CMR ENGINEERING COLLEGE

KANDLAKOYA (V), MEDCHAL (M), HYDERABAD

STEP MATERIAL ON OPERATING SYSTEMS

DEPARTMENT OF CSE (AI&ML)

UGC AUTONOMOUS

Subject: Operating Systems

Year: II Year/I Sem(R20)

Academic Year: 2021-2022

INSTITUTE VISION AND MISSION

VISION

To be recognized as a premier institution in offering value based and futuristic quality technical education to meet the technological needs of the society

MISSION

- To impart value based quality technical education through innovative teaching and learning methods
- To continuously produce employable technical graduates with advanced technical skills to meet the current and future technological needs of the society
- To prepare the graduates for higher learning with emphasis on academic and industrial research

DEPARTMENT VISION AND MISSION

VISION

To produce globally competent and industry ready graduates in Computer Science & Engineering by imparting quality education with a know-how of cutting edge technology and holistic personality

MISSION

To offer high quality education in Computer Science & Engineering in order to build core competence for the graduates by laying solid foundation in Applied Mathematics, and program framework with a focus on concept building

The department promotes excellence in teaching, research, and collaborative activities to prepare graduates for professional career or higher studies

Creating intellectual environment for developing logical skills and problem solving strategies, thus to develop, able and proficient computer engineer to compete in the current global scenario

PART –A
UNIT WISE SHORT QUESTION AND ANSWERS
UNIT-I

1. What is an Operating System?

Ans: An operating system is a program that manages the computer hardware. It also provides a basis for application programs and act as an intermediary between a user of a computer and the computer hardware. It controls and coordinates the use of the hardware among the various application programs for the various users.

Some of the popular Operating Systems are DOS, Windows, Ubuntu, Solaris etc.

2. What are the main functions of OS?

Ans:The main functions of an OS are:

a. Process Management
b. Memory Management
c. Input/ Output Management
d. Storage/ File system management

3.What is the main purpose of an operating system?

Ans: There are two main purposes of an operating system:

1.It is designed to make sure that a computer system performs well by managing its computational activities.

2.It provides an environment for the development and execution of programs.

4.What is a Kernel?

Ans: Kernel is the part of OS which handles all details of sharing resources and device handling.

1. It can be considered as the core of OS which manages the core features of an OS.

2.Its purpose is to handle the communication between software and hardware

3.Its services are used through system calls.

5.What are the main functions of a Kernel?

Ans:The main functions of a Kernel are:

1.Process management
2.Device management
3.Memory management
4.Interrupt handling

5.I/O communication
6.File system management

6.What are Real-time systems?

Ans:Real-time systems are used when rigid time requirements have been placed on the operation of a processor. It has well defined and fixed time constraints.

7. Define Throughput ?
Ans: Number of processes that complete their execution per time unit.

8.**What are Batch Systems?**
Ans: Batch systems are quite appropriate for executing large jobs that need little interaction. The user can submit jobs and return later for the results. It is not necessary to wait while the job is processed. Operators batched together jobs with similar needs and ran them through the computer as a group.

9.What are the different operating systems?
Ans: 1.Batched operating systems
2.Distributed operating systems
3.Time sharing operating systems
4.Multi-programmed operating systems
5.Real-time operating systems

10.Describe the objective of Multi-programming.

Ans: The main objective of Multi-programming is to have a process running at all times. With this design, CPU utilization is said to be maximized.

11.What is Time- sharing system?

Ans: In a Time-sharing system, the CPU executes multiple jobs by switching among them, also known as multitasking. This process happens so fast that users can interact with each program while it is running.

12.What are System Calls?
Ans: System calls provide the interface between a process and the Operating system. System Calls are also called as Monitor call or Operating-system function call. When a system call is executed, it is treated as by the hardware as software interrupt. Control passes through the interrupt vector to a service routine in the operating system, and the mode bit is set to monitor mode.

13. List the services provided by an Operating System?
Ans: 1.Program execution
2.I/O Operation
3.File System manipulation
4.Communication
5.Error detection

14. What is the difference between Hard Real Time System and Soft Real Time System?

Ans: A hard real time system guarantees that critical tasks complete on time. In a soft real time system, a critical real-time task gets priority over the other tasks, and retains that priority until it completes. Soft real time systems have more limited utility than do hard real-time systems.

15. What are the design goals of an Operating System?

Ans: The requirements can be divided into two basic groups: User goals and System goals.Users desire that the system should be convenient and easy to use, easy to learn, reliable, safe and fast. The Operating system should be easy to design, implement, and maintain. Also it should be flexible, reliable, error free and efficient. These are some of the requirements, which are vague and have no general solution.

16. What are the five major categories of System Calls?

Ans : 1.Process Control
2.FIle management
3.Device management
4.Information maintenance
5.Communications

17.Define Elapsed CPU time and Maximum CPU time?

Ans: Elapsed CPU Time: Total CPU time used by a process to date.

Maximum CPU Time: Maximum amount of CPU time a process may use.

UNIT-2

1.Define Process.

Ans: A process is a program that is running and under execution. On batch systems, it is called as a "job" while on time sharing systems, it is called as a "task".

2.What are the different states of a process?

Ans: A list of different states of process:

1.New Process
2.Running Process
3.Waiting Process
4.Ready Process
5.Terminated Process

3.Explain the basic functions of process management.

Ans: Important functions of process management are:

1.Creation and deletion of system processes.
2.Creation and deletion of users.
3.CPU scheduling.
4.Process communication and synchronization.

4.Define Arrival Time , Completion Time,Turn Around Time,Burst Time?

Ans: Arrival Time: Time at which the process arrives in the ready queue.

Completion Time: Time at which process completes its execution.

Turn Around Time: Time Difference between completion time and arrival time.
Turn Around Time = Completion Time – Arrival Time

Burst Time: Time required by a process for CPU execution.

5.Define Waiting Time(W.T):

Ans: Time Difference between turnaround time and burst time.
Waiting Time = Turn Around Time – Burst Time

6.What is a Thread?

Ans:A Thread is a light weight process.It is a basic unit of CPU utilization. In general, a thread is composed of a thread ID, program counter, register set, and the stack.

7..What is the basic difference between pre-emptive and non-pre-emptive scheduling.

Ans: Pre-emptive scheduling allows interruption of a process while it is executing and taking the CPU to another process while non-pre-emptive scheduling ensures that a process keeps the CPU under control until it has completed execution.

8..What is context switching?

Ans: Context is associated with each process encompassing all the information describing the current execution state of the process.
1.When the OS saves the context of program that is currently running and restores the context of the next ready to run process, it is called as context switching.
2.It is important for multitasking OS.

9..Explain about SJF algorithm.

Ans: Process which have the shortest burst time are scheduled first.If two processes have the same bust time then FCFS is used to break the tie. It is a non-preemptive scheduling algorithm.

10.What are the benefits of multi-threaded programming?
Ans: It makes the system more responsive and enables resource sharing. It leads to the use of multi-process architecture. It is more economical and preferred.

11.What are the different scheduling algorithms
Ans : First-Come, First-Served (FCFS) Scheduling.
Shortest-Job-First (SJF) Scheduling.
Priority Scheduling.
Round Robin(RR) Scheduling.

12. What is RR scheduling algorithm?

Ans: RR (round-robin) scheduling algorithm is primarily aimed for time-sharing systems. A circular queue is a setup in such a way that the CPU scheduler goes around that queue, allocating CPU to each process for a time interval of up to around 10 to 100 milliseconds.

13..What is Process Control Block (PCB)?
Ans : Each process is represented in the operating system by a process control block also called a task control block. It contains many pieces of information associated with a specific process. It simply acts as a repository for any information that may vary from process to process. It contains the following information:

1. Process state
2. Program counters
3. CPU registers
4. CPU scheduling information
5. Memory management information
6. Accounting information
7. I/O status information

14. What is the use of fork and exec system calls?

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

15. What is a Dispatcher?

Ans : The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:

1.Switching context

2.Switching to user mode

3.Jumping to the proper location in the user program to restart that program.

UNIT-3

1. What is deadlock?

Ans: Deadlock is a specific situation or condition where two processes are waiting for each other to complete so that they can start. But this situation causes hang for both of them.

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. ... A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other.

2. What are the four necessary and sufficient conditions behind the deadlock?

Ans: These are the 4 conditions:

1) **Mutual Exclusion Condition**: It specifies that the resources involved are non-sharable.

2) **Hold and Wait Condition**: It specifies that there must be a process that is holding a resource already allocated to it while waiting for additional resource that are currently being held by other processes.

3) **No-Preemptive Condition**: Resources cannot be taken away while they are being used by processes.

4) **Circular Wait Condition**: It is an explanation of the second condition. It specifies that the processes in the system form a circular list or a chain where each process in the chain is waiting for a resource held by next process in the chain.

3. What do you know about a Pipe? When is it used?

Ans: It is an IPC mechanism used for one way communication between two processes which are related.
A single process doesn't need to use pipe. It is used when two process wish to communicate one-way.

4. What is a named pipe?

Ans: A traditional pipe is unnamed and can be used only for the communication of related process. If unrelated processes are required to communicate - named pipes are required.
- It is a pipe whose access point is a file available on the file system. When this file is opened for reading, a process is granted access to the reading end of the pipe. Similarly, when the file is opened for writing, the process is granted access to writing end of the pipe.

5. What are the various IPC mechanisms?

Ans : Various IPC mechanisms are:

a. Sockets
b. Pipes
c. Shared memory
d. Signals
e. Message Queues

6. What is a semaphore?

Ans: A semaphore is hardware or a software tag variable whose value indicates the status of a common resource.
- Its purpose is to lock the common resource being used. A process which needs the resource will check the semaphore to determine the status of the resource followed by the decision for proceeding.
- In multitasking operating systems, the activities are synchronized by using the semaphore techniques

7. What kind of operations are possible on a semaphore?

Ans: There are Two kind of operations are possible on a semaphore - 'wait' and 'signal'.

8. Explain about Mutex.

Ans: Mutex - 'Mutual Exclusion Lock' is a lock which protects access to shared data resource. Threads can create and initialize a mutex to be used later.
Before entering a critical region the mutex is locked. It is unlocked after exiting the critical region. If any thread tries to lock the mutex during this time, it can't do so.

9. What is a critical section?

Ans: It is a section of code which can be executed only by one process at a time.

10. What is synchronization? What are the different synchronization mechanisms?

Ans: Synchronization means controlling access to a resource that is available to two or more threads or process. Different synchronization mechanisms are:

1.Mutex
2.Semaphores
3.Monitors
4.Condition variables
5.Critical regions
6.Read/ Write locks

11. What is Banker's algorithm?

Ans: Banker's algorithm is used to avoid deadlock. It is the one of deadlock-avoidance method. It is named as Banker's algorithm on the banking system where bank never allocates available cash in such a manner that it can no longer satisfy the requirements of all of its customers.

12. What is semaphore?

Ans: Semaphore is a protected variable or abstract data type that is used to lock the resource being used. The value of the semaphore indicates the status of a common resource.

There are two types of semaphore:

1. Binary semaphores

2. Counting semaphores

13. What is a binary Semaphore?

Ans: Binary semaphore takes only 0 and 1 as value and used to implement mutual exclusion and synchronize concurrent processes.

14. When is a system in safe state?
Ans: The set of dispatch-able processes is in a safe state if there exists at least one Temporal order in which all processes can be run to completion without resulting in a deadlock.

UNIT-4

1. What is a virtual memory?

Ans: Virtual memory is a memory management technique for letting processes execute outside of memory. This is very useful especially is an executing program cannot fit in the physical memory.

2. State the main difference between logical from physical address space.

Ans: Logical address refers to the address that is generated by the CPU. On the other hand, physical address refers to the address that is seen by the memory unit.

3. What is fragmentation?

Ans: Fragmentation is memory wasted. It can be internal if we are dealing with systems that have fixed-sized allocation units, or external if we are dealing with systems that have variable-sized allocation units.

4. What is a socket?

Ans: A socket provides a connection between two applications. Each endpoint of a communication is a socket.

5. What is the main function of the memory-management unit?
Ans: The runtime mapping from virtual to physical addresses is done by a hardware device called a memory management unit (MMU).

6. Define swapping.

Ans: 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.

7. Define secondary memory.

Ans: This memory holds those pages that are not present in main memory.

The secondary memory is usually a high speed disk. It is known as the swap device, and the section of the disk used for this purpose is known as swap space.

8. What is the basic approach of page replacement?

Ans: 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.

9. What are the various page replacement algorithms used for page replacement?

Ans: 1.FIFO page replacement

2.Optimal page replacement

3.LRU page replacement

4. LRU approximation page replacement

5.Counting based page replacement

6. Page buffering algorithm.

10. What are the major problems to implement demand paging?

Ans: The two major problems to implement demand paging is developing

a. Frame allocation algorithm

b. Page replacement algorithm

11. What Is Demand Paging?

Ans: Demand paging is referred when not all of a process's pages are in the RAM, then the OS brings the missing(and required) pages from the disk into the RAM.

12. What is caching?

Ans: Caching is the processing of utilizing a region of fast memory for a limited data and process. A cache memory is usually much efficient because of its high access speed.

UNIT-5

1. What is a file?

Ans: 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.

1. File attributes: Name, identifier, type, size, location, protection, time, date
2.File operations: creation, reading, writing, repositioning, deleting, truncating, appending, renaming
3. File types: executable, object, library, source code etc.

2. List the various file attributes.

Ans: 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, date and user identification.

3. What are the various file operations?

Ans: The six basic file operations are

1.Creating a file
2.Writing a file
3.Reading a file
4.Repositioning within a file
5.Deleting a file
6.Truncating a file

4. What are the information associated with an open file?

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

a) File pointer
b) File open count
c) Disk location of the file
d) Access rights

5. What are the different accessing methods of a file?

Ans: The different types of accessing a file are:

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

6. What is Directory?

Ans : 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.

a) Search for a file
b) Create a file
c) Delete a file
d) Rename a file
e) List directory
f) Traverse the file system

7. What are the allocation methods of a disk space?

Ans: Three major methods of allocating disk space which are widely in use are

a. Contiguous allocation

b. Linked allocation
c. Indexed allocation

8. when designing the file structure for an operating system, what attributes are considered?

Ans: Typically, the different attributes for a file structure are naming, identifier, supported file types, and location for the files, size, and level of protection.

9. What is root partition?

Ans: Root partition is where the operating system kernel is located. It also contains other potentially important system files that are mounted during boot time.

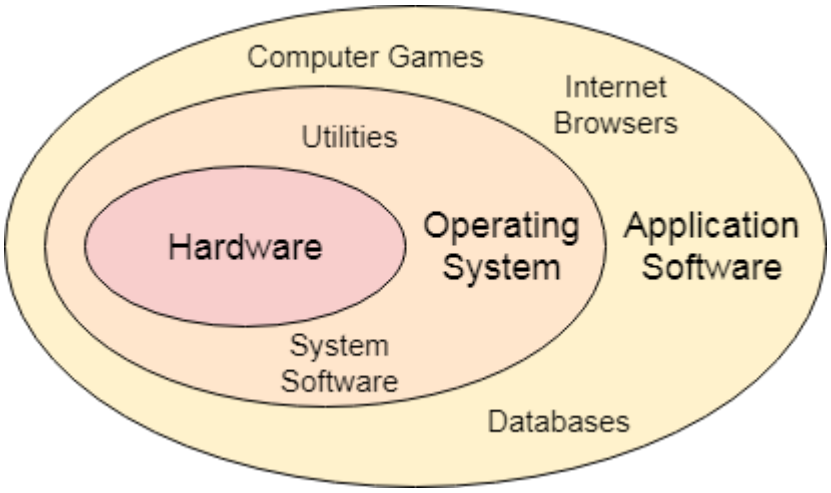
10. What is the purpose of an I/O status information?

Ans: I/O status information provides information about which I/O devices are to be allocated for a particular process. It also shows which files is opened, and other I/O device state.

