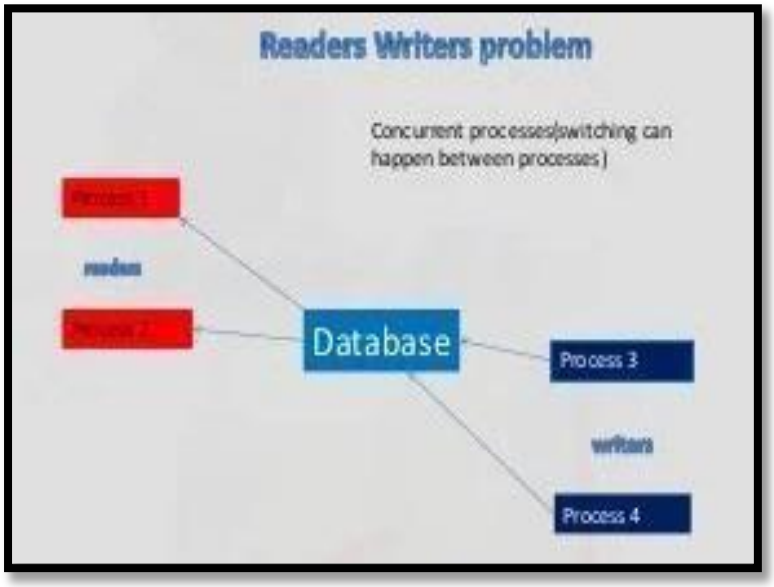
This problem states that there are K numbers of philosophers sitting around a circular table with one chopstick placed between each pair of philosophers. The philosopher will be able to eat if he can pick up two chopsticks that are adjacent to the philosopher. This problem deals with theallocation of limited resources

.



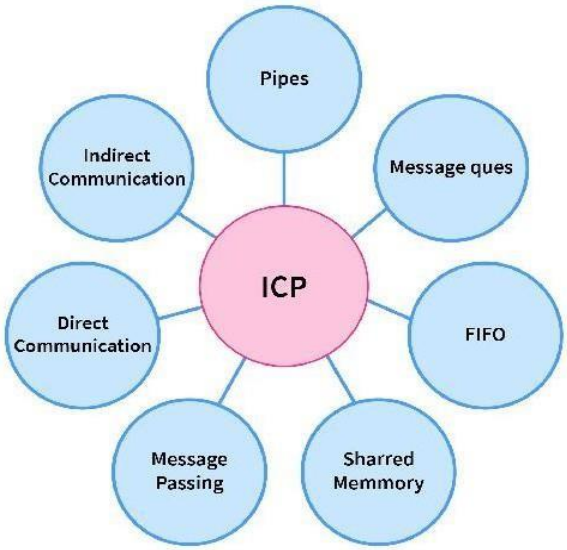
4. Readers and Writers Problem

This problem occurs when many threads of execution try to access the same shared resources ata time. Some threads may read, and some may write. In this scenario, we may get faulty outputs.



4. Explain about Inter Process Communication methods

The different approaches to implement inter process communication are given as follows –



Pipes: It is a half-duplex method (or one-way communication) used for IPC between two related processes. It is like a scenario like filling the water with a tap into a bucket. The filling process is writing into the pipe and the reading process is retrieved from the pipe.

Message Queues: We have a linked list to store messages in a kernel of OS and a message queue is identified using "message queue identifier".

FIFO: Used to communicate between two processes that are not related. Full-duplex method - Process P1 is able to communicate with Process P2, and vice versa.

Shared Memory: Multiple processes can access a common shared memory. Multiple processes communicate by shared memory, where one process makes changes at a time and then others view the change. Shared memory does not use kernel.

Message Passing: In IPC, this is used by a process for communication and synchronization.
Processes can communicate without any shared variables, therefore it can be used in a distributed environment on a network. It is slower than the shared memory technique. It has two actions sending (fixed size message) and receiving messages.

Direct Communication: In this, processes that wanna communicate must name the sender or receiver. A pair of communicating processes must have one link between them. A link establishes between every pair of communicating processes.

Indirect Communication: Pairs of communicating processes have shared mailboxes. Link is established between pairs of processes. Sender process puts the message in the port or mailbox of a receiver process and receiver process takes out (or deletes) the data from the mailbox.

Shorts Answers

1. Explain about Semaphores?

A semaphore is a signalling mechanism, and a thread that is waiting on a semaphore can be signalled by another thread. Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations wait and signal that are used for process synchronization.

Wait: The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
   {
     while (S<=0);
     S--;
   }
```

Signal: The signal operation increments the value of its argument S.