PART –B
UNIT WISE ESSAY QUESTION AND ANSWERS
UNIT-I

1. **Explain the structure and functions of operating systems**.

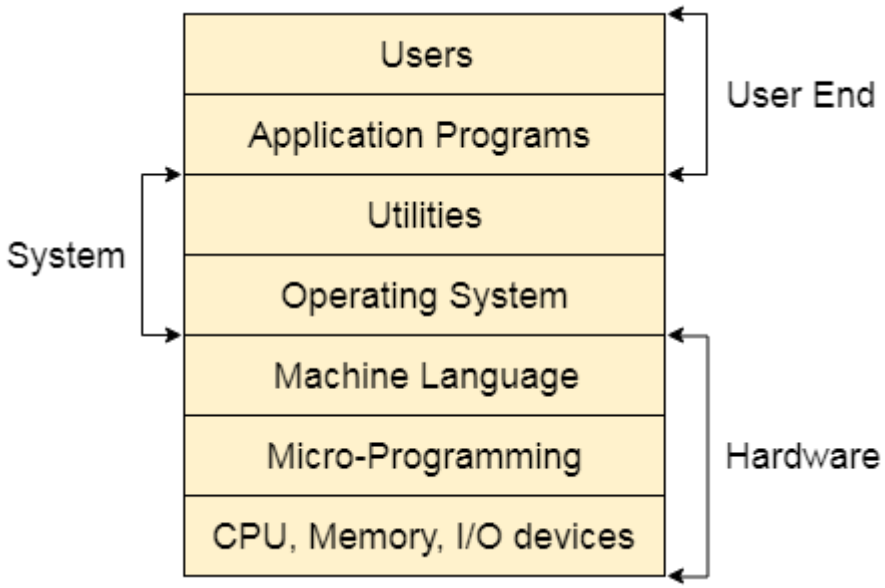
An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.



Structure of a Computer System

A Computer System consists of:

- Users (people who are using the computer)
- Application Programs (Compilers, Databases, Games, Video player, Browsers, etc.)
- System Programs (Shells, Editors, Compilers, etc.)
- Operating System (A special program which acts as an interface between user and hardware)
- Hardware (CPU, Disks, Memory, etc)



What does an Operating system do?

1. Process Management
2. Process Synchronization
3. Memory Management
4. CPU Scheduling
5. File Management
6. Security.

In an operating system software performs each of the function:

1. **Process management**:- Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

2. **Memory management:-** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.

3. **File management**:- It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

4. **Device Management**: Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.

5. **I/O System Management:** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

6. **Secondary-Storage Management**: Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.

7. **Security**:- Security module protects the data and information of a computer system against malware threat and authorized access.

8. **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

9. **Networking:** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.

10. **Job accounting**: Keeping track of time & resource used by various job and users.

11. **Communication management**: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

Features of Operating System (OS)

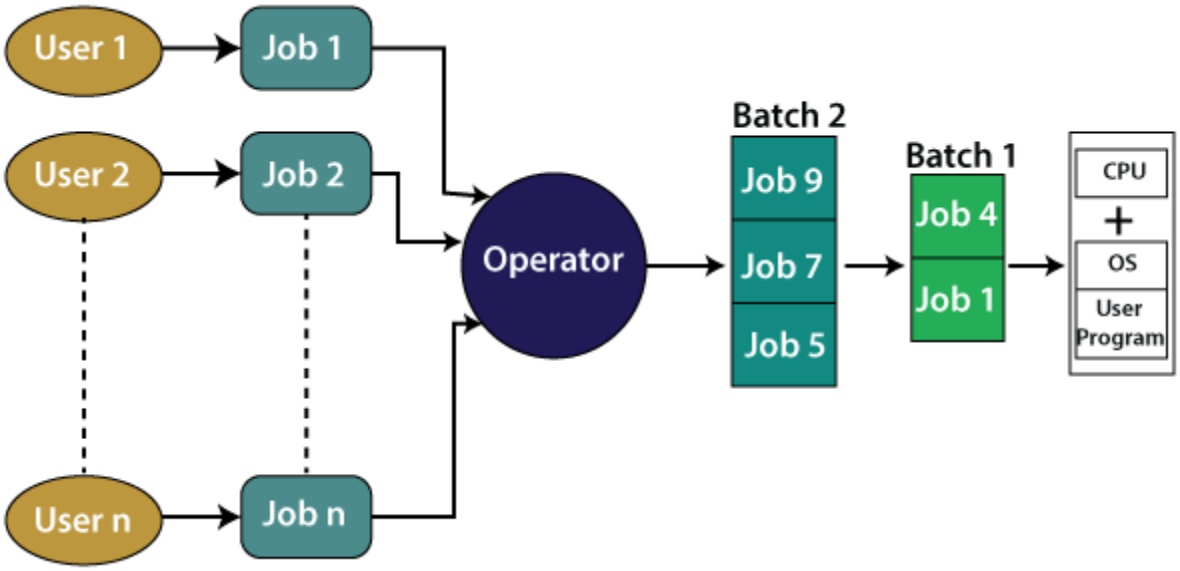
Here is a list important features of OS:

- Protected and supervisor mode
- Allows disk access and file systems Device drivers Networking Security
- Program Execution
- Memory management Virtual Memory Multitasking
- Handling I/O operations
- Manipulation of the file system

- Error Detection and handling
- Resource allocation
- Information and Resource Protection

2. Explain Batch and Multi-programmed operating systems.

Batch Processing Operating System:



The interaction between a user and the computer does not occur in this system. The user is required to prepare jobs on punch cards in the form of batches and submit them to the computer operator. The computer operator sorts the jobs or programs and keeps similar programs or jobs in the same batch and run as a group to speed up processing. It is designed to execute one job at a time. Jobs are processed on a first-come, first-serve basis, i.e., in the order of their submission without any human intervention.

For example, the credit card bill generated by banks is an example of batch processing. A separate bill is not generated for each credit card purchase, rather a single bill that includes all purchases in a month is generated through batch processing. The bill details are collected and held as a batch, and then it is processed to generate the bill at the end of the billing cycle. Similarly, in a payroll system, the salaries of employees of the company are calculated and generated through the batch processing system at the end of each month.

Advantages of Batch processing operating system:

- Repeated jobs can be completed easily without any human intervention

- Hardware or system support is not required to input data in batch systems
- It can work offline, so it causes less stress on the processor as it knows which task to process next and how long the task will last.
- It can be shared among multiple users.
- You can set the timing of batch jobs so that when the computer is not busy, it can start processing the batch jobs such as at night or any other free time.

Disadvantages of batch processing operating systems:

- You need to train the computer operators for using the batch system.
- It is not easy to debug this system.
- If any error occurs in one job, the other jobs may have to wait for an uncertain time.

Multi-programmed OS:

Definition: In the multi-programming system, one or multiple programs can be loaded into its **main memory** for getting to execute. It is capable only one program or process to get **CPU** for executes for their instructions, and other programs wait for getting their turn. Main goal of using of multiprogramming system is overcome issue of under utilization of CPU and **primary memory.**

Multi-programming operating system is designed based on this principle that sub segmenting parts of transient area to store individual programs, and then resource management routines are attached with basic function of operating system**.**

In the multiprogramming system, multiple users can perform their tasks concurrently, and it can be stored into main memory. CPU has ability to deliver time to several programs while sitting in idle mode, when one is getting engage along with I/O operations.

An OS does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.

- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle, unless there are no jobs to process.

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

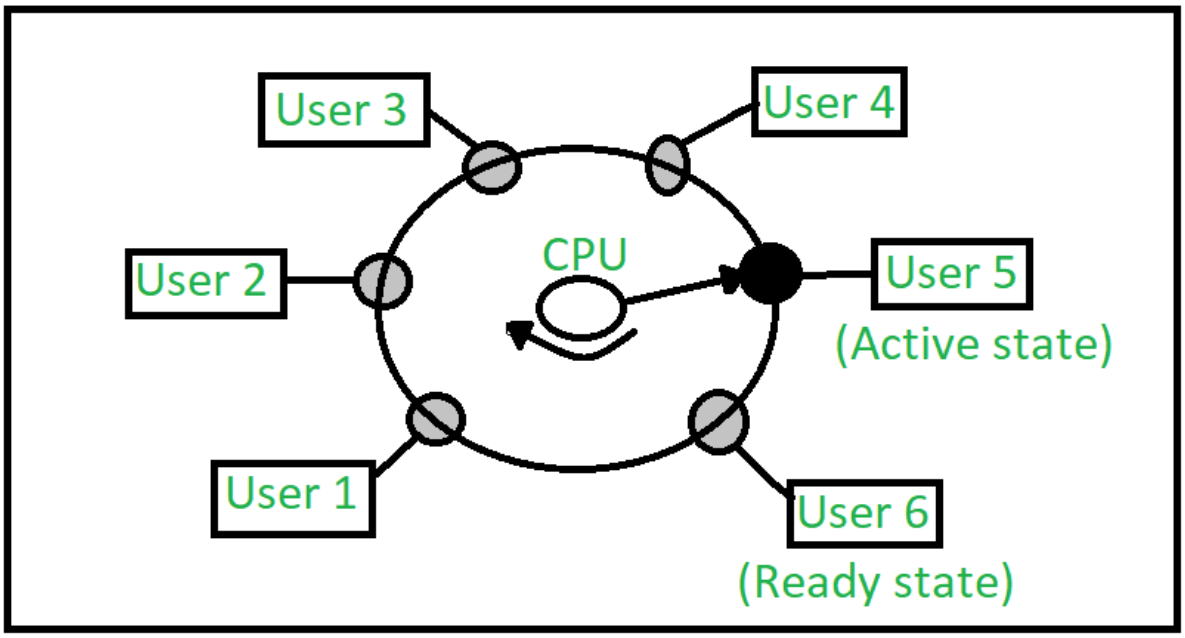
- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

3. Explain timeshared operating systems.

Ans : Time –Shared operating systems:

A time shared operating system allows multiple users to share computers simultaneously. Each action or order at a time the shared system becomes smaller, so only a little CPU time is required for each user. As the system rapidly switches from one user to another, each user is given the impression that the entire computer system is dedicated to its use, although it is being shared among multiple users.

A **time shared operating system** uses CPU scheduling and multi-programming to provide each with a small portion of a shared computer at once. Each user has at least one separate program in memory. A program loaded into memory and executes, it performs a short period of time either before completion or to complete I/O.This short period of time during which user gets attention of CPU is known as **time slice, time slot or quantum**. It is typically of the order of 10 to 100 milliseconds. Time shared operating systems are more complex than multiprogrammed operating systems. In both, multiple jobs must be kept in memory simultaneously, so the system must have memory management and security. To achieve a good response time, jobs may have to swap in and out of disk from main memory which now serves as a backing store for main memory. A common method to achieve this goal is virtual memory, a technique that allows the execution of a job that may not be completely in memory.



In above figure the user 5 is **active state** but user 1, user 2, user 3, and user 4 are in **waiting state** whereas user 6 is in **ready state.**

1. **Active State –**
   The user's program is under the control of CPU. Only one program is available in this state.
2. **Ready State –**
   The user program is ready to execute but it is waiting for for it's turn to get the CPU.More than one user can be in ready state at a time.
3. **Waiting State –**
   The user's program is waiting for some input/output operation. More than one user can be in a waiting state at a time.

Requirements of Time Sharing Operating System :

An alarm clock mechanism to send an interrupt signal to the CPU after every time slice. Memory Protection mechanism to prevent one job's instructions and data from interfering with other jobs.

Advantages :

1. Each task gets an equal opportunity.
2. Less chances of duplication of software.
3. CPU idle time can be reduced.

Disadvantages :

1. Reliability problem.
2. One must have to take of security and integrity of user programs and data.
3. Data communication problem.

4. Discuss about Distributed and Real-time operating systems.

Ans: Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

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.

The advantages of distributed systems are as follows −

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Real-time operating systems:

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the **response time**. So in this method, the response time is very less as compared to online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will

fail. For example, scientific experiments, medical image systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects likes undersea exploration and planetary rovers, etc.

5. Discuss about 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 the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

Program execution

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management −

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

I/O Operation

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

File system manipulation

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Communication

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error handling

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management : In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Protection

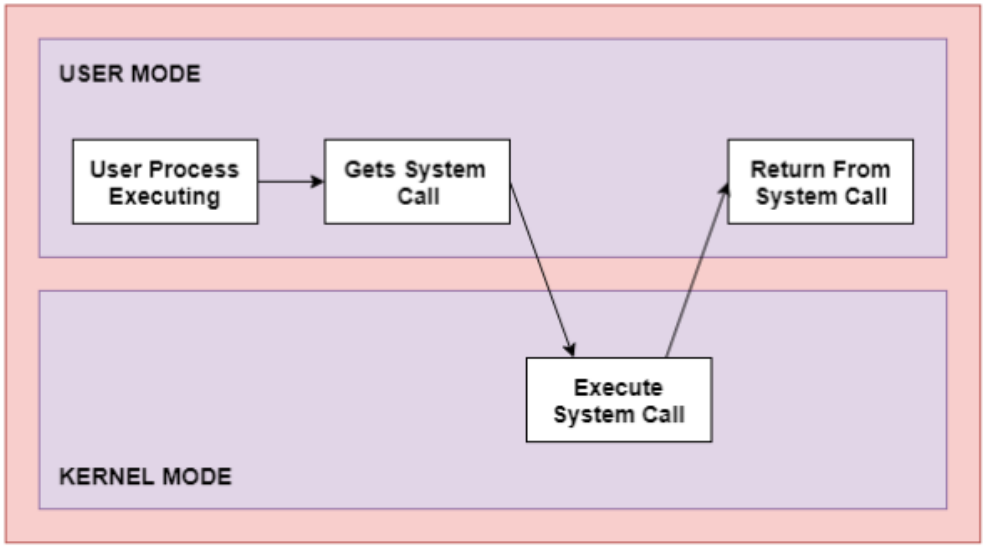
Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection −

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

6. Explain various system calls and its importance.

The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions. They are also included in the manuals used by the assembly level programmers. System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.



As can be seen from this diagram, the processes execute normally in the user mode until a system call interrupts this. Then the system call is executed on a priority basis in the kernel mode. After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed.

In general, system calls are required in the following situations −

- If a file system requires the creation or deletion of files. Reading and writing from files also require a system call.
- Creation and management of new processes.
- Network connections also require system calls. This includes sending and receiving packets.
- Access to a hardware devices such as a printer, scanner etc. requires 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()

There are many different system calls as shown above. Details of some of those system calls are as follows −

open()

The open() system call is used to provide access to a file in a file system. This system call allocates resources to the file and provides a handle that the process uses to refer to the file. A file can be opened by multiple processes at the same time or be restricted to one process. It all depends on the file organisation and file system.

read()

The read() system call is used to access data from a file that is stored in the file system. The file to read can be identified by its file descriptor and it should be opened using open() before it can be read. In general, the read() system calls takes three arguments i.e. the file descriptor, buffer which stores read data and number of bytes to be read from the file.

write()

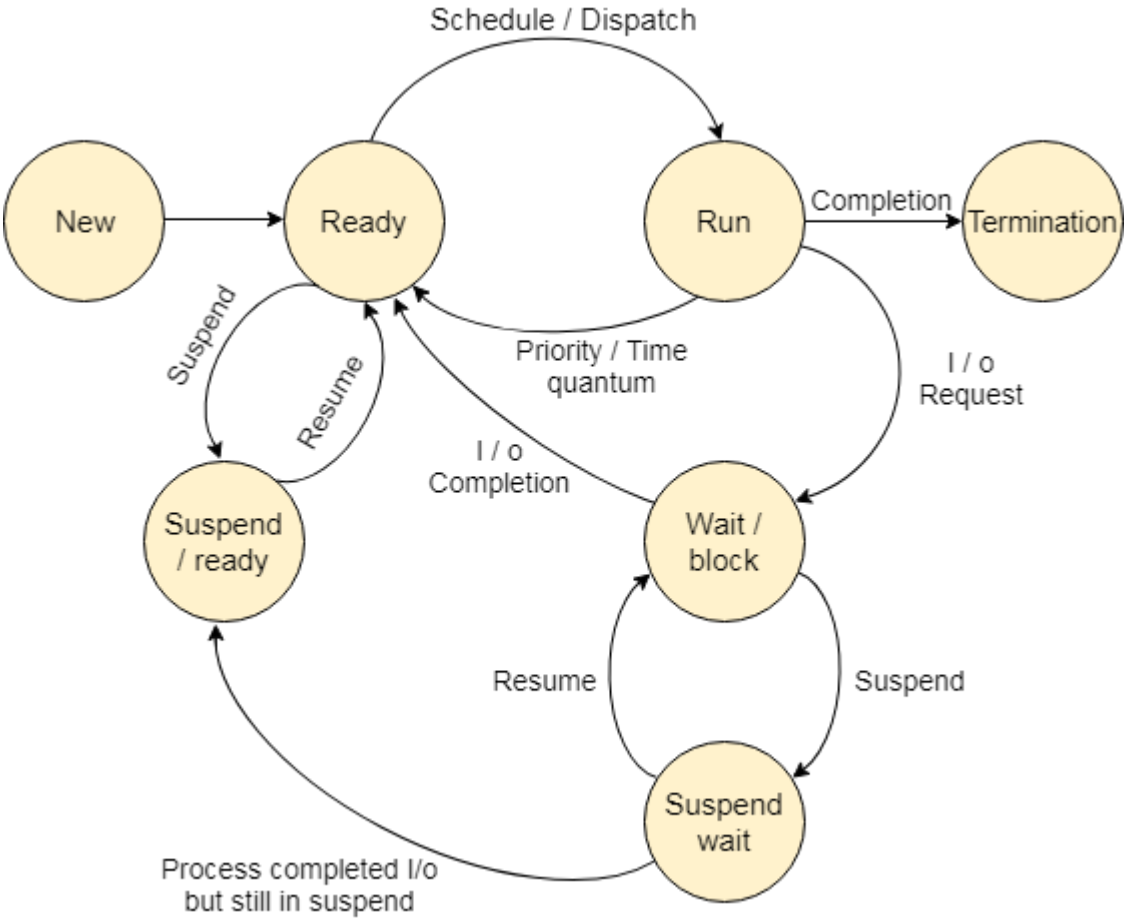
The write() system calls writes the data from a user buffer into a device such as a file. This system call is one of the ways to output data from a program. In general, the write system calls takes three arguments i.e. file descriptor, pointer to the buffer where data is stored and number of bytes to write from the buffer.

close()

The close() system call is used to terminate access to a file system. Using this system call means that the file is no longer required by the program and so the buffers are flushed, the file metadata is updated and the file resources are de-allocated.

UNIT-II

1.Explain process states and various operations on process.



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.

1. New

A program which is going to be picked up by the OS into the main memory is called a new process.

2. 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.

3. Running

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.

4. Block or wait

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.

5. Completion or termination

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.

6. Suspend ready

A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.

If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority process out into the secondary memory. The suspend ready processes remain in the secondary memory until the main memory gets available.

7. Suspend wait : Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory. Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and make room for

the higher priority process. These processes complete their execution once the main memory gets available and their wait is finished.

Operations on the Process

1. Creation

Once the process is created, it will be ready and come into the ready queue (main memory) and will be ready for the execution.

2. Scheduling

Out of the many processes present in the ready queue, the Operating system chooses one process and start executing it. Selecting the process which is to be executed next, is known as scheduling.

3. Execution

Once the process is scheduled for the execution, the processor starts executing it. Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.

4. Deletion/killing

Once the purpose of the process gets over then the OS will kill the process. The Context of the process (PCB) will be deleted and the process gets terminated by the Operating system.

2. Discuss about different types of scheduler.

Process Scheduling handles the selection of a process for the processor on the basis of a scheduling algorithm and also the removal of a process from the processor. It is an important part of multiprogramming operating system.

There are many scheduling queues that are used in process scheduling. When the processes enter the system, they are put into the job queue. The processes that are ready to execute in the main memory are kept in the ready queue. The processes that are waiting for the I/O device are kept in the I/O device queue.

The different schedulers that are used for process scheduling are −

Long Term Scheduler

The job scheduler or long-term scheduler selects processes from the storage pool in the secondary memory and loads them into the ready queue in the main memory for execution.

The long-term scheduler controls the degree of multiprogramming. It must select a careful mixture of I/O bound and CPU bound processes to yield optimum system throughput. If it selects too many CPU bound processes then the I/O devices are idle and if it selects too many I/O bound processes then the processor has nothing to do.

The job of the long-term scheduler is very important and directly affects the system for a long time.

Short Term Scheduler

The short-term scheduler selects one of the processes from the ready queue and schedules them for execution. A scheduling algorithm is used to decide which process will be scheduled for execution next.

The short-term scheduler executes much more frequently than the long-term scheduler as a process may execute only for a few milliseconds.

The choices of the short term scheduler are very important. If it selects a process with a long burst time, then all the processes after that will have to wait for a long time in the ready queue. This is known as starvation and it may happen if a wrong decision is made by the short-term scheduler.

A diagram that demonstrates long-term and short-term schedulers is given as follows −

Medium Term Scheduler

The medium-term scheduler swaps out a process from main memory. It can again swap in the process later from the point it stopped executing. This can also be called as suspending and resuming the process.

This is helpful in reducing the degree of multiprogramming. Swapping is also useful to improve the mix of I/O bound and CPU bound processes in the memory.

3. Discuss about threads and types of threads.

There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Thread is often referred to as a lightweight process.

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 are 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 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. Example: Window Solaris.

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.

Components of Threads

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

Benefits of Threads

1. **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.

2. **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
3. **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
4. **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
5. **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
6. **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There are a stack and register for each thread.

User Level threads	**Kernel Level Threads**
These threads are implemented by users.	These threads are implemented by Operating systems
These threads are not recognized by operating systems,	These threads are recognized by operating systems,
In User Level threads, the Context switch requires no hardware support.	In Kernel Level threads, hardware support is needed.
These threads are mainly designed as dependent threads.	These threads are mainly designed as independent threads.
In User Level threads, if one user-level thread performs a blocking operation then the entire process will be blocked.	On the other hand, if one kernel thread performs a blocking operation then another thread can continue the execution.
Example of User Level threads: Java thread, POSIX threads.	Example of Kernel level threads: Window Solaris.
Implementation of User Level thread is done by a thread library and is easy.	While the Implementation of the kernel-level thread is done by the operating system and is complex.
This thread can run on any OS	This is specific to the operating system.

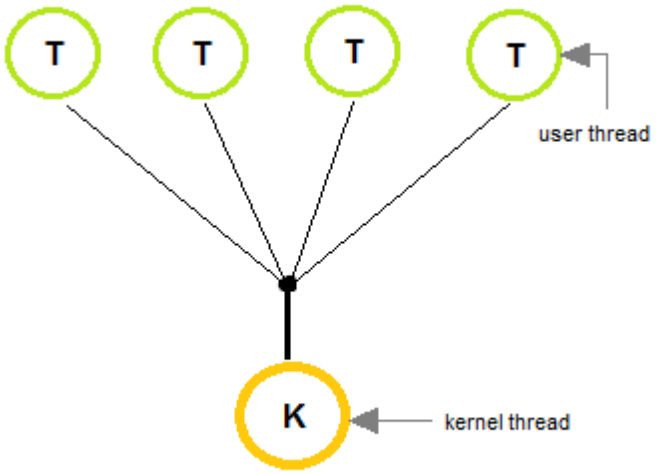
Multithreading Models

The user threads must be mapped to kernel threads, by one of the following strategies:

- Many to One Model
- One to One Model
- Many to Many Model

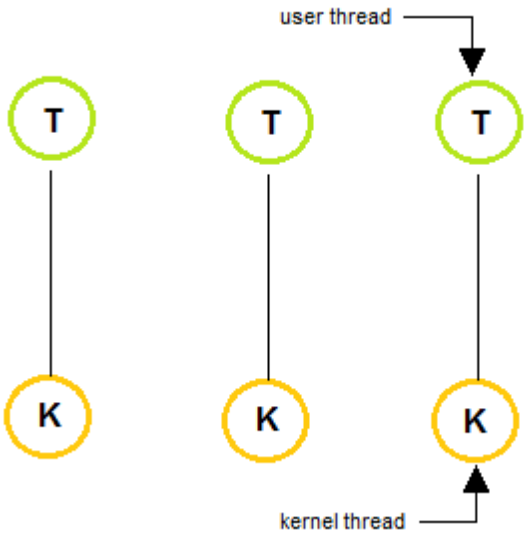
Many to One Model

- In the **many to one** model, many user-level threads are all mapped onto a single kernel thread.
- Thread management is handled by the thread library in user space, which is efficient in nature.
- In this case, if user-level thread libraries are implemented in the operating system in some way that the system does not support them, then the Kernel threads use this many-to-one relationship model.



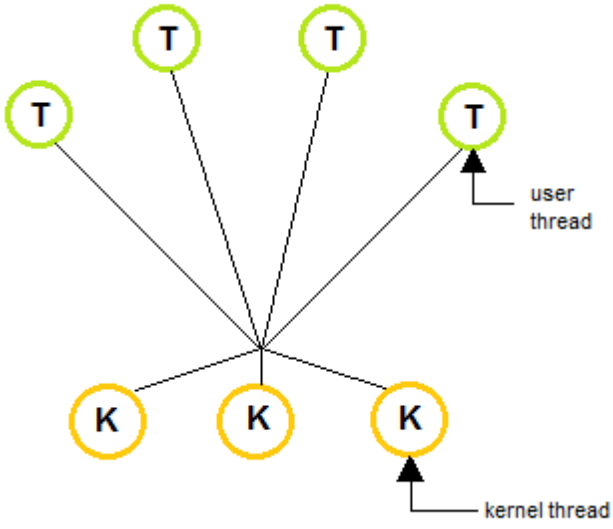
One to One Model

1. The **one to one** model creates a separate kernel thread to handle each and every user thread.
2. Most implementations of this model place a limit on how many threads can be created.
3. Linux and Windows from 95 to XP implement the one-to-one model for threads.
4. This model provides more concurrency than that of many to one Model.



Many to Many Model

1. The **many to many** model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.
2. Users can create any number of threads.
3. Blocking the kernel system calls does not block the entire process.
4. Processes can be split across multiple processors.



What are Thread Libraries?

Thread libraries provide programmers with API for the creation and management of threads.Thread libraries may be implemented either in user space or in kernel space. The user space involves API functions implemented solely within the user space, with no kernel support. The kernel space involves system calls and requires a kernel with thread library support.

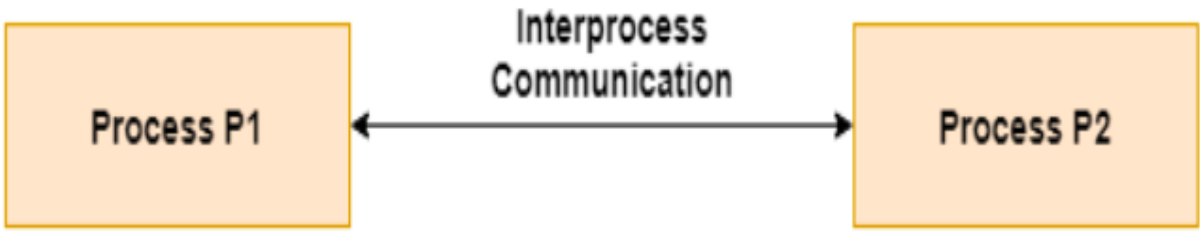
Three types of Threads

1. **POSIX Pitheads** may be provided as either a user or kernel library, as an extension to the POSIX standard.
2. **Win32 threads** are provided as a kernel-level library on Windows systems.
3. **Java threads**: Since Java generally runs on a Java Virtual Machine, the implementation of threads is based upon whatever OS and hardware the JVM is running on, i.e. either Pitheads or Win32 threads depending on the system.

4. Explain about inter-process Communication and its importance.

Ans: Inter-process communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates interprocess communication is as follows −



Synchronization in Inter-process Communication

Synchronization is a necessary part of inter-process communication. It is either provided by the inter-process control mechanism or handled by the communicating processes. Some of the methods to provide synchronization are as follows −

- **Semaphore**

  A semaphore is a variable that controls the access to a common resource by multiple processes. The two types of semaphores are binary semaphores and counting semaphores.

- **Mutual Exclusion**

  Mutual exclusion requires that only one process thread can enter the critical section at a time. This is useful for synchronization and also prevents race conditions.

- **Barrier**

  A barrier does not allow individual processes to proceed until all the processes reach it. Many parallel languages and collective routines impose barriers.

- **Spinlock**

  This is a type of lock. The processes trying to acquire this lock wait in a loop while checking if the lock is available or not. This is known as busy waiting because the process is not doing any useful operation even though it is active.

Approaches to Inter-process Communication

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

- **Pipe**

  A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods. Pipes are used in all POSIX systems as well as Windows operating systems.

- **Socket**

  The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for interprocess communication.

- **File**

  A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.

- **Signal**

  Signals are useful in inter-process communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

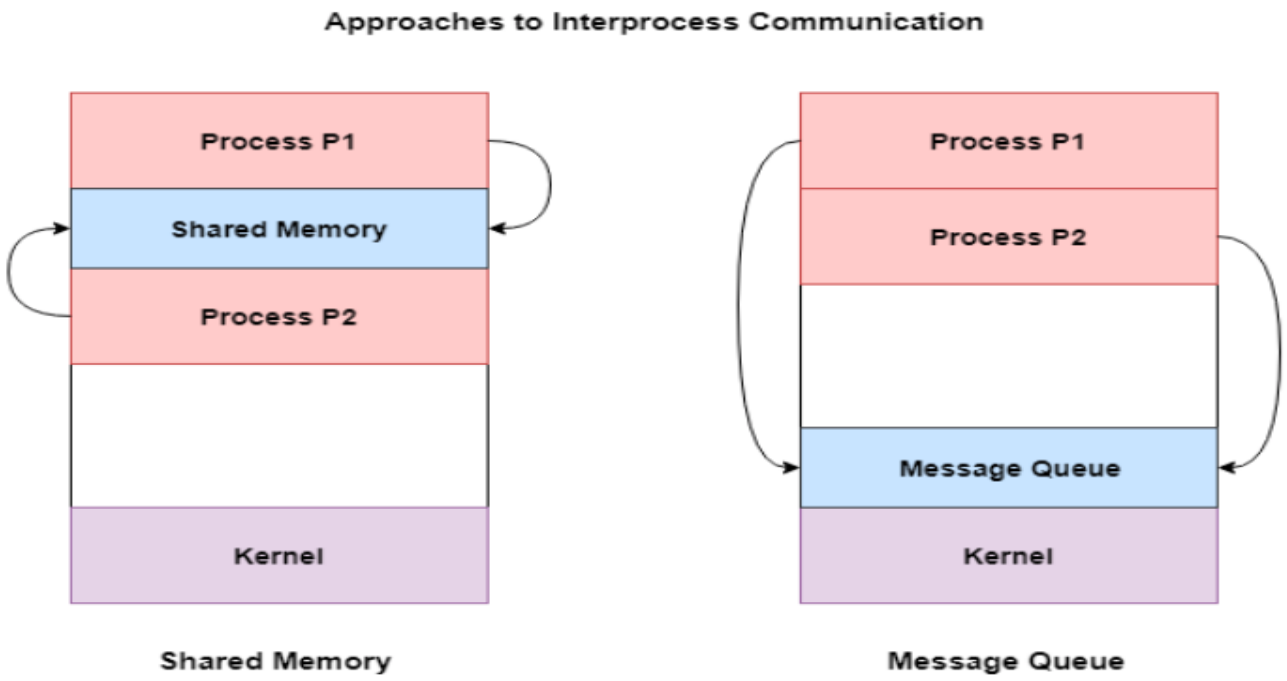
- **Shared Memory**

  Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

- **Message Queue**

  Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for interprocess communication and are used by most operating systems.