```
signal(S)
   {
     S++;
   }.
```

2. Explain the necessary Deadlock condition?

The four necessary conditions for a deadlock to arise are as follows.

Mutual Exclusion: Only one process can use a resource at any given time i.e. the resources are non-sharable.

Hold and wait: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.

No preemption: The resource can be released by a process voluntarily i.e. after execution of the process.

Circular Wait: A set of processes are waiting for each other in a circular fashion.

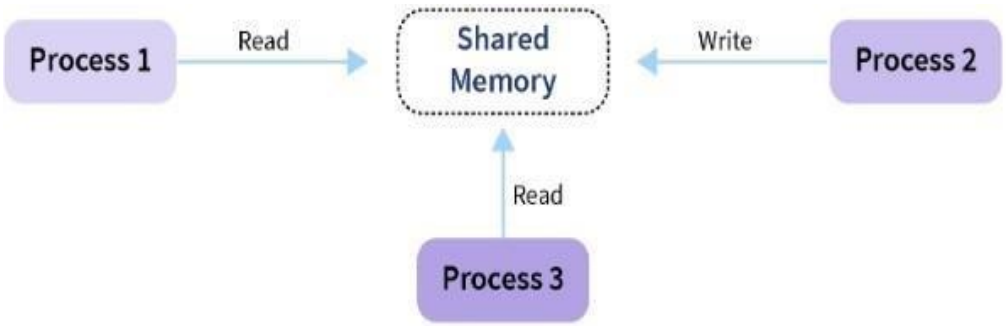
For example, let's say there are a set of processes {$\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$} such that $\mathbf{P_0}$ depends on $\mathbf{P_1}$, $\mathbf{P_1}$ depends on $\mathbf{P_2}$, $\mathbf{P_2}$ depends on $\mathbf{P_3}$ and $\mathbf{P_3}$ depends on $\mathbf{P_0}$.

This creates a circular relation between all these processes and they have to wait forever to be executed.

3. What is Synchronization? Explain?

Synchronization is the way by which processes that share the same memory space are managed in an operating system. It helps maintain the consistency of data by using variables or hardware so that only one process can make changes to the shared memory at a time. There are various solutions for the same such as **Semaphores, Mutex Locks, Synchronization Hardware**, etc.

Example: A cloud storage service like **Dropbox, Google Drive or Nextcloud,** that are synchronizes changes file data across multiple Internet-connected devices. When a file is changed on any synchronized device, the data is automatically updated on all of them.



4. Explain Deadlock characteristics?

Deadlock characterization describes the distinctive features that are the cause of deadlock occurrence. Deadlock is a condition in the multiprogramming environment where the executing processes get stuck in the middle of execution waiting for the resources that have been held by the other waiting processes thereby preventing the execution of the processes.

a. Deadlock Prerequisites
b. Systems Resource Allocation Graph

Deadlock Prerequisites

Mutual Exclusion: Only one process can use a resource at any given time i.e. the resources are non-sharable.

Hold and wait: A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.

No preemption: The resource can be released by a process voluntarily i.e. after execution of the process.

Circular Wait: A set of processes are waiting for each other in a circular fashion.

For example, let's say there are a set of processes {$\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$} such that $\mathbf{P_0}$ depends on $\mathbf{P_1}$, $\mathbf{P_1}$ depends on $\mathbf{P_2}$, $\mathbf{P_2}$ depends on $\mathbf{P_3}$ and $\mathbf{P_3}$ depends on $\mathbf{P_0}$.

This creates a circular relation between all these processes and they have to wait forever to be executed.

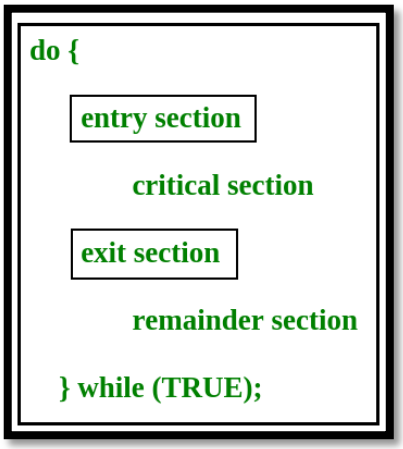
System Resource Allocation Graph

The system reallocation graph is a directed graph that briefs you about the deadlock more precisely. Like every graph, it also has a set of vertices and a set of edges. Further, the set of vertices can be classified into two types of nodes P and R. Where P is the set of vertices indicating the set of active processes and R is the set of vertices indicating all types of resources in the system.

When a process requests for a resource it denoted by the request edge in the resource-allocation graph. The request edge is a directed edge from the requesting process Pi to requested resource Rj i.e. Pi -> Rj.

5. Explain the critical section?

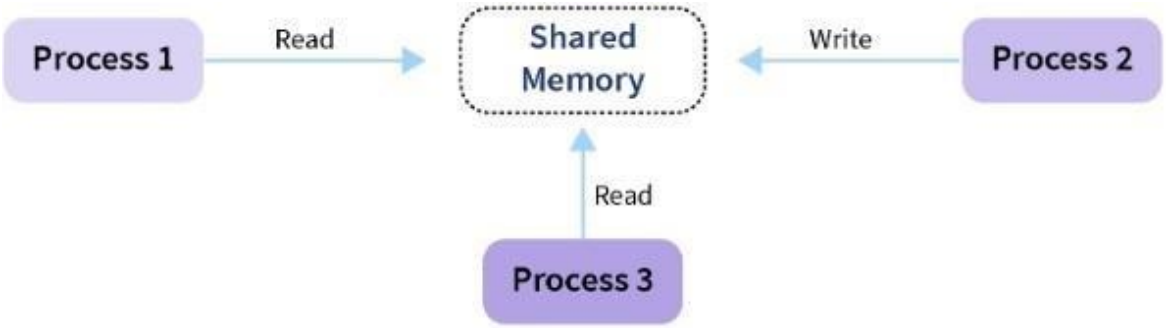
The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.



In the above diagram, the entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

6. What is Process Synchronization?

An operating system is software that manages all applications on a device and basically helps in the smooth functioning of our computer. Because of this reason, the operating system has to perform many tasks, and sometimes simultaneously. This isn't usually a problem unless these simultaneously occurring processes use a common resource.



Let us take a look at why exactly we need Process Synchronization. For example, If a process1 is trying to read the data present in a memory location while another process2 is trying to change the data present at the same location, there is a high chance that the data read by the process3 will be incorrect. These three processes are works simultaneously.

7. Write about Deadlock Avoidance?

Deadlock Avoidance is used by Operating System to Avoid Deadlock in the System. The processes need to specify the maximum resources needed to the Operating System so that the Operating System can simulate the allocation of available resources to the requesting processes and check if it is possible to satisfy the need of all the processes requirements.

In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution. The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

The resource allocation state of a system can be defined by the instances of available and allocated resources, and the maximum instance of the resources demanded by the processes.

8. What is Deadlock Recovery?

Deadlock recovery performs when a deadlock is detected. When deadlock detected, then our system stops working, and after the recovery of the deadlock, our system start working again. after the detection of deadlock, a method/way must require to recover that deadlock to run the system again. The method/way is called as deadlock recovery.

c. Deadlock recovery through preemption
d. Deadlock recovery through rollback
e. Deadlock recovery through killing processes

Deadlock Recovery through Preemption

The ability to take a resource away from a process, have another process use it, and then give it back without the process noticing. It is highly dependent on the nature of the resource. Deadlock recovery through preemption is too difficult or sometime impossible.

Deadlock Recovery through Rollback

In this case of deadlock recovery through rollback, whenever a deadlock is detected, it is easy to see which resources are needed. To do the recovery of deadlock, a process that owns a needed resource is rolled back to a point in time before it acquired some other resource just by starting one of its earlier checkpoints.

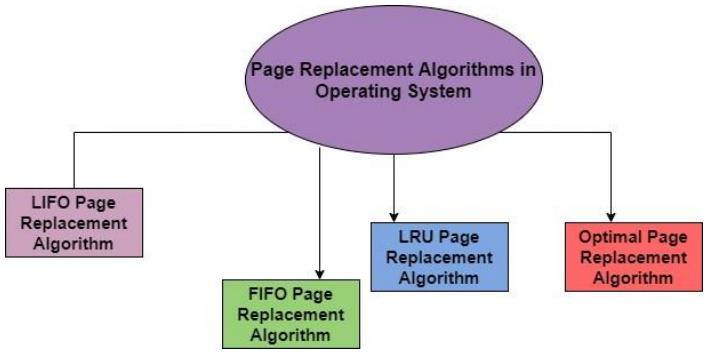
Deadlock Recovery through Killing Processes

This method of deadlock recovery through killing processes is the simplest way of deadlock recovery. Sometime it is best to kill a process that can be return from the beginning with no ill effects.

UNIT – IV
MEMORYMANAGEMENT

TYPES OF PAGE REPLACEMENT ALGORITHMS

Page Replacement Algorithms: It is basically a memory error, and it occurs when the current programs attempt to access the memory page for mapping into virtual address space, but it is unable to load into the physical memory then this is referred to as Page fault.

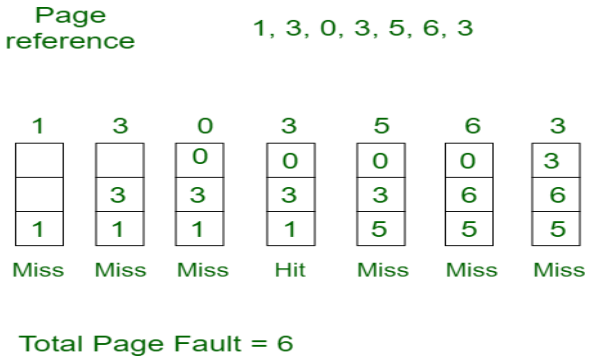


1. FIFO Page Replacement Algorithm

It is a very simple way of Page replacement and is referred to as First in First Out. This algorithm mainly replaces the oldest page that has been present in the main memory for the longest time.

- This algorithm is implemented by keeping the track of all the pages in the queue.
- As new pages are requested and are swapped in, they are added to the tail of a queue and the page which is at the head becomes the victim.
- This is not an effective way of page replacement but it can be used for small systems.

Example 1: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames. Find the number of page faults.



Advantages

- This algorithm is simple and easy to use.
- FIFO does not cause more overhead.

Disadvantages

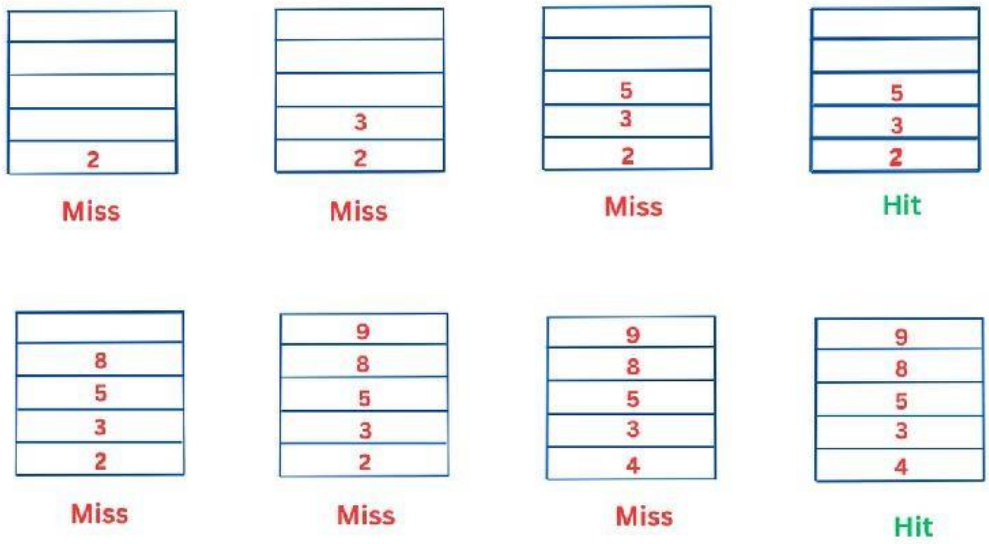
- This algorithm does not make the use of the frequency of **last used time rather** it just replaces the Oldest Page.
- There is an increase in **page faults** as page frames increases.
- The performance of this algorithm is the worst.

Implementation of FIFO in OS

Consider an example of the FIFO page fault replacement algorithm:

Let's suppose we have 5 page frames and we have page references: 2, 3, 5, 2, 8, 9, 4, 5. We have to find the total number of page faults.

Note: The first occurrence of any page will always be a page miss.



LRU Page Replacement Algorithm in OS:

This algorithm stands for "Least recent used" and this algorithm helps the Operating system to search those pages that are used over a short duration of time frame.

- The page that has not been used for the longest time in the main memory will be selected for replacement.
- This algorithm is easy to implement.
- This algorithm makes use of the counter along with the even-page.

Advantages of LRU

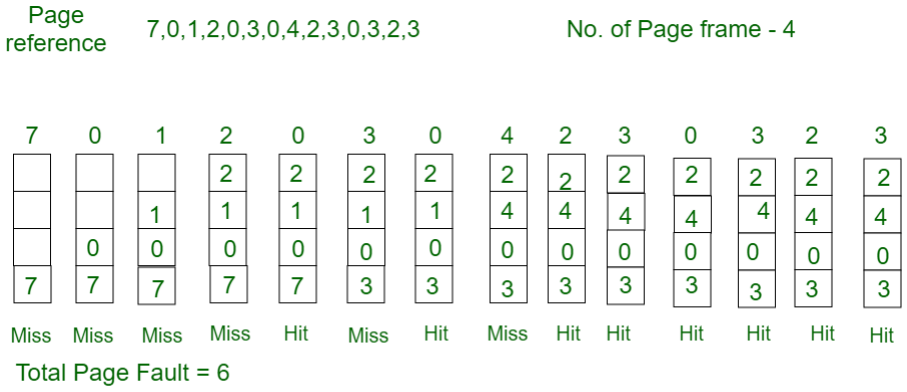