A diagram that demonstrates message queue and shared memory methods of interprocess communication is as follows −



5. Discuss about scheduling criteria and various cpu scheduling algorithms.

Scheduling criteria:

Different CPU scheduling algorithms have different properties and the choice of a particular algorithm depends on the various factors. Many criteria have been suggested for comparing CPU scheduling algorithms.

The criteria include the following:

CPUUTILIZATION:

The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.

Throughput:

A measure of the work done by CPU is the number of processes being executed and completed per unit time. This is called throughput. The throughput may vary depending up on the length or duration ofprocesses.

1. **Turnaroundtime** –

   For a particular process, an important criteria is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in ready queue, executing in CPU, and waiting for I/O.

2. **Waitingtime** –

   A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waitinginthereadyqueue.

3. **Responsetime** –

   In an interactive system, turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criteria is the time taken from submission of the process of request until the first response is produced. This measure is called response time.

CPU SCHEDULING ALGORITHMS:

FCFS Scheduling

First come first serve (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs.

Advantages of FCFS

- Simple
- Easy
- First come, First serv

Disadvantages of FCFS

1. The scheduling method is non preemptive, the process will run to the completion.

2. Due to the non-preemptive nature of the algorithm, the problem of starvation may occur.
3. Although it is easy to implement, but it is poor in performance since the average waiting time is higher as compare to other scheduling algorithms.

Example

Let's take an example of The FCFS scheduling algorithm. In the Following schedule, there are 5 processes with process ID **P0, P1, P2, P3 and P4**. P0 arrives at time 0, P1 at time 1, P2 at time 2, P3 arrives at time 3 and Process P4 arrives at time 4 in the ready queue. The processes and their respective Arrival and Burst time are given in the following table.

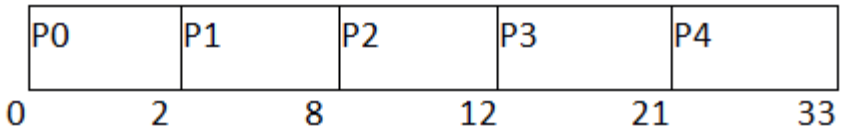
The Turnaround time and the waiting time are calculated by using the following formula.

1 Turn Around Time = Completion Time - Arrival Time
2 Waiting Time = Turnaround time - Burst Time

The average waiting Time is determined by summing the respective waiting time of all the processes and divided the sum by the total number of processes.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
0	0	2	2	2	0
1	1	6	8	7	1
2	2	4	12	8	4
3	3	9	21	18	9
4	4	12	33	29	17

Avg Waiting Time=31/5



(Gantt chart)

Shortest Job First (SJF) Scheduling

Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.

However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

Advantages of SJF

1. Maximum throughput
2. Minimum average waiting and turnaround time

Disadvantages of SJF

1. May suffer with the problem of starvation
2. It is not implementable because the exact Burst time for a process can't be known in advance.

There are different techniques available by which, the CPU burst time of the process can be determined. We will discuss them later in detail.

Example

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

Since, No Process arrives at time 0 hence; there will be an empty slot in the **Gantt chart** from time 0 to 1 (the time at which the first process arrives).

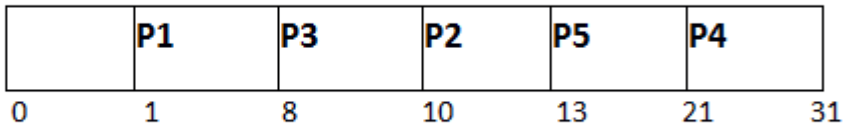
According to the algorithm, the OS schedules the process which is having the lowest burst time among the available processes in the ready queue.

Till now, we have only one process in the ready queue hence the scheduler will schedule this to the processor no matter what is its burst time.

This will be executed till 8 units of time. Till then we have three more processes arrived in the ready queue hence the scheduler will choose the process with the lowest burst time.

Among the processes given in the table, P3 will be executed next since it is having the lowest burst time among all the available processes.

So that's how the procedure will go on in **shortest job first (SJF)** scheduling algorithm.



Avg Waiting Time = 27/5

Shortest Remaining Time First (SRTF) Scheduling Algorithm:

This Algorithm is the **preemptive version** of **SJF scheduling**. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

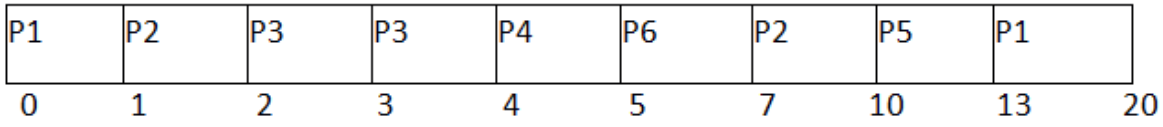
Once all the processes are available in the **ready queue**, No preemption will be done and the algorithm will work as **SJF scheduling**. The context of the process is saved in the **Process Control Block** when the process is removed from the execution and the next process is scheduled. This PCB is accessed on the **next execution** of this process.

Example

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
1	0	8	20	20	12	0
2	1	4	10	9	5	1
3	2	2	4	2	0	2

4	3	1	5	2	1	4
5	4	3	13	9	6	10
6	5	2	7	2	0	5



Avg Waiting Time = 24/6

The Gantt chart is prepared according to the arrival and burst time given in the table.

1. Since, at time 0, the only available process is P1 with CPU burst time 8. This is the only available process in the list therefore it is scheduled.
2. The next process arrives at time unit 1. Since the algorithm we are using is SRTF which is a preemptive one, the current execution is stopped and the scheduler checks for the processwiththeleastbursttime.

   Till now, there are two processes available in the ready queue. The OS has executed P1 for one unit of time till now; the remaining burst time of P1 is 7 units. The burst time of Process P2 is 4 units. Hence Process P2 is scheduled on the CPU according to the algorithm.
3. The next process P3 arrives at time unit 2. At this time, the execution of process P3 is stopped and the process with the least remaining burst time is searched. Since the process P3 has 2 unit of burst time hence it will be given priority over others.
4. The Next Process P4 arrives at time unit 3. At this arrival, the scheduler will stop the execution of P4 and check which process is having least burst time among the available processes (P1, P2, P3 and P4). P1 and P2 are having the remaining burst time 7 units and 3 units respectively.

P3 and P4 are having the remaining burst time 1 unit each. Since, both are equal hence the scheduling will be done according to their arrival time. P3 arrives earlier than P4 and therefore it will be scheduled again.

5. The Next Process P5 arrives at time unit 4. Till this time, the Process P3 has completed its execution and it is no more in the list. The scheduler will compare the remaining burst time of all the available processes. Since the burst time of process P4 is 1 which is least among all hence this will be scheduled.
6. The Next Process P6 arrives at time unit 5, till this time, the Process P4 has completed its execution. We have 4 available processes till now, that are P1 (7), P2 (3), P5 (3) and P6 (2). The Burst time of P6 is the least among all hence P6 is scheduled. Since, now, all the processes are available hence the algorithm will now work same as SJF. P6 will be executed till its completion and then the process with the least remaining time will be scheduled.

Once all the processes arrive, No preemption is done and the algorithm will work as SJF.

Priority Scheduling

In Priority scheduling, there is a priority number assigned to each process. In some systems, the lower the number, the higher the priority. While, in the others, the higher the number, the higher will be the priority. The Process with the higher priority among the available processes is given the CPU. There are two types of priority scheduling algorithm exists. One is **Preemptive** priority s

Non Preemptive Priority Scheduling

In the Non Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them. Once the process gets scheduled, it will run till the completion. Generally, the lower the priority number, the higher is the priority of the process. The people might get confused with the priority numbers, hence in the GATE, there clearly mention which one is the highest priority and which one is the lowest one.

Example

In the Example, there are 7 processes P1, P2, P3, P4, P5, P6 and P7. Their priorities, Arrival Time and burst time are given in the table.

Process ID	Priority	Arrival Time	Burst Time
1	2	0	3
2	6	2	5
3	3	1	4
4	5	4	2
5	7	6	9
6	4	5	4
7	10	7	10

We can prepare the Gantt chart according to the Non Preemptive priority scheduling.

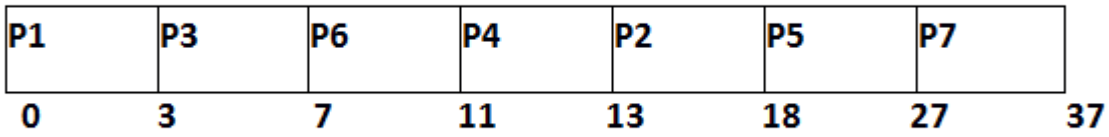
The Process P1 arrives at time 0 with the burst time of 3 units and the priority number 2. Since No other process has arrived till now hence the OS will schedule it immediately.

Meanwhile the execution of P1, two more Processes P2 and P3 are arrived. Since the priority of P3 is 3 hence the CPU will execute P3 over P2.

Meanwhile the execution of P3, All the processes get available in the ready queue. The Process with the lowest priority number will be given the priority. Since P6 has priority number assigned as 4 hence it will be executed just after P3.

After P6, P4 has the least priority number among the available processes; it will get executed for the whole burst time.

Since all the jobs are available in the ready queue hence All the Jobs will get executed according to their priorities. If two jobs have similar priority number assigned to them, the one with the least arrival time will be executed.



From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, waiting time and response time will be determined.

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
1	2	0	3	3	3	0	0
2	6	2	5	18	16	11	13
3	3	1	4	7	6	2	3
4	5	4	2	13	9	7	11
5	7	6	9	27	21	12	18
6	4	5	4	11	6	2	7
7	10	7	10	37	30	18	27

Avg Waiting Time = (0+11+2+7+12+2+18)/7 = 52/7 units

Preemptive Priority Scheduling

In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the preemptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.

Once all the jobs get available in the ready queue, the algorithm will behave as non-preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

Example

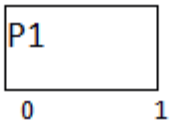
There are 7 processes P1, P2, P3, P4, P5, P6 and P7 given. Their respective priorities, Arrival Times and Burst times are given in the table below.

Process Id	Priority	Arrival Time	Burst Time
1	2(L)	0	1
2	6	1	7
3	3	2	3
4	5	3	6
5	4	4	5

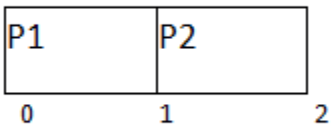
6	10(H)	5	15
7	9	15	8

GANTT chart Preparation

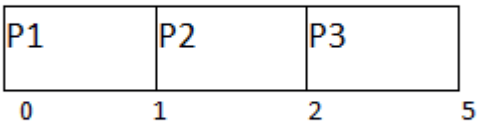
At time 0, P1 arrives with the burst time of 1 units and priority 2. Since no other process is available hence this will be scheduled till next job arrives or its completion (whichever is lesser).



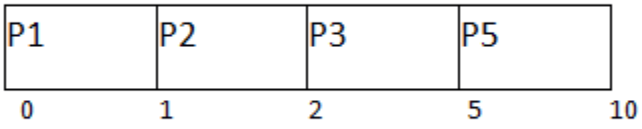
At time 1, P2 arrives. P1 has completed its execution and no other process is available at this time hence the Operating system has to schedule it regardless of the priority assigned to it.



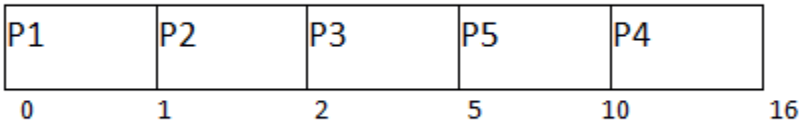
The Next process P3 arrives at time unit 2, the priority of P3 is higher to P2. Hence the execution of P2 will be stopped and P3 will be scheduled on the CPU.



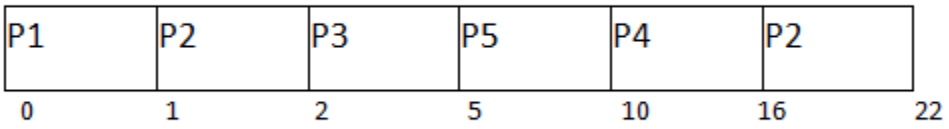
During the execution of P3, three more processes P4, P5 and P6 becomes available. Since, all these three have the priority lower to the process in execution so PS can't preempt the process. P3 will complete its execution and then P5 will be scheduled with the priority highest among the available processes.



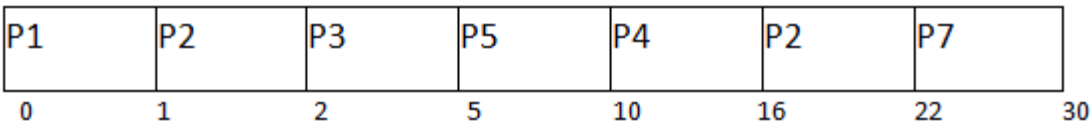
Meanwhile the execution of P5, all the processes got available in the ready queue. At this point, the algorithm will start behaving as Non Preemptive Priority Scheduling. Hence now, once all the processes get available in the ready queue, the OS just took the process with the highest priority and execute that process till completion. In this case, P4 will be scheduled and will be executed till the completion.



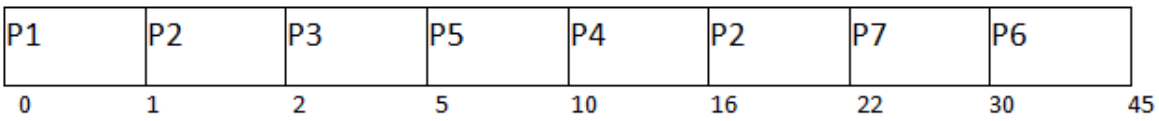
Since P4 is completed, the other process with the highest priority available in the ready queue is P2. Hence P2 will be scheduled next.



P2 is given the CPU till the completion. Since its remaining burst time is 6 units hence P7 will be scheduled after this.



The only remaining process is P6 with the least priority, the Operating System has no choice unless of executing it. This will be executed at the last.



The Completion Time of each process is determined with the help of GANTT chart. The turnaround time and the waiting time can be calculated by the following formula.