- It is an efficient technique.
- With this algorithm, it becomes easy to identify the faulty pages that are not needed for a long time.
- It helps in Full analysis.

Disadvantages of LRU

- It is expensive and has more complexity.
- There is a need for an additional data structure.

➢ In this algorithm, page will be replaced which is least recently used

Example-3: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults.



Here LRU has same number of page fault as optimal but it may differ according to question.

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults
0 is already their so —> 0 Page fault. when 3 came it will take the place of 7 because it is least recently used —>1 Page fault
0 is already in memory so —> 0 Page fault.
4 will takes place of 1 —> 1 Page Fault
Now for the further page reference string —> 0 Page fault because they are already available in the memory

Optimal Page Replacement Algorithm:

This algorithm mainly replaces the page that will not be used for the longest time in the future. The practical implementation of this algorithm is not possible.

- Practical implementation is not possible because we cannot predict in advance those pages that will not be used for the longest time in the future.
- This algorithm leads to less number of page faults and thus is the best-known algorithm

Also, this algorithm can be used to measure the performance of other algorithms.

Advantages of OPR

- This algorithm is easy to use.
- This algorithm provides excellent efficiency and is less complex.
- For the best result, the implementation of data structures is very easy

Disadvantages of OPR

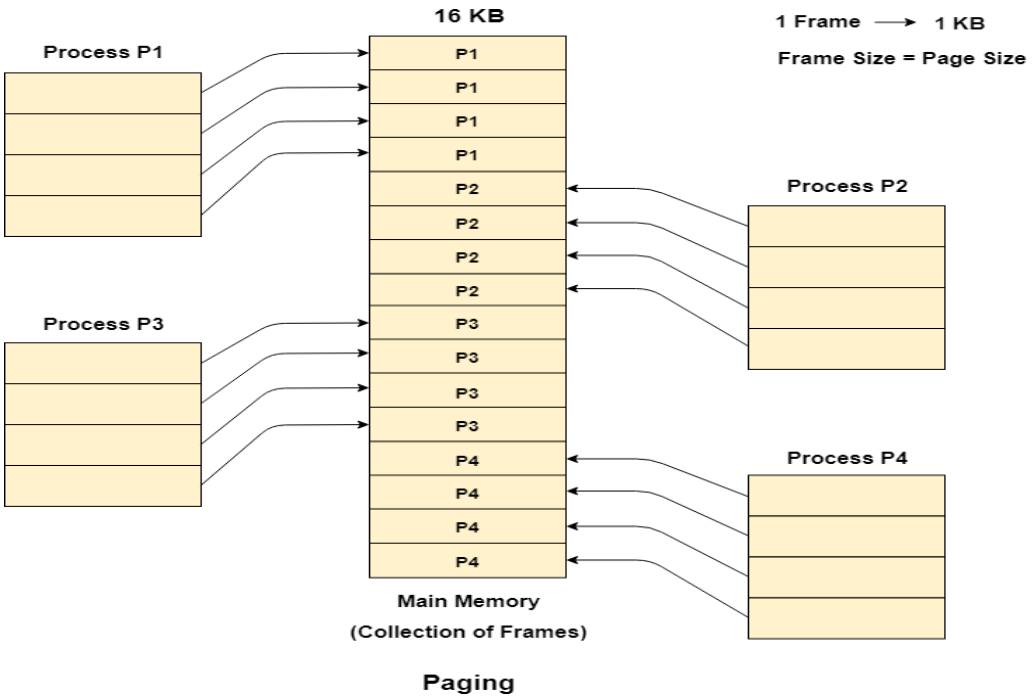
- In this algorithm future awareness of the program is needed.
- Practical Implementation is not possible because the operating system is unable to track the future request

Random Page Replacement Algorithm

As indicated from the name this algorithm replaces the page randomly. This Algorithm can work like any other page replacement algorithm that is LIFO, FIFO, Optimal, and LRU.

BRIEFLY EXPLAIN ABOUT PAGING.

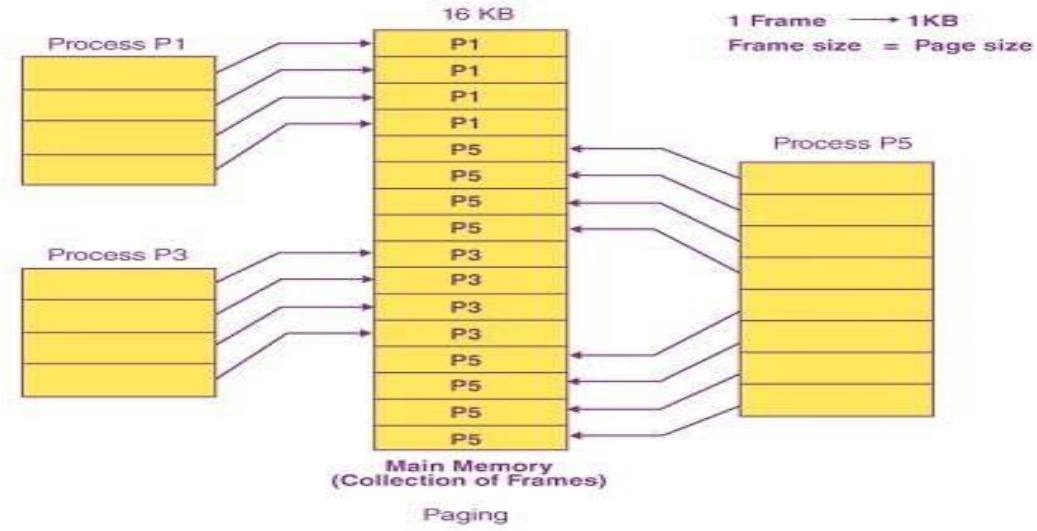
Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory.



Paging

Let us consider that, P2 and P4 are moved to waiting state after some time. Now, 8 frames become empty and therefore other pages can be loaded in that empty place. The process P5 of size 8 KB (8 pages) is waiting inside the ready queue.

Given the fact that, we have 8 non contiguous frames available in the memory and paging provides the flexibility of storing the process at the different places. Therefore, we can load the pages of process P5 in the place of P2 and P4.



Paging

Memory Management Unit:
The purpose of Memory Management Unit (MMU) is to convert the logical address into the physical address. The logical address is the address generated by the CPU for every page while the physical address is the actual address of the frame where each page will be stored.
When a page is to be accessed by the CPU by using the logical address, the operating system needs to obtain the physical address to access that page physically.

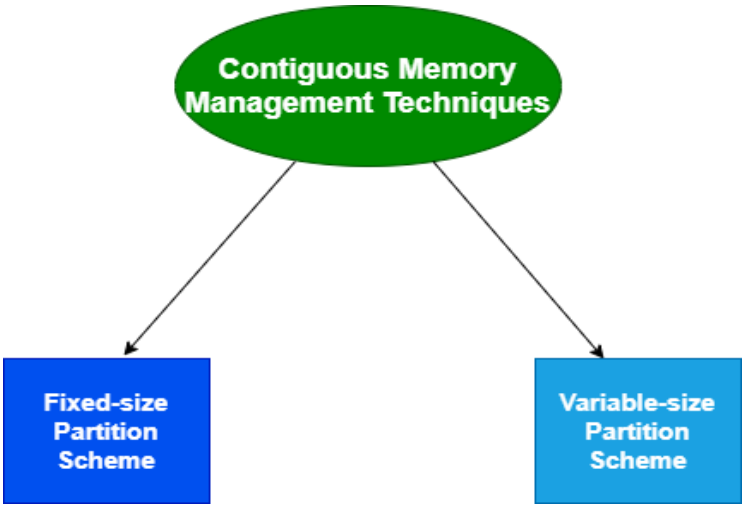
The logical address has two parts.

- Page Number
- Offset

Memory management unit of OS needs to convert the page number to the frame number.

EXPLAIN ABOUT CONTIGUOUS MEMORY ALLOCATION.

In the Contiguous Memory Allocation, each process is contained in a single contiguous section of memory. In this memory allocation, all the available memory space remains together in one place which implies that the freely available memory partitions are not spread over here and there across the whole memory space.
In Contiguous memory allocation which is a memory management technique, whenever there is a request by the user process for the memory then a single section of the contiguous memory block is given to that process according to its requirement. Contiguous Memory allocation is achieved just by dividing the memory into the fixed-sized partition.
The memory can be divided either in the fixed-sized partition or in the variable-sized partition in order to allocate contiguous space to user processes.

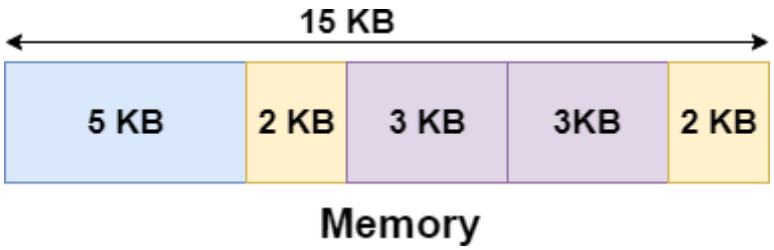


1. Fixed-size Partition Scheme: This technique is also known as Static partitioning. In this scheme, the system divides the memory into fixed-size partitions. The partitions may or may not be the same size. The size of each partition is fixed as indicated by the name of the technique and it cannot be changed.

In this partition scheme, each partition may contain exactly one process. There is a problem that this technique will limit the degree of multiprogramming because the number of partitions will basically decide the number of processes.

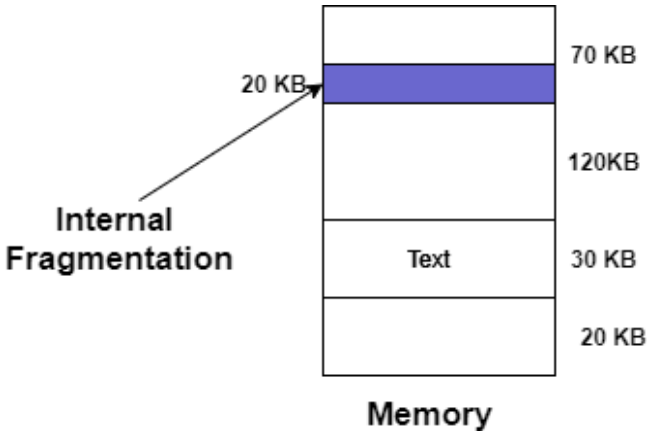
Whenever any process terminates then the partition becomes available for another process.

Example: Let's take an example of fixed size partitioning scheme, we will divide a memory size of 15 KB into fixed-size partitions:



It is important to note that these partitions are allocated to the processes as they arrive and the partition that is allocated to the arrived process basically depends on the algorithm followed.

If there is some wastage inside the partition then it is termed **Internal Fragmentation**.

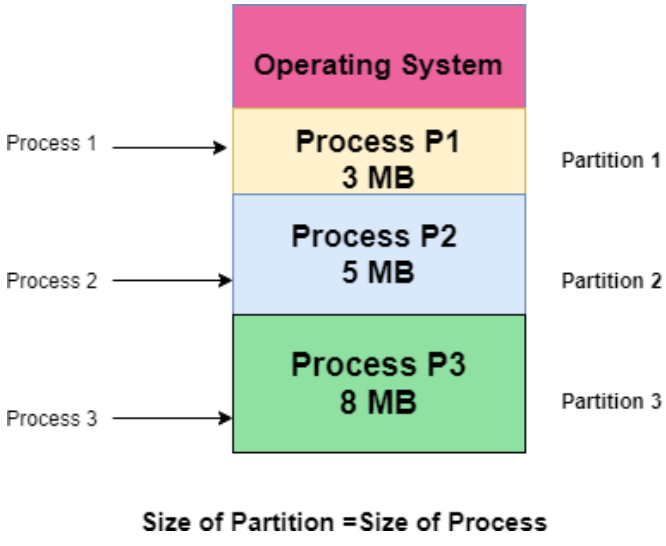


2. Variable-size Partition Scheme:

This scheme is also known as **Dynamic partitioning** and is came into existence to overcome the drawback i.e internal fragmentation that is caused by **Static partitioning**. In this partitioning, scheme allocation is done dynamically.

The size of the partition is not declared initially. Whenever any process arrives, a partition of size equal to the size of the process is created and then allocated to the process. Thus the size of each partition is equal to the size of the process.

As partition size varies according to the need of the process so in this partition scheme there is no **internal fragmentation.**



Advantages:

The advantages of a fixed-size partition scheme are:

- Because all of the blocks are the same size, this scheme is simple to implement. All we have to do now is divide the memory into fixed blocks and assign processes to them.
- It is easy to keep track of how many blocks of memory are left, which in turn decides how many more processes can be given space in the memory.
- As at a time multiple processes can be kept in the memory, this scheme can be implemented in a system that needs multiprogramming.

Disadvantages:

Though the fixed -size partition scheme has many advantages, it also has some disadvantages:

- As the size of the blocks is fixed, we will not be able to allot space to a process that has a greater size than the block.
- The size of the blocks decides the degree of multiprogramming, and only that many processes can remain in the memory at once as the number of blocks.
- If the size of the block is greater than the size of the process, we have no other choice but to assign the process to this block, but this will lead to much empty space left behind in the block. This empty space could've been used to accommodate a different process. This is called internal fragmentation. Hence, this technique may lead to space wastage.

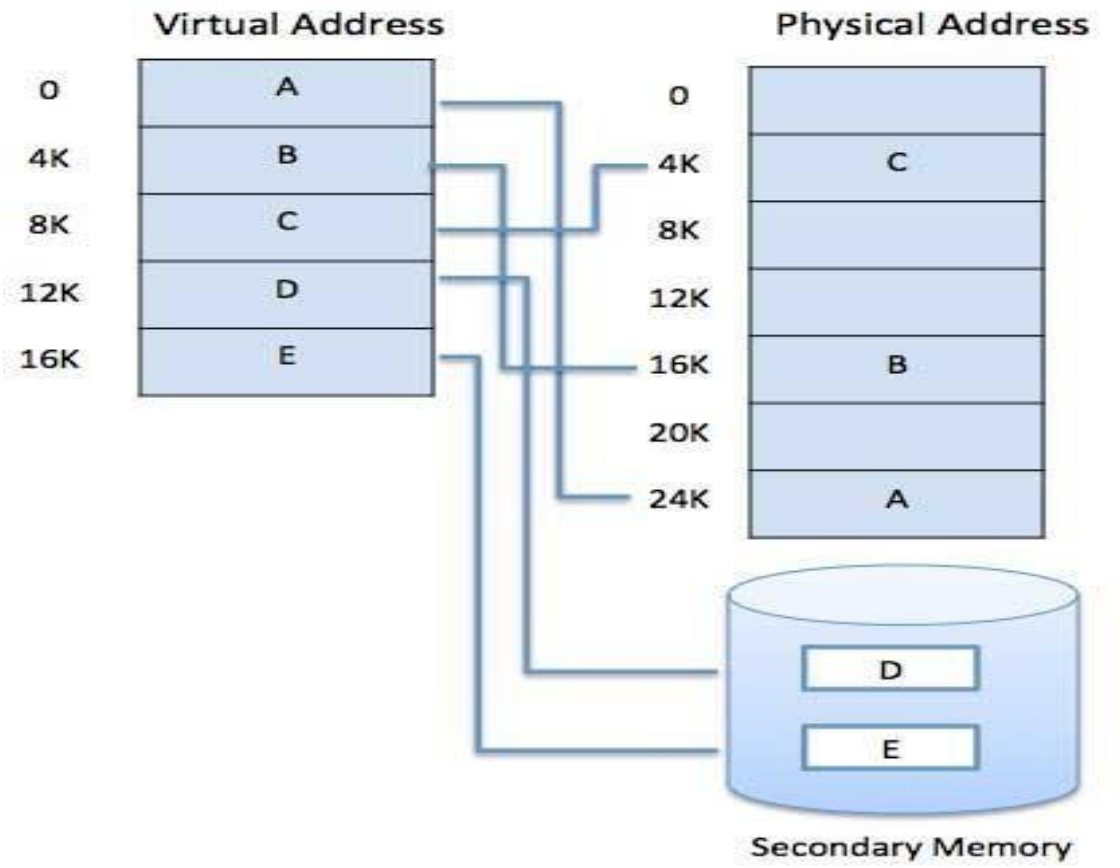
BRIEFLY EXPLAIN ABOUT VIRTUAL MEMORY

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.



Advantages of Virtual Memory

- The degree of Multiprogramming will be increased.
- User can run large application with less real RAM.
- There is no need to buy more memory RAMs.

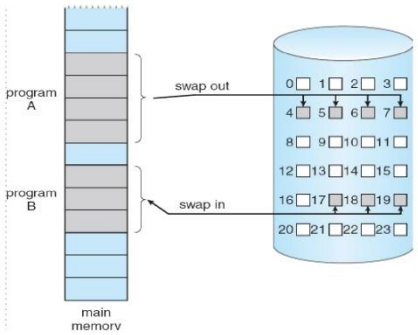
Disadvantages of Virtual Memory

- The system becomes slower since swapping takes time.
- It takes more time in switching between applications.
- The user will have the lesser hard disk space for its use.

SHORT ANSWERS

Explain About Demand Paging.

- To be simple in demand paging the entire process should be in the disk in the form of pages
- In this technique a page is brought into memory for its execution only when it is demanded
- It is a combination of paging and swapping



Demand Paging in OS

Here in the above diagram all the pages are loaded into backing store (hard disk). By the mechanism of swapping when the main memory requests the page Only then it is loaded from hard disk As main memory is small in size and cannot handle large programs only few pages are loaded into main memory after completing its execution it is swapped out simply and new process is then swapped in.

Discuss About Swapping.

Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called swap space.

The purpose of the swapping in operating system is to access the data present in the hard disk and bring it to RAM so that the application programs can use it. The thing to remember is that swapping is used only when data is not present in RAM