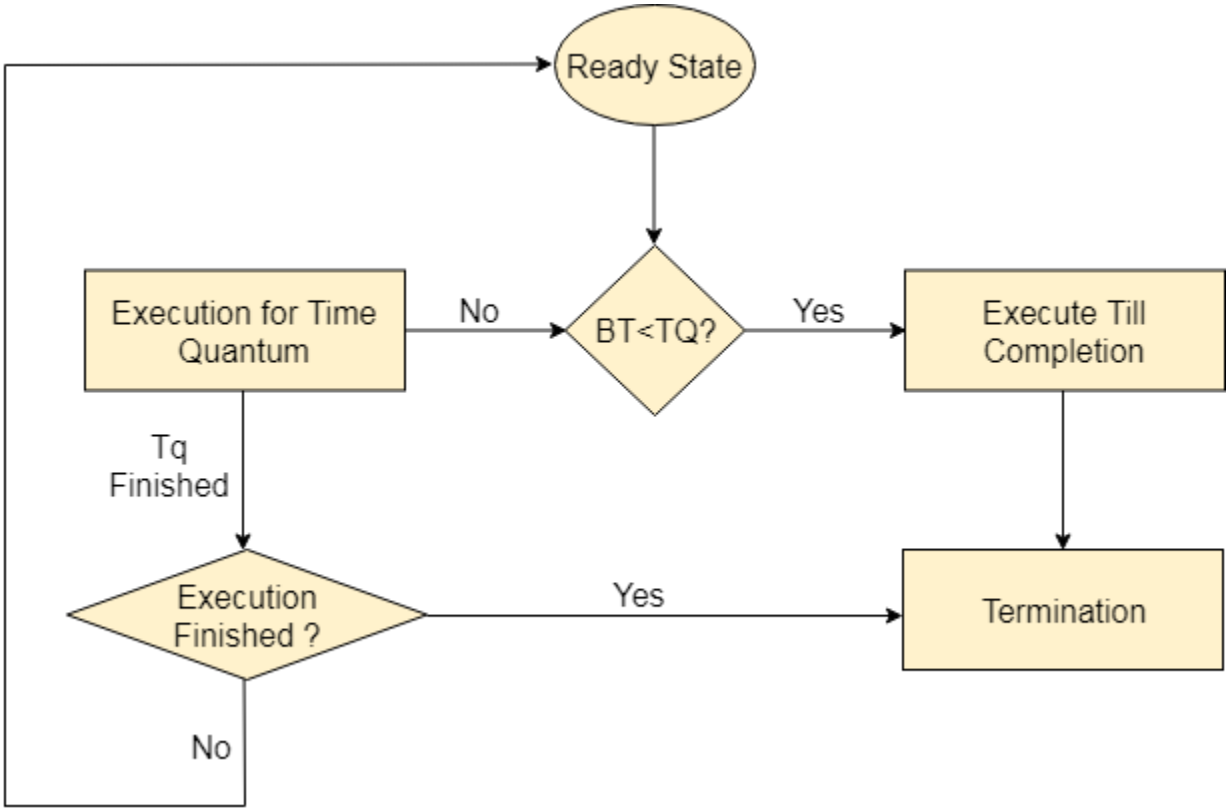
1. Turnaround Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turn around Time	Waiting Time
1	2	0	1	1	1	0
2	6	1	7	22	21	14
3	3	2	3	5	3	0
4	5	3	6	16	13	7
5	4	4	5	10	6	1
6	10	5	15	45	40	25
7	9	6	8	30	24	16

Avg Waiting Time = (0+14+0+7+1+25+16)/7 = 63/7 = 9 units

Round Robin Scheduling Algorithm

Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the **preemptive version** of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a **cyclic way**. A certain time slice is defined in the system which is called time **quantum**. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **terminate** else the process will go back to the **ready queue** and waits for the next turn to complete the execution.



Advantages

1. It can be actually implementable in the system because it is not depending on the burst time.
2. It doesn't suffer from the problem of starvation or convoy effect.
3. All the jobs get a fare allocation of CPU.

Disadvantages

1. The higher the time quantum, the higher the response time in the system.
2. The lower the time quantum, the higher the context switching overhead in the system.
3. Deciding a perfect time quantum is really a very difficult task in the system.

RR Scheduling Example

In the following example, there are six processes named as P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. The time quantum of the system is 4 units.

Process ID	Arrival Time	Burst Time
1	0	5
2	1	6
3	2	3
4	3	1
5	4	5
6	6	4

According to the algorithm, we have to maintain the ready queue and the Gantt chart. The structure of both the data structures will be changed after every scheduling.

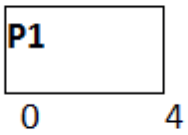
Ready Queue:

Initially, at time 0, process P1 arrives which will be scheduled for the time slice 4 units. Hence in the ready queue, there will be only one process P1 at starting with CPU burst time 5 units.

P1
5

GANTT chart

The P1 will be executed for 4 units first.



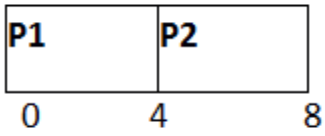
Ready Queue

Meanwhile the execution of P1, four more processes P2, P3, P4 and P5 arrives in the ready queue. P1 has not completed yet, it needs another 1 unit of time hence it will also be added back to the ready queue.

P2	P3	P4	P5	P1
6	3	1	5	1

GANTT chart

After P1, P2 will be executed for 4 units of time which is shown in the Gantt chart.



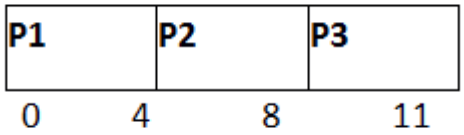
Ready Queue

During the execution of P2, one more process P6 is arrived in the ready queue. Since P2 has not completed yet hence, P2 will also be added back to the ready queue with the remaining burst time 2 units.

P3	P4	P5	P1	P6	P2
3	1	5	1	4	2

GANTT chart

After P1 and P2, P3 will get executed for 3 units of time since its CPU burst time is only 3 seconds.



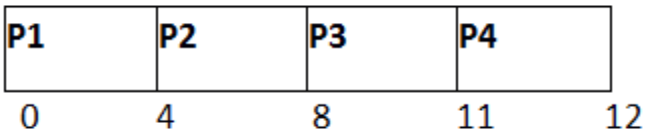
Ready Queue

Since P3 has been completed, hence it will be terminated and not be added to the ready queue. The next process will be executed is P4.

P4	P5	P1	P6	P2
1	5	1	4	2

GANTT chart

After, P1, P2 and P3, P4 will get executed. Its burst time is only 1 unit which is lesser then the time quantum hence it will be completed.



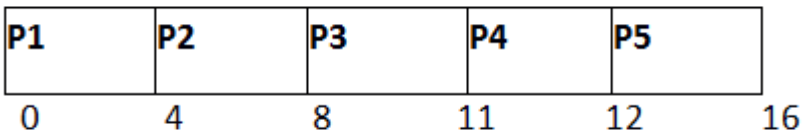
Ready Queue

The next process in the ready queue is P5 with 5 units of burst time. Since P4 is completed hence it will not be added back to the queue.

P5	P1	P6	P2
5	1	4	2

GANTT chart

P5 will be executed for the whole time slice because it requires 5 units of burst time which is higher than the time slice.



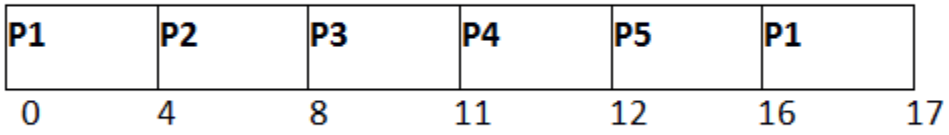
Ready Queue

P5 has not been completed yet; it will be added back to the queue with the remaining burst time of 1 unit.

P1	P6	P2	P5
1	4	2	1

GANTT Chart

The process P1 will be given the next turn to complete its execution. Since it only requires 1 unit of burst time hence it will be completed.



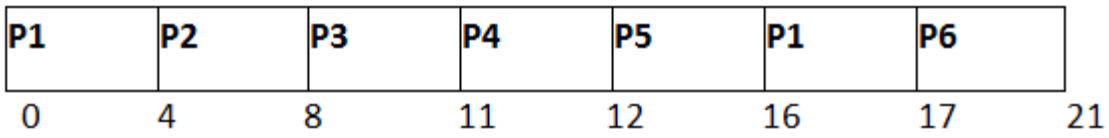
Ready Queue

P1 is completed and will not be added back to the ready queue. The next process P6 requires only 4 units of burst time and it will be executed next.

P6	P2	P5
4	2	1

GANTT chart

P6 will be executed for 4 units of time till completion.



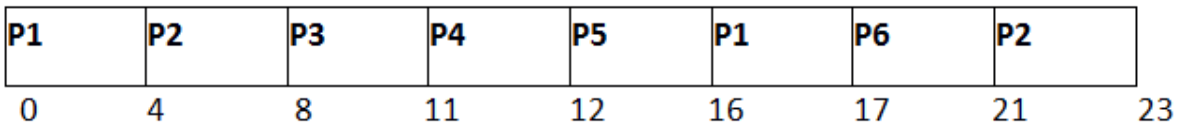
Ready Queue

Since P6 is completed, hence it will not be added again to the queue. There are only two processes present in the ready queue. The Next process P2 requires only 2 units of time.

P2	P5
2	1

GANTT Chart

P2 will get executed again, since it only requires only 2 units of time hence this will be completed.



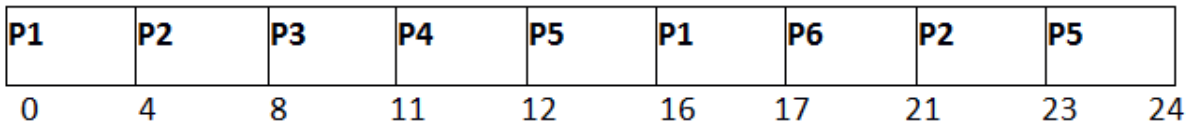
Ready Queue

Now, the only available process in the queue is P5 which requires 1 unit of burst time. Since the time slice is of 4 units hence it will be completed in the next burst.

P5
1

GANTT chart

P5 will get executed till completion.



The completion time, Turnaround time and waiting time will be calculated as shown in the table below.

As, we know,

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

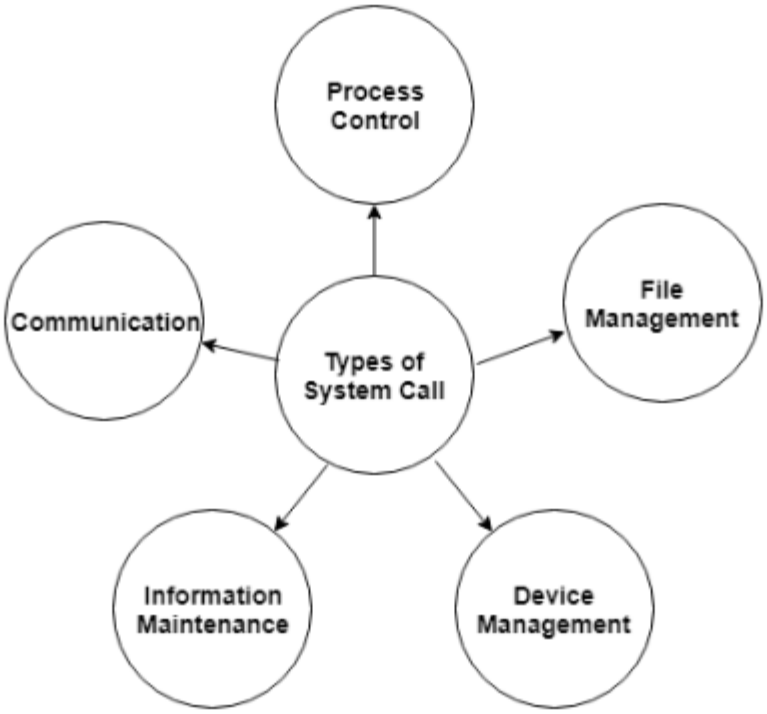
Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	17	17	12
2	1	6	23	22	16
3	2	3	11	9	6
4	3	1	12	9	8
5	4	5	24	20	15
6	6	4	21	15	11

Avg Waiting Time = (12+16+6+8+15+11)/6 = 76/6 units

6.Explain system call interface for process management.

Types of System Calls

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



Here are the types of system calls −

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()	pipe() shmget()

Types of System Calls	Windows	Linux
	MapViewOfFile()	mmap()

There are many different system calls as shown above. Details of some of those system calls are as follows −

wait()

In some systems, a process may wait for another process to complete its execution. This happens when a parent process creates a child process and the execution of the parent process is suspended until the child process executes. The suspending of the parent process occurs with a wait() system call. When the child process completes execution, the control is returned back to the parent process.

exec()

This system call runs an executable file in the context of an already running process. It replaces the previous executable file. This is known as an overlay. The original process identifier remains since a new process is not created but data, heap, stack etc. of the process are replaced by the new process.

fork()

Processes use the fork() system call to create processes that are a copy of themselves. This is one of the major methods of process creation in operating systems. When a parent process creates a child process and the execution of the parent process is suspended until the child process executes. When the child process completes execution, the control is returned back to the parent process.

exit()

The exit() system call is used by a program to terminate its execution. In a multithreaded environment, this means that the thread execution is complete. The operating system reclaims resources that were used by the process after the exit() system call.

kill()

The kill() system call is used by the operating system to send a termination signal to a process that urges the process to exit.However, kill system call does not necessary mean killing the process and can have various meanings.

UNIT-III

1. Explain about Deadlock in detailed.

Deadlock:

❖ when several processes compete for a finite number of resources, a situation may arise where a process requests a resource and the resource is not available at that time, in that time the process enters a wait state.

❖ It may happen that waiting processes will never again change state because the resources that they have requested are held by other waiting processes.

❖ This situation is called a deadlock.

❖ Deadlock can arise if four conditions hold simultaneously.

1) **Mutual exclusion**: only one process at a time can use a resource.

2) **Hold and wait**: a process holding at least one resource is waiting to acquire additional resources held by other processes.

3) **No preemption**: a resource can be released only voluntarily by the process holding it, after that process has completed its task.

4) **Circular wait:** there exists a set {P0, P1, …, P0} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, …, Pn–1 is waiting for a resource that is held by Pn, and P0 is waiting for a resource that is held by P0.

2. Write about Resource-Allocation Graph .

Resource-Allocation Graph:

❖ Deadlocks can be described in terms of a directed graph called a system resourceallocation graph.

❖ This graph consists of a set of vertices V and a set of edges E.

1) V is partitioned into two types: a) P = {P1, P2, …, Pn}, the set consisting of all the processes in the system.

b) R = {R1, R2, …, Rm}, the set consisting of all resource types in the system

2) Request edge – directed edge P1 ® Rj

3) Assignment edge – directed edge Rj ® Pi

The following symbols are used in the resource allocation graph: a) Process

b) Resource Type with 4 instances

c) Pi requests instance of Rj

d) Pi is holding an instance of Rj

Example of a Resource Allocation Graph :

The above graph is describing the following situation :

1) The sets P, R and E are : P = { P1, P2, P3 }

R = { R1, R2, R3 } E = { P1 -> R1, P2 -> R3, R1 -> P2, R2 -> P2, R2 -> P1, R3 -> P3 }

2) Resource instances :

- One instance of resource type R1 - Two instances of resources type R2

- One instance of resource type R3 - Three instances of resource type R4

3) Process states :

- Process P1 is holding an instance of resource type R2 and is waiting for an instance of resource type R1.

- Process P2 is holding an instance of R1 and R2 and is waiting for an instance of resource type R3.

- Process P3 is holding an instance of R3.

If the graph consists no cycles, then no process in the system is deadlocked. If the graph contains a cycle, then a deadlock may exist.

3.Explain about Deadlock prevention

✓ Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold.

a) Mutual Exclusion:

✓ The mutual exclusion condition must hold for non sharable resources. For example, a printer can not be accessed by more than one process at a time.

✓ But Read-only files are good example for sharable resources and can not be involved in deadlock.

✓ A process never needs to wait for sharable resources.

✓ Hence we can not prevent deadlocks by denying the mutual exclusion condition because some resources are basically non sharable. b) Hold and Wait:

✓ To ensure that the hold and wait condition never occurs in the system. There are two protocols to break the hold and wait.

Protocol 1 – Each process to request and be allocated all its resources before it begins execution.

Protocol 2 – A process request some resources only when the process has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources which are hold by it.

✓ By using an example we differentiate between these two protocols: A process that copies data from a tape drive to a disk file, sorts the disk file and then prints the results to a printer.

✓ According to protocol 1, the process must request all the resources such as tape drive, disk drive and printer before staring its execution.

✓ If they are available, they will be allocated to it. Otherwise, the process has to wait to start its execution until getting the all resources.

✓ Though all resources are allocated initially, the process can use the printer at the end.

✓ According to protocol 2, the process can request initially only the tape drive and disk file.

✓ It copies from the tape drive from disk and release both. Again the process request the disk file and printer.

✓ After copying the disk file to the printer, it releases two resources and terminates

Disadvantages: These two protocols have two main disadvantages.

1) Resource utilization may be low – many of the resources may be allocated but unused for a long period. In the example, the process release the tape drive and disk file, and then again request the disk file printer. If they are available at that time we can allocate them. Otherwise the process must wait.

2) Starvation is possible – A process that needs several popular resources may have to wait indefinitely because at least one of the resources that it needs is always allocated to some other process.

c) No Preemption: The third necessary condition is that there is no preemption of resources that have already been allocated. To ensure that this condition does not hold, we can use the following two protocols:

Protocol 1 –

✓ If a process is holding some resources and requests another resource that cannot be immediately allocated to it.