Advantages of Swapping:

- It helps the CPU to manage multiple processes within a single main memory.
- It helps to create and use virtual memory.
- It improves the main memory utilization.

Disadvantages of Swapping:

- If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.

Write About Logical Address.

- Users can access the logical address of the Program.
- **Logical Address** is the address generated by the CPU when a program is under execution.
- The user can access the physical address with the help of a logical address.

- Logical address is also known as **Virtual Address**. This is the address of instructions or data used by the program.
- Logical address is just like a reference which is used to access Physical Memory Location By the CPU.
- The set of all logical addresses that are generated by any program is referred to as Logical Address Space.
- The logical address does not exist physically in the memory and thus termed as a Virtual address.

Difference Between Fixed Partitioning And Variable Partitioning.

S.NO	Fixed partitioning	Variable partitioning
1.	In multi-programming with fixed partitioning the main memory is divided into fixed sized partitions.	In multi-programming with variable partitioning the main memory is not divided into fixed sized partitions.
2.	Only one process can be placed in a partition.	In variable partitioning, the process is allocated a chunk of free memory.
3.	It does not utilize the main memory effectively.	It utilizes the main memory effectively.
4.	There is presence of internal fragmentation and external fragmentation.	There is external fragmentation.
5.	Degree of multi-programming is less.	Degree of multi-programming is higher.
6.	It is more easier to implement.	It is less easier to implement.
7.	There is limitation on size of process.	There is no limitation on size of process.

UNIT- V
FILE AND I/O MANAGEMENT, OSSECURITY

File Operations In OS (or) Different File Operations:

There are mainly nine basic file operations in os:

- Creating a file
- Writing a file
- Reading a file
- Opening a file
- Renaming a file
- Deleting a file
- Truncating a file
- Appending a file
- Closing a file

1. **Creating a file**: Two steps are needed to create a file. First, one is checking whether the space is available, or not. If the space is available then made an entry for the new file must be made in the directory. The entry includes the name of the file, the path of the file etc.

2. **Writing file:** To write a file, we have to know two things one is the name of the file and the second is the information or data to be written on the file.

3. **Reading a file:** To read a file, first of all, we search the directories for the file. If the file is found, the system needs to keep a 'read' pointer to the location in the file where the next read is to take place once the reader has taken place, the read pointer is updated.

4. **Opening a file:** It is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it tells the operating system to invoke the open system call and passes the file name to the file system.

5. **Renaming a file:** This operation is used to rename the existing file.

6. **Deleting files:** To delete a file, first of all, search the directory for the named file, then release the file space and erase the directory entry.

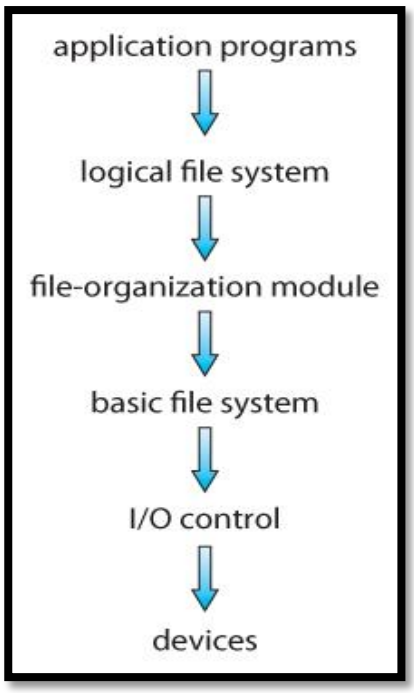
7. **Truncating a file:** To truncate a file, remove the file contents only, but the attributes areas it.

8. **Appending a file:** This operation adds data to the end of the file.

9. **Closing a file:** When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. In closing, it deallocates all the internal descriptors that were created when the file was opened.

Explain about File Structure

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.
Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.
The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.



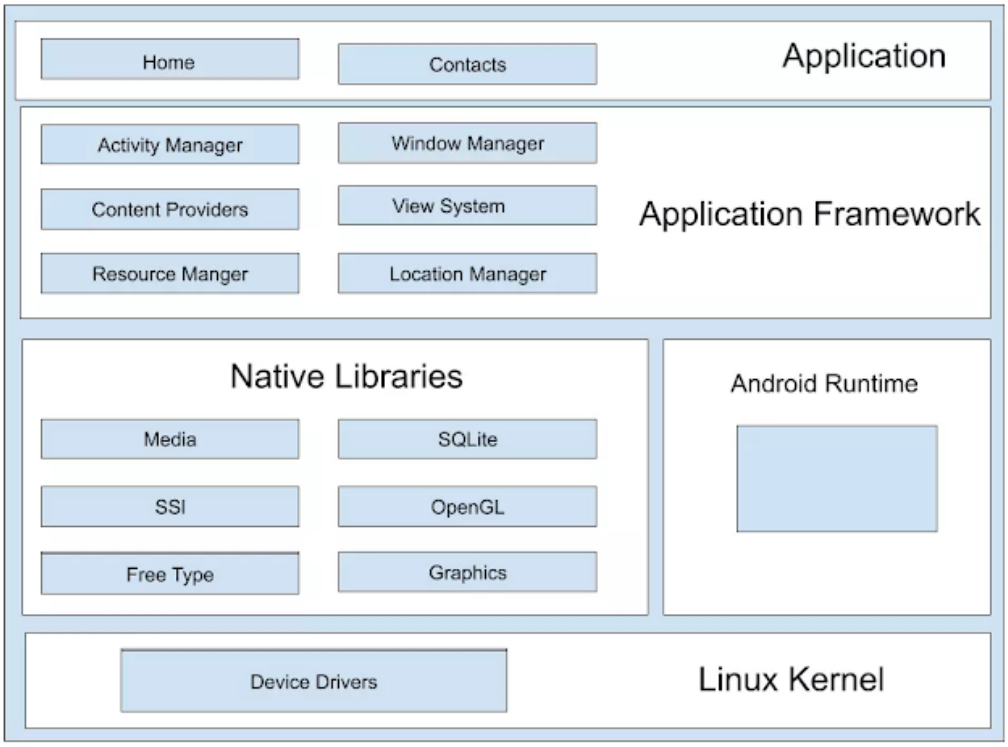
1. When an application program asks for a file, the first request is directed to the logical file system. The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
2. Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files, the logical blocks need to be mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.
3. Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
4. I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.