✓ All the resources currently being are preempted. The preempted resources are added to the list of resources for which the process is waiting.

✓ The process will be restarted when it regain its old resources as well as the new ones. Protocol 2 –

✓ A process requests some resources, we first check whether they are available. If they are available, we can allocate them.

✓ If they are not available, first we check whether they are allocated to some other process that is waiting for additional resources.

✓ If so, preempt them from that process and allocated to the requesting process.

✓ If they are not available, the requesting process must wait. ✓ It may be happen that while it is waiting, some of its existing resources may be preempted due to the requesting of some other process.

d) Circular Wait: The fourth and final condition for deadlocks is the circular-wait. To ensure that this condition never holds in the system. To do so, each process requests resources in an increasing order of enumeration. Let R = { R1, R2, R3, …….., Rm } be the set of resource types. We assign to each resource type unique integer number, which allows us to compare two resources and to determine whether one precedes another in our ordering.

We define a one-to-one function F: R->N, Where N is the set of natural numbers.

For example, F (tape drive) =1 F (disk drive) =5

F (printer) =12

✓ To prevent the deadlocks we can follow the following protocols:

Protocol 1-

✓ Each process can request resources in an increasing order only. A process can request the resources if and only if F (Rj) > F (Ri).

✓ If several instances of the same resource type are needed, as single request for all of them must be issued.

Protocol 2 –

✓ Whenever a process requests an instance of resource type Rj , it has released any resources Ri such that F (Ri) ³ F (Rj).

✓ If these two protocols are used, then the circular wait condition cannot be hold.

✓ Let the set of processes { P0, P1, ……., Pn }where Pi is waiting for a resource Ri , which is held by Pi+1.

✓ Process Pi+1 is holding resource Ri, while requesting resource Ri+1, we must have F (Ri) < F (Ri+1), for all i.

✓ But this condition means that F (R0) < F (R1) < …….. F (Rn) < F (R0). By transitivity, F (R0) < F (R0), which is impossible.

✓ Therefore, there can be no circular wait.

If we follow these protocols to fail one of the 4 conditions for the deadlock, we can easily prevent deadlock.

4.Explain about Deadlock Avoidance with an example.

✓ It requires that the operating system be given in advance additional information concerning which resources a process will request.

✓ And use during its lifetime.

✓ With this knowledge we can make a decision whether the current process request can be satisfied or delayed.

✓ A deadlock avoidance algorithm dynamically examines the resource-allocation state to ensure that a circular wait condition can never exist.

✓ The number of available and allocated resources and the maximum demands of the processes define the resource allocation state.

✓ A state is safe if the system can allocate the resources to each process (up its maximum) in some order and still avoid a deadlock.

✓ A system is in a safe state only there exists a safe sequence.

✓ A sequence of processes < P1, P2 ……., Pn> is a safe sequence for the current allocation state if, for each Pi, the resources that Pi can still request can be satisfied by the currently available resources plus the resources held by all the Pj, with j < i. In this situation,

a) If Pi resource needs are not immediately available, then Pi can wait until all Pj have finished.

b) When Pj is finished, Pi can obtain needed resources, execute, and return allocated resources, and terminate.

c) When Pi terminates, Pi+1 can obtain its needed resources, and so on. If there is no safe sequence, the system is said to be unsafe.

✓ A safe sate is not a deadlock state. Conversely, a deadlock state is an unsafe state.

✓ Not all unsafe states are deadlocks. In an unsafe state, the operating system cannot prevent processes from requesting resources such that a deadlock occurs.

✓ The behavior of the processes controls unsafe state.

Ex: Suppose a system with 12 magnetic tape drives and 3 processes: P0, P1 and P2 . Process P0 requires 10 tape drives, process P1 requires 4 and process P2 requires 9 tape drives.Suppose at time t0, Process P0 is holding 5 tape drives, process P1 is holding 2 tape drives and process P2 is holding 2 tape drives. Thus there are 3 free tape drives.

Process Maximum Needs Current Needs P0 10 5 P1 4 2 P2 9 2

x: Suppose a system with 12 magnetic tape drives and 3 processes: P0, P1 and P2 . Process P0 requires 10 tape drives, process P1 requires 4 and process P2 requires 9 tape drives.Suppose at time t0, Process P0 is holding 5 tape drives, process P1 is holding 2 tape drives and process P2 is holding 2 tape drives. Thus there are 3 free tape drives.

Process Maximum Needs Current Needs P0 10 5 P1 4 2 P2 9 2

At time t0, the system is in safe state. The sequence <P1, P0, P2> satisfies the safety condition.

Safety Algorithm:

This algorithm is used to find out whether or not the system is in a safe state.

This algorithm consists the following steps:

1. Let Work and Finish be vectors of length m and n, respectively. Initialize: Work = Available

Finish [i] = false for i - 1,3, …, n.2.

2. Find and i such that both:

(a) Finish [i] = false (b) Needi Work If no such i exists, go to step 4.

3. Work = Work + Allocationi Finish[i] = true

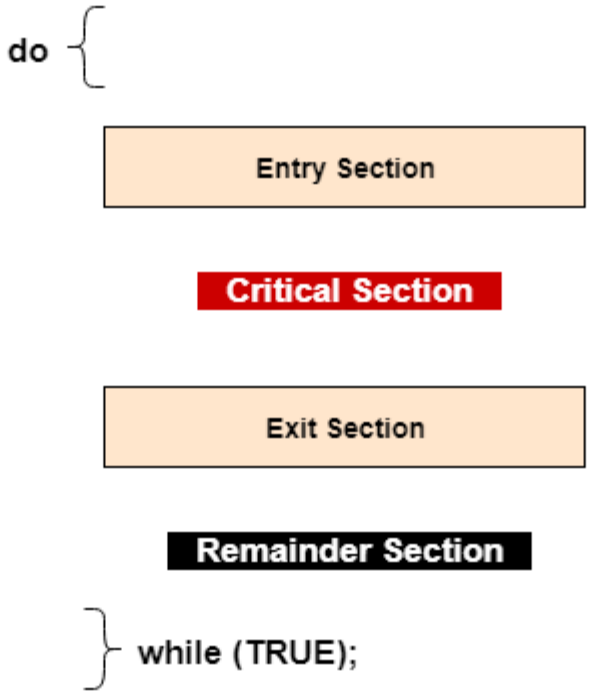
go to step 2.

4. If Finish [i] == true for all i, then the system is in a safe state.

5. Discuss about critical section problem.

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.

A diagram that demonstrates the critical section is as follows −



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.

Solution to the Critical Section Problem

The critical section problem needs a solution to synchronize the different processes. The solution to the critical section problem must satisfy the following conditions −

- **Mutual Exclusion**

  Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

- **Progress**

Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.

- **Bounded Waiting**

  Bounded waiting means that each process must have a limited waiting time. Itt should not wait endlessly to access the critical section.

6.Explain about Semaphore in detailed.

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.

The definitions of wait and signal are as follows −

- **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++;
```

```
}
```

Types of Semaphores

There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows −

- **Counting Semaphores**

  These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources. If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.

- **Binary Semaphores**

  The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

Advantages of Semaphores

Some of the advantages of semaphores are as follows −

- Semaphores allow only one process into the critical section. They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.
- There is no resource wastage because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
- Semaphores are implemented in the machine independent code of the microkernel. So they are machine independent.

Disadvantages of Semaphores

Some of the disadvantages of semaphores are as follows −

- Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.

- Semaphores are impractical for last scale use as their use leads to loss of modularity. This happens because the wait and signal operations prevent the creation of a structured layout for the system.
- Semaphores may lead to a priority inversion where low priority processes may access the critical section first and high priority processes later.

Classical probems of synchronization are as follows:

• The Bounded Buffer Problem (also called the The Producer-Consumer Problem)

• The Readers-Writers Problem

• The Dining Philosophers Problem

These problems are used to test nearly every newly proposed synchronization scheme or primitive.

1. **The Bounded Buffer Problem(Producer Consumer Problem):**

Consider,

• a buffer which can store n items

• a producer process which creates the items (1 at a time)

• a consumer process which processes them (1 at a time)

A producer cannot produce unless there is an empty buffer slot to fill.

A consumer cannot consume unless there is at least one produced item.

Semaphore empty=N, full=0, mutex=1;

process producer

{

while (true)

{

empty.acquire();

mutex.acquire();

```
// produce
mutex.release();
full.release();
}
}
process consumer
{
while (true) {
full.acquire();
mutex.acquire();
// consume
mutex.release();
empty.release();
}
}
```

The semaphore mutex provides mutual exclusion for access to the buffer

1.The Readers-Writers Problem

A data item such as a file is shared among several processes.

Each process is classified as either a reader or writer.Multiple readers may access the file simultaneously.

A writer must have exclusive access (i.e., cannot share with either a reader or another writer).A solution gives priority to either readers or writers.

• **readers' priority:** no reader is kept waiting unless a writer has already obtained permission to access the database

• **writers' priority:** if a writer is waiting to access the database, no new readers can start reading.

A solution to either version may cause starvation in the readers' priority version, writers may starving the writers' priority version, readers may starve

A semaphore solution to the readers' priority version (without addressing starvation):

```
Semaphore mutex = 1;
Semaphore db = 1;
int readerCount = 0;
process writer {
db.acquire();
// write
db.release();
}
process reader {
// protecting readerCount
mutex.acquire();
++readerCount;
if (readerCount == 1)
db.acquire();
mutex.release();
// read
// protecting readerCount
mutex.acquire();
--readerCount;
if (readerCount == 0)
```

db.release;

mutex.release();

}

readerCount is a <cs> over which we must maintain control and we use mutex to do so.

3. The Dining Philosophers Problem

philosophers sit around a table thinking and eating. When a philosopher thinks she does not interact with her colleagues. Periodically, a philosopher gets hungry and tries to pick up the chopstick on his left and on his right. A philosopher may only pick up one chopstick at a time and, obviously, cannot pick up a chopstick already in the hand of neighbor philosopher.

The dining philosophers problems is an example of a large class or concurrency control problems; it is a simple representation of the need to allocate several resources among several processes in a deadlock-free and starvation-free manner.

A semaphore solution:

// represent each chopstick with a semaphore

Semaphore chopstick[] = new Semaphore[5]; // all = 1 initially

process philosopher_i

{

while (true)

{

// pick up left chopstick

chopstick[i].acquire();

// pick up right chopstick

chopstick[(i+1) % 5].acquire();

// eat

// put down left chopstick

```
chopstick[i].release();

// put down right chopstick

chopstick[(i+1) % 5].release();