Explain Android Architecture With A Neat Diagram.

Android architecture:

Android architecture stack is group into five parts:

- Linux Kernel
- Native libraries
- Android Runtime
- Application Framework
- Applications



Android Architecture

1. Linux Kernel:
Linux Kernel is the core of the android operating system architecture that exists at the root of android architecture. It is responsible for device drivers, power management, memory management, device management, and resource management.

2. Native Libraries:
On the top of the Linux kernel, there are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libs), etc. The WebKit library is responsible for browser support, SQLite is for database, Free Type for font support, Media for playing and recording audio and video formats.

3. Android runtime:
In the android runtime, there are core libraries and DVM(Dalvik Virtual Machine) which is responsible for run android applications. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4. Android framework:
On the top of Native libraries and android runtime, there is an Android framework. Android framework includes Android APIs such as UI (User Interface), telephony, resources, locations, Content Providers(data), and package managers. It provides a lot of classes and interfaces for android application development.

5. Applications:
On top of the Android framework, there are applications. All applications such as home, contact, settings, games, gallery, browsers are using the android framework that uses android runtime and libraries. Android runtime and native libraries are using the Linux kernel.

Explain About File Allocation Methods.

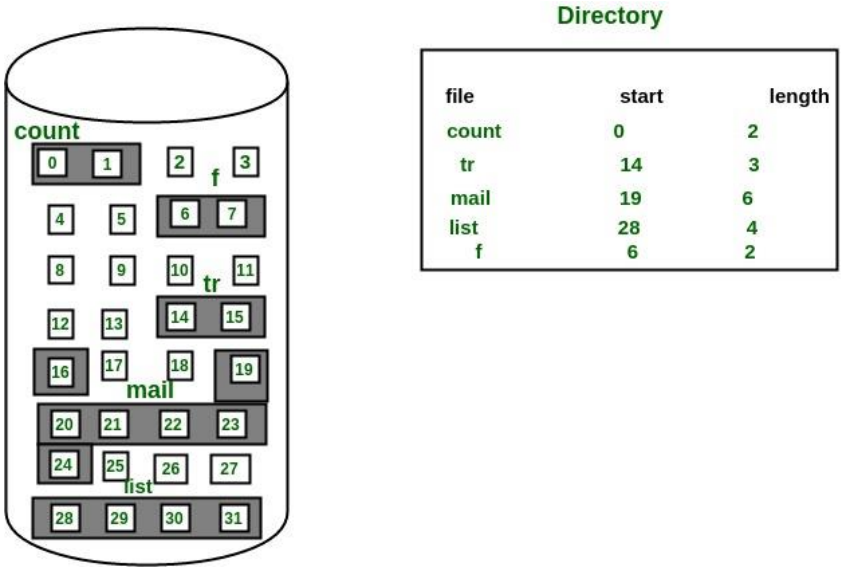
A file allocation method is a way that an operating system stores and retrieves files on a storage device, such as a hard drive or SSD.

Different types of file allocation methods

There are several different types of file allocation methods, such as:

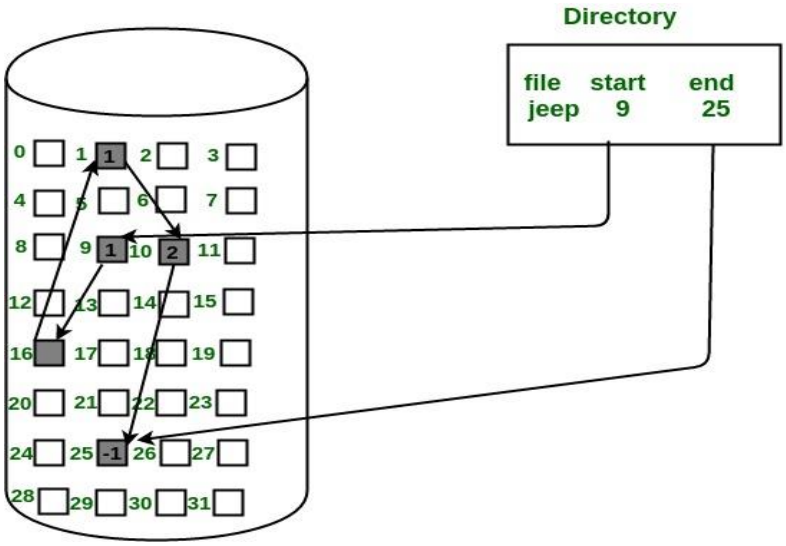
1. **Contiguous allocation**:

   In this method, the operating system stores the files on the storage device as a contiguous block of disk space. This means that all the sectors of the file are stored together, without any gaps in between. Contiguous allocation is simple and efficient, but it can lead to fragmentation, which occurs when the free space on the disk is scattered in small chunks rather than being contiguous.



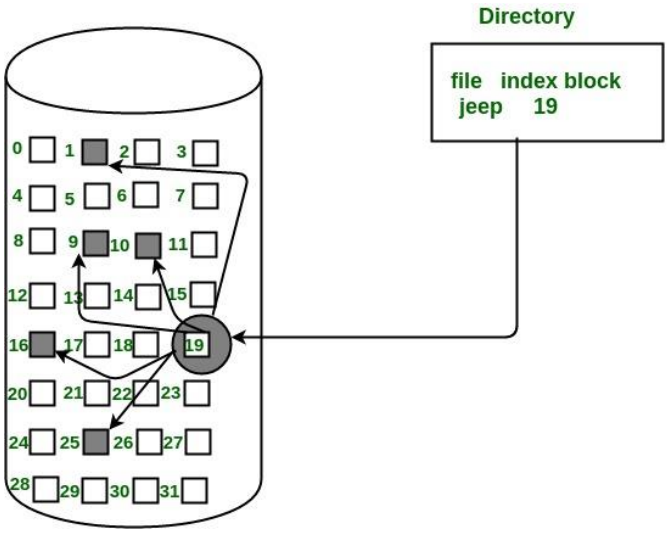
2. **Linked allocation:**

   In this method, the operating system stores each file as a series of blocks (or "clusters") that are linked together. Each block has a pointer to the next block in the file, so the operating system can follow the chain of pointers to locate the blocks and read or write the file. Linked allocation is more flexible than contiguous allocation, because it allows files to be stored in non-contiguous blocks on the disk.



3. **Indexed allocation:**

   In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block as shown in the image:



Explain About File Access Methods.

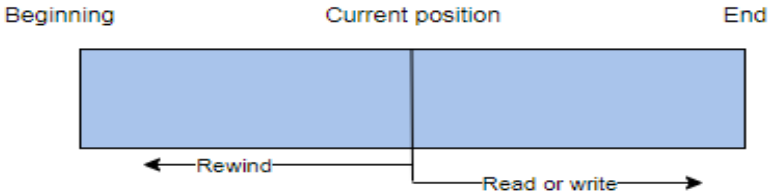
Different methods of file access in the operating system are given below:

- Sequential file access
- Direct file access
- Index file access

Sequential file access:

This is the most common file access method. The information in the file is processed in order, one record after the other. All the information needs to be loaded sequentially for random access inside the given file. Usually, it contains read, write and rewind file operations.

The read operation (read next) reads the data stored on the next position of the pointer and moves the pointer forward by one read point. The write operation (write next) adds the data at the end of the file and moves the pointer forward to the newly written data. The rewind operation moves the pointer backward until the file's required information appears. An example of this file access method can be accessing data from a tape.

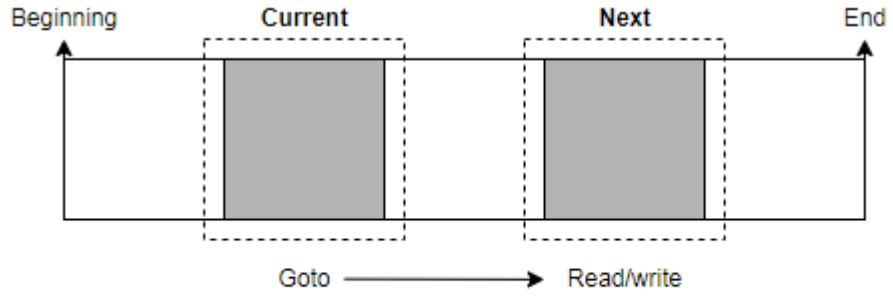


Sequential file access

Direct file access:

This method is also known as the relative access method. A file is composed of fixed-length logical records called blocks. These blocks can be accessed in any order. The file is accessed as a numbered sequence of blocks or records. Therefore, we may read block 10, then read block 40, and then write block 5.

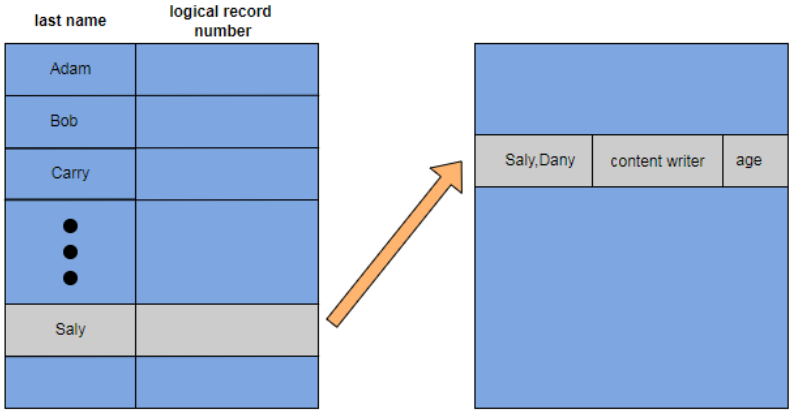
There are no constraints on the order of reading or writing for a direct-access file. Direct access files are of great use for intermediate access to large amounts of information, and most databases are based on this type of file access.



Direct file access

Index file access:

A direct-access method serves as the foundation for this file access technique. An index is created for the entire file, which contains pointers to the various blocks inside the disk. To find a record in the file, we first traverse through the index and then use the pointer to access the file directly. However, in this file access, extra memory is required for keeping the index file, which burdens the memory resources. The following shows the index file and how it works:



SHORT ANSWERS

Describe The File Sharing.

File sharing is the public or private sharing of computer data or space in a network with various levels of access privilege. While files can easily be shared outside a network the term file sharing almost always means sharing files in a network, even if in a small local area network.

File sharing allows a number of people to use the same file or file by some combination of being able to read or view it, write to or modify it, copy it, or print it. Typically, a file sharing system has one or more administrators. Users may all have the same or may have different levels of access privilege.

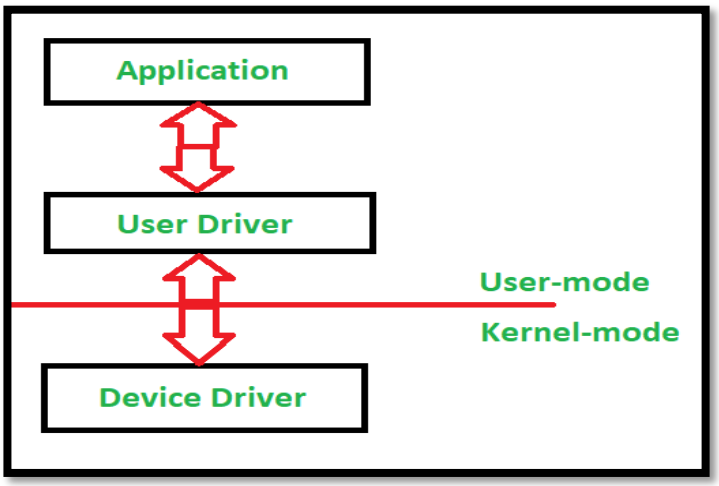
File sharing has been a feature of mainframe and multi-user computer systems for many years. With the advent of the Internet, a file transfer system called the File Transfer Protocol (FTP) has become widely-used. FTP can be used to access (read and possibly write to) files shared among a particular set of users with a password to gain access to files shared from an FTP server site.

Any multi-user operating system will provide some form of file sharing. Files can also be shared in file systems distributed over different points in a network. File sharing is involved in groupware and a number of other types of applications.

Write About Device Drivers.

Operating System takes help from device drivers to handle all I/O devices. A device driver is a computer program that operates or controls a particular device attached to a computer or automaton. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without knowing precise details about the hardware being used.

More commonly known as a driver, a device driver or hardware driver is a group of files that enable one or more hardware devices to communicate with the computer's operating system. Without drivers, the computer could not send and receive data correctly to hardware devices like printers.



Distinguish between Authentication and Authorization.

Authentication	Authorization
Authentication is the process of identifying a user to provide access to a system.	Authorization is the process of giving permission to access the resources.
In this, the user or client and server are verified.	In this, it is verified that if the user is allowed through the defined policies and rules.
It is usually performed before the authorization.	It is usually done once the user is successfully authenticated.

It requires the login details of the user, such as user name & password, etc.	It requires the user's privilege or security level.
Data is provided through the Token Ids.	Data is provided through the access tokens.
Example: Entering Login details is necessary for the employees to authenticate themselves to access the organizational emails or software.	**Example:** After employees successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles.
Authentication credentials can be partially changed by the user as per the requirement.	Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it

Write About File Attributes.

Attributes of the File:

- **Name:** Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.
- **Identifier:** Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension .txt, A video file can have the extension .mp4.
- **Type:** In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.
- **Location:** In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.
- **Size:** The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.
- **Protection:** The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of Users.
- **Time and Date:** Every file carries a time stamp which contains the time and date on which the file is last modified.

****