// think

}

}
```

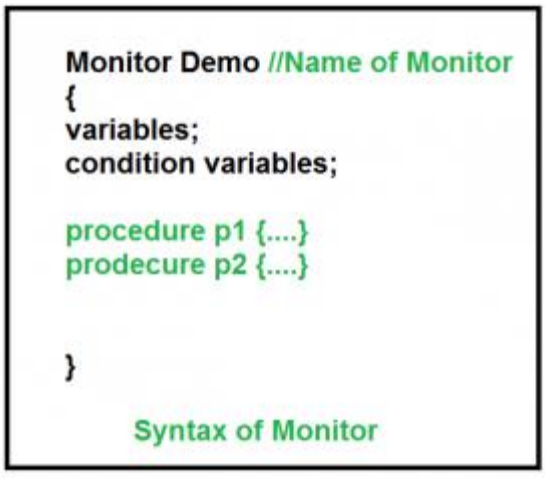
This solution guarantees no two neighboring philosophers eat simultaneously, but has the possibility of creating a deadlock.

7. Discuss about monitors in operating systems with example.

The monitor is one of the ways to achieve Process synchronization. The monitor is supported by programming languages to achieve mutual exclusion between processes. For example Java Synchronized methods. Java provides wait() and notify() constructs.

1. It is the collection of condition variables and procedures combined together in a special kind of module or a package.
2. The processes running outside the monitor can't access the internal variable of the monitor but can call procedures of the monitor.
3. Only one process at a time can execute code inside monitors.

Syntax:



Condition Variables:

Two different operations are performed on the condition variables of the monitor.

Wait.

signal.

letsaywehave2conditionvariables

condition x, y; // Declaring variable

Waitoperation

x.wait() : Process performing wait operation on any condition variable are suspended. The suspended processes are placed in block queue of that condition variable.

Note: Each condition variable has its unique block queue.

Signaloperation

x.signal(): When a process performs signal operation on condition variable, one of the blocked processes is given chance.

If (x block queue empty)

// Ignore signal

else

// Resume a process from block queue.

AdvantagesofMonitor:

Monitors have the advantage of making parallel programming easier and less error prone than using techniques such as semaphore.

DisadvantagesofMonitor:

Monitors have to be implemented as part of the programming language . The compiler must generate code for them. This gives the compiler the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes. Some languages that do support monitors are Java, C#, Visual Basic ,Ada and concurrent Euclid.

8.Discuss briefly about Inter process communication in process synchronization.

IPC: Inter-process Communication

IPC stands for Inter-process Communication. As the name suggests, it is a technology that ***enables multiple threads in one or more processes to communicate with each other***. It allows the exchange of data between multiple threads of one or different processes or systems connected via a network.

This technology can also handle multiple requests simultaneously which means it can be used for sharing data between threads in a single or multiple processes. IPC is used by operating systems to share a large amount of data. However, it is not supported by a single process operating system like DOS and it can also affect performance.

In this technology, the requests are synchronized and implemented simultaneously. So, all the processes run smoothly and communicate with each other without affecting each other and cooperating with each other. Thus, it enables a programmer to coordinate the activities of different programs that are running simultaneously in an operating system. Using this technology, a program can handle multiple requests of users simultaneously.

Characteristics of IPC:

- **Synchronous or Asynchronous communication**: A message is added to remote queues by sending process, whereas, it is removed by the receiving process from the local queues. This communication between these two processes can be synchronous or asynchronous.
- **Message destinations**: It is a local port that remains in the computer and is known as an integer. It has one receiver and multiple senders.
- **Reliability**: It is reliable communication. The message is sent securely without any loss in packets and any duplication or corruption.
- **Ordering**: The messages should be sent only in sender order.

Advantages of IPC:

- It makes information sharing possible simultaneously between different programs.
- A task can be separated into subtasks and run on separate processors and then using IPC exchange of information can take place.
- In IPC, it is easy to maintain and debug the program as it can be separated into different chunks of code, which can function independently.
- The programmer can perform various other tasks simultaneously such as listening to music, editing, compiling, and more.

Disadvantages of IPC:

- It can be slow as compared to direct function calls.
- It requires to write some messages passing APIs if you want to write an OS piece.
- Some issues related to memory protection and synchronization may occur that should be resolved.
- It does not allow the processors to write to the same memory location.

VARIOUS APPROACHES IN IPC

Pipes

Pipe is widely used for communication between two related processes. This is a half-duplex method, so the first process communicates with the second process. However, in order to achieve a full-duplex, another pipe is needed.

Message Passing:

It is a mechanism for a process to communicate and synchronize. Using message passing, the process communicates with each other without resorting to shared variables.

IPC mechanism provides two operations:

- Send (message)- message size fixed or variable
- Received (message)

Message Queues:

A message queue is a linked list of messages stored within the kernel. It is identified by a message queue identifier. This method offers communication between single or multiple processes with full-duplex capacity.

Direct Communication:

In this type of inter-process communication process, should name each other explicitly. In this method, a link is established between one pair of communicating processes, and between each pair, only one link exists.

Indirect Communication:

Indirect communication establishes like only when processes share a common mailbox each pair of processes sharing several communication links. A link can communicate with many processes. The link may be bi-directional or unidirectional.

Shared Memory:

Shared memory is a memory shared between two or more processes that are established using shared memory between all the processes. This type of memory requires to protected from each other by synchronizing access across all the processes.

FIFO:

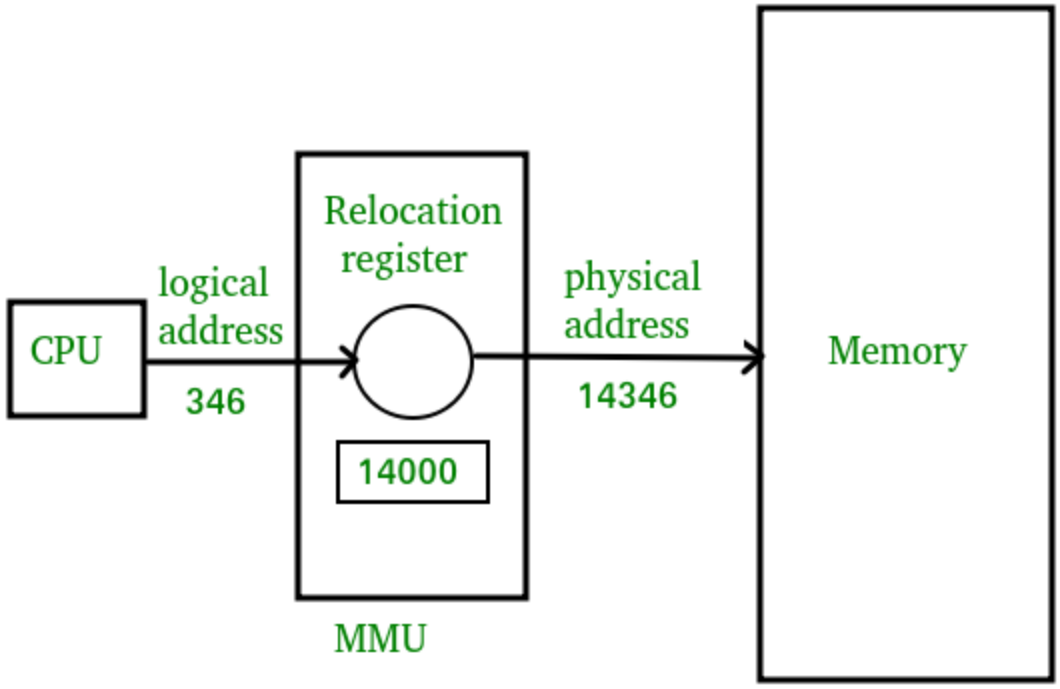
Communication between two unrelated processes. It is a full-duplex method, which means that the first process can communicate with the second process, and the opposite can also happen.

UNIT-IV

1. Discuss Logical and physical address space in memory management.

Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective.
The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logicaladdressspace.



Mapping virtual-address to physical-addresses

Differences between Logical and Physical Address in Operating System

1. The basic difference between Logical and physical address is that Logical address is generated by CPU in perspective of a program whereas the physical address is a location that exists in the memory unit.
2. Logical Address Space is the set of all logical addresses generated by CPU for a program whereas the set of all physical address mapped to corresponding logical addresses is called Physical Address Space.
3. The logical address does not exist physically in the memory whereas physical address is a location in the memory that can be accessed physically.
4. Identical logical addresses are generated by Compile-time and Load time address binding methods whereas they differs from each other in run-time address binding method. Please refer this for details.
5. The logical address is generated by the CPU while the program is running whereas the physical address is computed by the Memory Management Unit (MMU).

Comparison Chart:

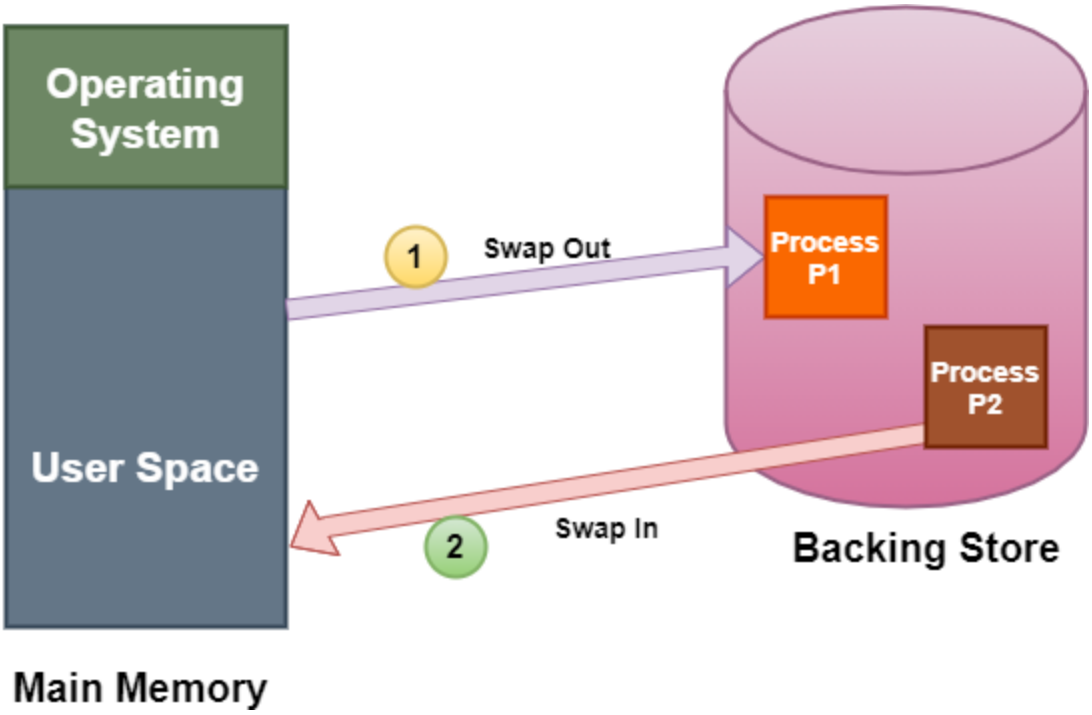
Parameter	LOGICAL ADDRESS	PHYSICAL ADDRESS
Basic	generated by CPU	location in a memory unit
Address Space	Logical Address Space is set of all logical addresses generated by CPU in reference to a program.	Physical Address is set of all physical addresses mapped to the corresponding logical addresses.
Visibility	User can view the logical address of a program.	User can never view physical address of program.
Generation	generated by the CPU	Computed by MMU
Access	The user can use the logical address to access the physical address.	The user can indirectly access physical address but not directly.

2. Explain about swapping and contiguous Allocation.

Swapping is a memory management technique and is used to temporarily remove the inactive programs from the main memory of the computer system. Any process must be in the memory for its execution, but can be swapped temporarily out of memory to a backing store and then again brought back into the memory to complete its execution. Swapping is done so that other processes get memory for their execution.

Due to the swapping technique performance usually gets affected, but it also helps in running multiple and big processes in parallel. **The swapping** process is also known as a technique for **memory compaction.** Basically, low priority processes may be swapped out so that processes with a higher priority may be loaded and executed.

Let us understand this technique with the help of a figure given below:



The above diagram shows swapping of two processes where the disk is used as a Backing store.

In the above diagram, suppose there is a multiprogramming environment with a round-robin scheduling algorithm; whenever the time quantum expires then the memory manager starts to swap out those processes that are just finished and swap another process into the memory that has been freed. And in the meantime, the CPU scheduler allocates the time slice to some other processes in the memory.

The swapping of processes by the memory manager is fast enough that some processes will be in memory, ready to execute, when the CPU scheduler wants to reschedule the CPU.

A variant of the swapping technique is the priority-based scheduling algorithm. If any higher-priority process arrives and wants service, then the memory manager swaps out lower priority processes and then load the higher priority processes and then execute them. When the process with higher priority finishes .then the process with lower priority swapped back in and continues its execution. This variant is sometimes known as roll in and roll out.

There are two more concepts that come in the swapping technique and these are: **swap in** and **swap out.**

Swap In and Swap Out in OS

The procedure by which any process gets removed from the **hard disk** and placed in the **main memory or RAM** commonly known as **Swap In.**

On the other hand, **Swap Out** is the method of removing a process from the **main memory or RAM** and then adding it to the **Hard Disk.**

Advantages of Swapping

The advantages/benefits of the Swapping technique are as follows:

1. The swapping technique mainly helps the CPU to manage multiple processes within a single main memory.
2. This technique helps to create and use virtual memory.
3. With the help of this technique, the CPU can perform several tasks simultaneously. Thus, processes need not wait too long before their execution.
4. This technique is economical.
5. This technique can be easily applied to priority-based scheduling in order to improve its performance.

Disadvantages of Swapping

The drawbacks of the swapping technique are as follows:

1. There may occur inefficiency in the case if a resource or a variable is commonly used by those processes that are participating in the swapping process.
2. If the algorithm used for swapping is not good then the overall method can increase the number of page faults and thus decline the overall performance of processing.
3. If the computer system loses power at the time of high swapping activity then the user might lose all the information related to the program.

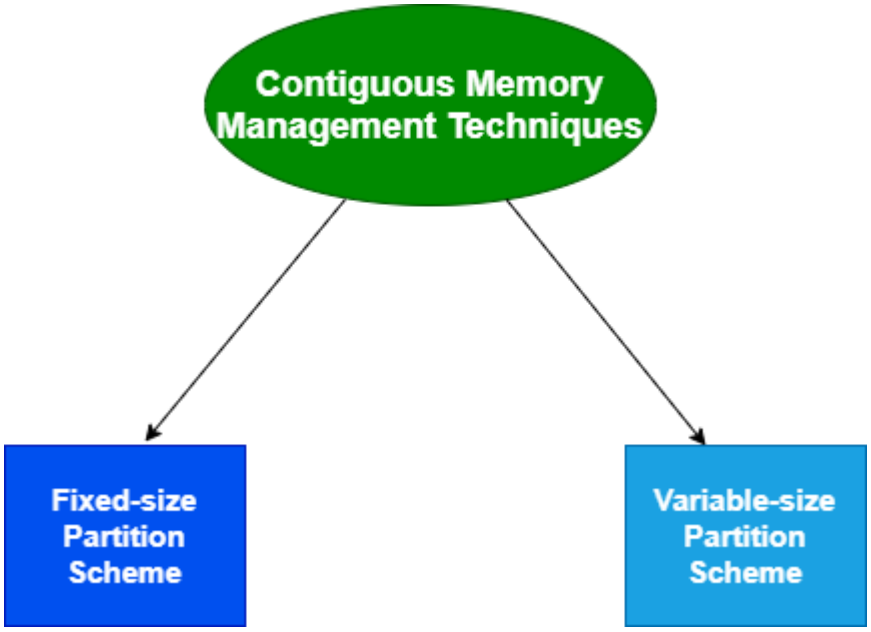
In the Contiguous Memory Allocation, each process is contained in a single contiguous section of memory. In this memory allocation, all the available memory space remains together in one place which implies that the freely available memory partitions are not spread over here and there across the whole

memory space.

CONTIGUOUS MEMORY ALLOCATION

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.



We will cover the concept of different Contiguous Memory allocation techniques one by one.

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.

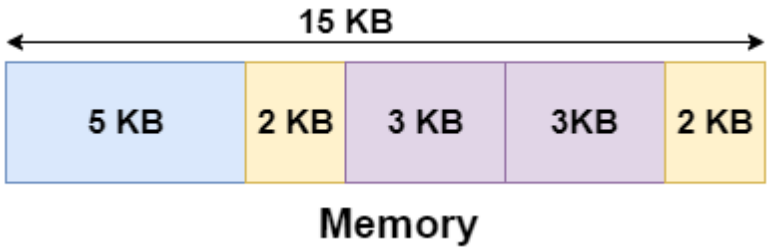
In this partition scheme, each partition may contain exactly one process. There is a problem that this technique will limit the degree of multiprogramming because the number of partitions will basically

decide the number of processes.

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

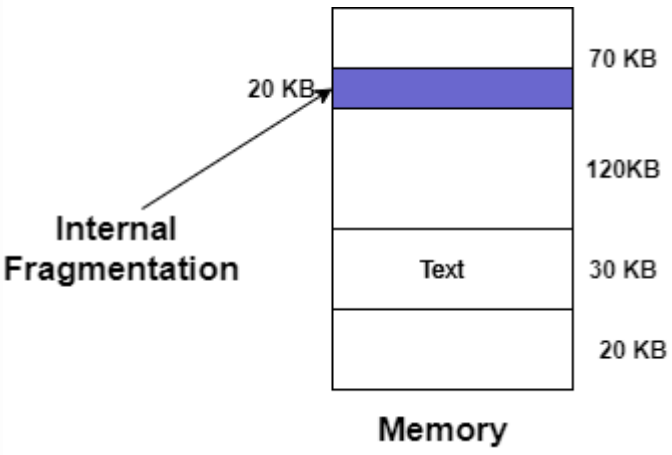
Example

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**.



Advantages of Fixed-size Partition Scheme

- This scheme is simple and is easy to implement
- It supports multiprogramming as multiple processes can be stored inside the main memory.
- Management is easy using this scheme

Disadvantages of Fixed-size Partition Scheme

Some disadvantages of using this scheme are as follows:

1. Internal Fragmentation

Suppose the size of the process is lesser than the size of the partition in that case some size of the partition gets wasted and remains unused. This wastage inside the memory is generally termed as Internal fragmentation

As we have shown in the above diagram the 70 KB partition is used to load a process of 50 KB so the remaining 20 KB got wasted.

2. **Limitation on the size of the process**

If in a case size of a process is more than that of a maximum-sized partition then that process cannot be loaded into the memory. Due to this, a condition is imposed on the size of the process and it is: the size of the process cannot be larger than the size of the largest partition.

3. External Fragmentation

It is another drawback of the fixed-size partition scheme as total unused space by various partitions cannot be used in order to load the processes even though there is the availability of space but it is not in the contiguous fashion.

4. Degree of multiprogramming is less

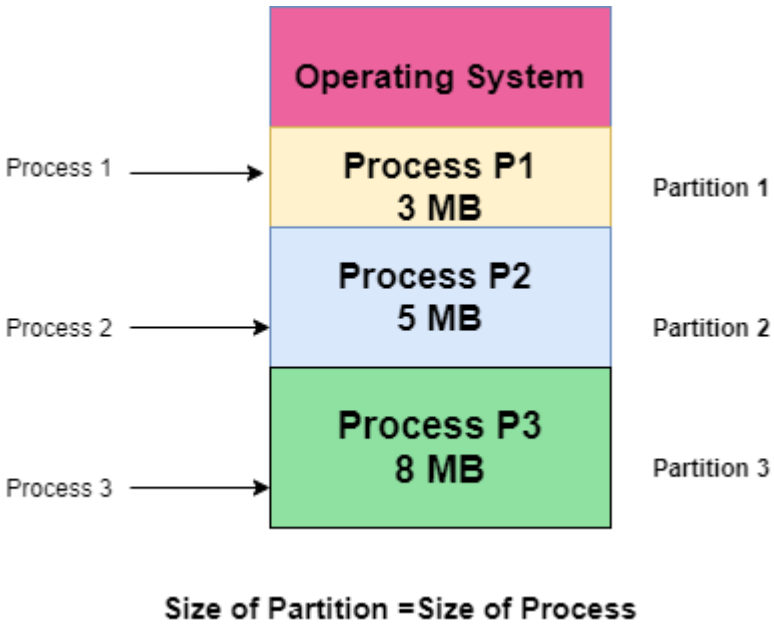
In this partition scheme, as the size of the partition cannot change according to the size of the process. Thus the degree of multiprogramming is very less and is fixed.

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 of Variable-size Partition Scheme

Some Advantages of using this partition scheme are as follows:

1. **No Internal Fragmentation** As in this partition scheme space in the main memory is allocated strictly according to the requirement of the process thus there is no chance of internal fragmentation. Also, there will be no unused space left in the partition.
2. **Degree of Multiprogramming is Dynamic** As there is no internal fragmentation in this partition scheme due to which there is no unused space in the memory. Thus more processes can be loaded into the memory at the same time.
3. **No Limitation on the Size of Process** In this partition scheme as the partition is allocated to the process dynamically thus the size of the process cannot be restricted because the partition size is decided according to the process size.

Disadvantages of Variable-size Partition Scheme

Some Disadvantages of using this partition scheme are as follows:

1. **External Fragmentation** As there is no internal fragmentation which is an advantage of using this partition scheme does not mean there will no external fragmentation. Let us understand this with the help of an example: In the above diagram- process P1(3MB) and process P3(8MB) completed their execution. Hence there are two spaces left i.e. 3MB and 8MB. Let's there is a Process P4 of size 15 MB comes. But the empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation. Because the rule says that process must be continuously present in the main memory in order to get executed. Thus it results in External Fragmentation.
2. **Difficult Implementation** The implementation of this partition scheme is difficult as compared to the Fixed Partitioning scheme as it involves the allocation of memory at run-time rather than during the system configuration. As we know that OS keeps the track of all the partitions but here allocation and deallocation are done very frequently and partition size will be changed at each time so it will be difficult for the operating system to manage everything.

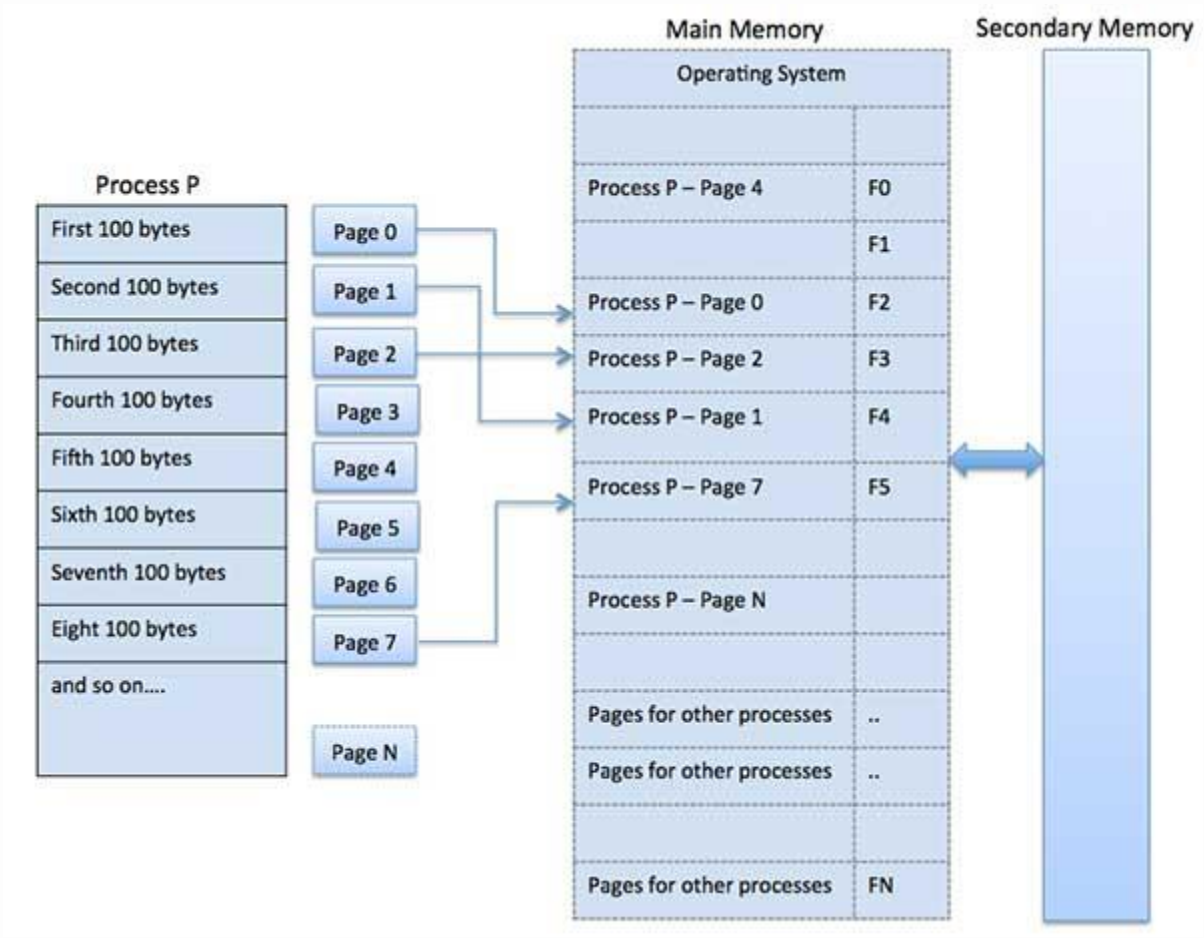
3. Discuss about paging and segmentation in detailed.

Paging:

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 that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



Address Translation

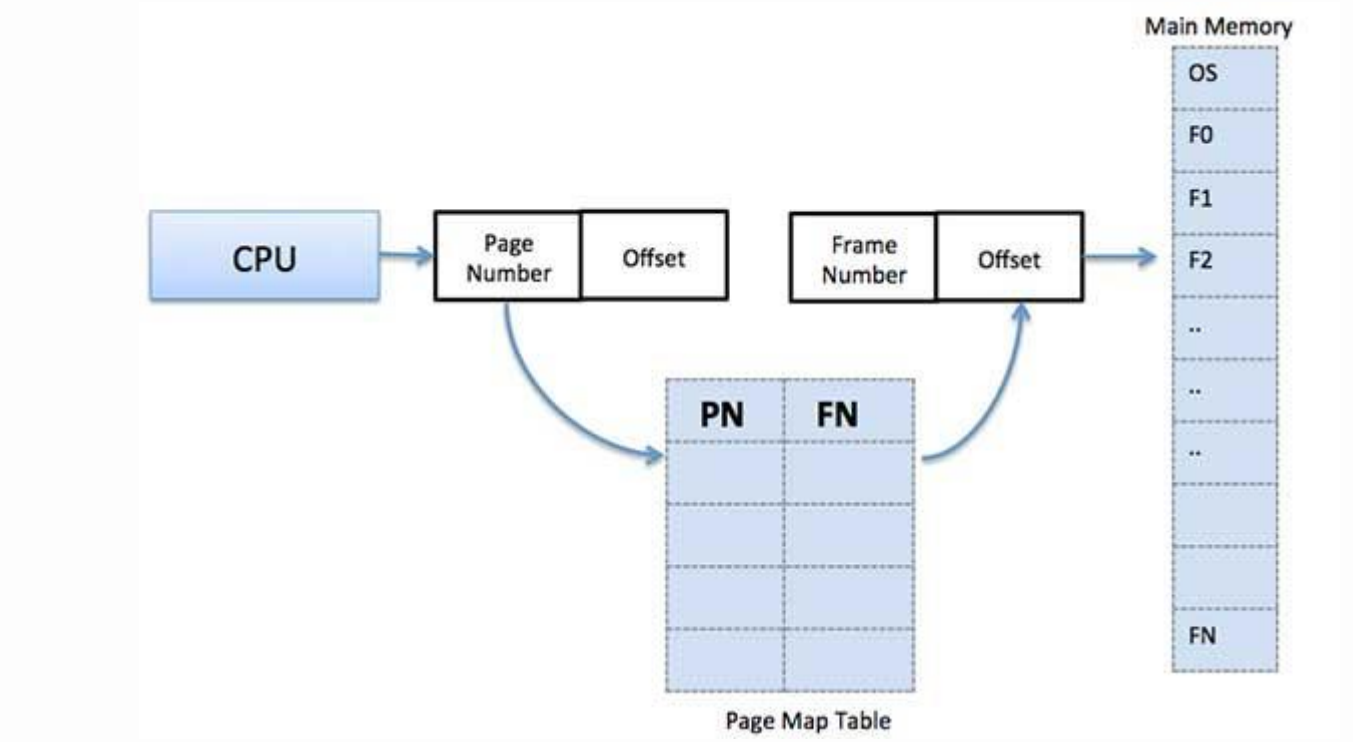
Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

Physical Address = Frame number + page offset

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.



When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Advantages and Disadvantages of Paging

Here is a list of advantages and disadvantages of paging −

- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.

- Page table requires extra memory space, so may not be good for a system having small RAM.

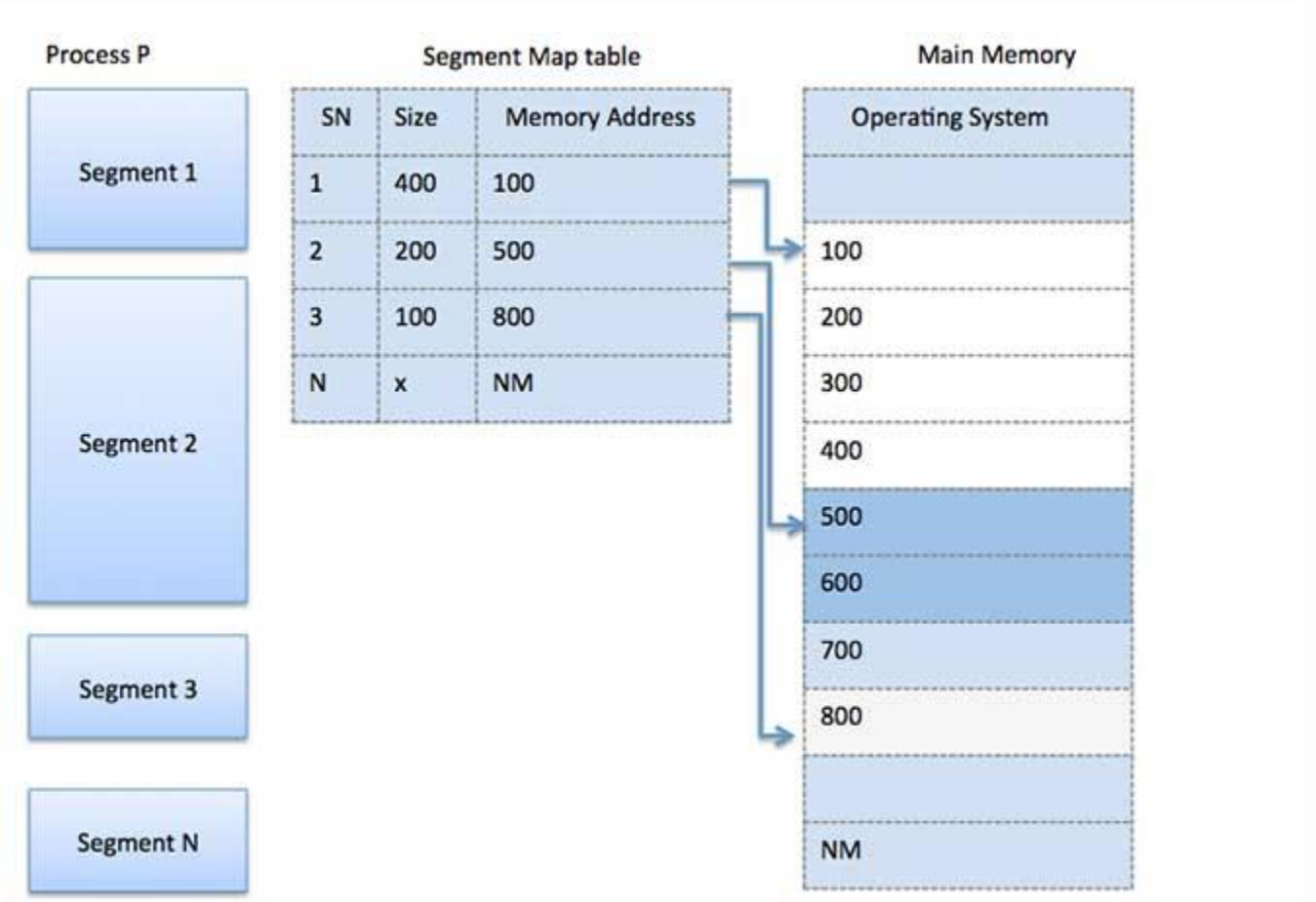
Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



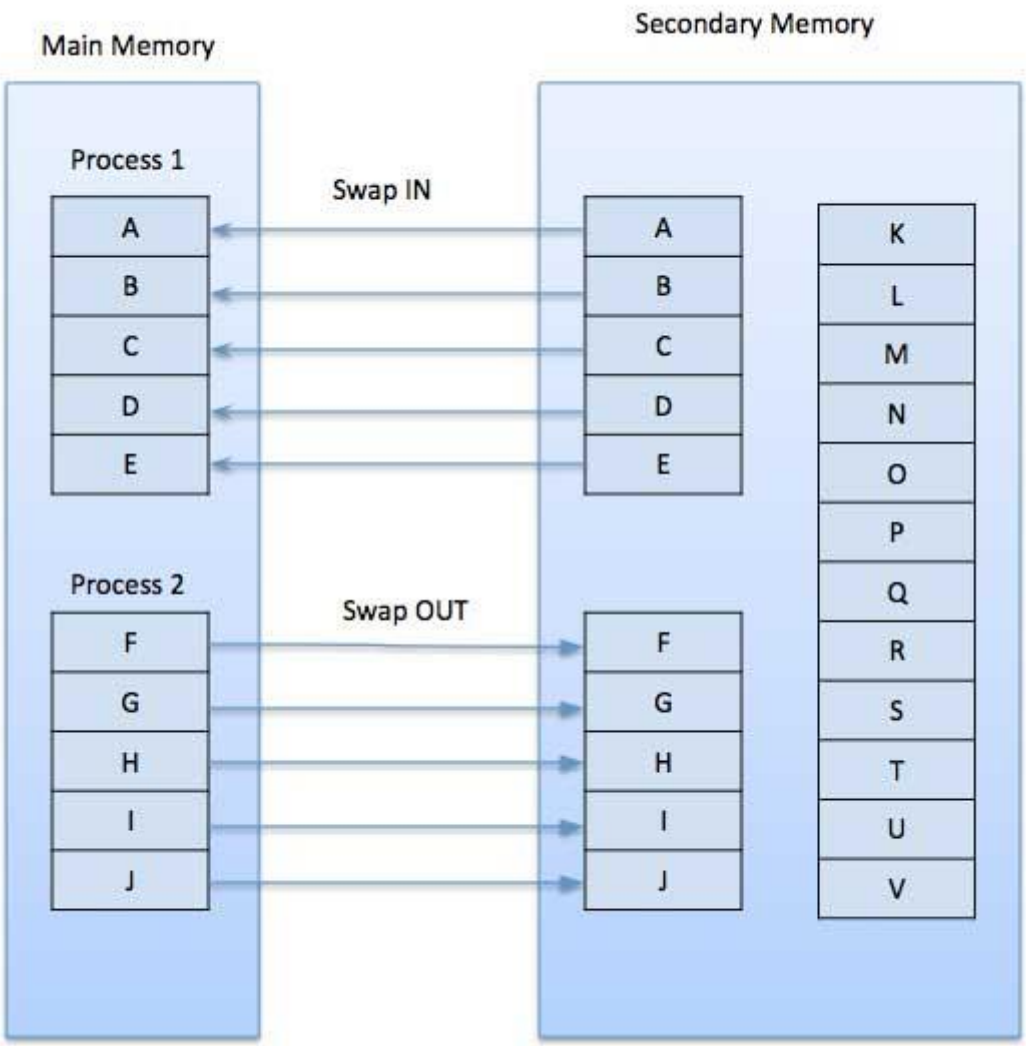
Following are the important differences between Paging and Segmentation.

Sr. No.	Key	Paging	Segmentation
1	Memory Size	In Paging, a process address space is broken into fixed sized blocks called pages.	In Segmentation, a process address space is broken in varying sized blocks called sections.
2	Accountability	Operating System divides the memory into pages.	Compiler is responsible to calculate the segment size, the virtual address and actual address.

3	Size	Page size is determined by available memory.	Section size is determined by the user.
4	Speed	Paging technique is faster in terms of memory access.	Segmentation is slower than paging.
5	Fragmentation	Paging can cause internal fragmentation as some pages may go underutilized.	Segmentation can cause external fragmentation as some memory block may not be used at all.
6	Logical Address	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
7	Table	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical address is divided into section number and section offset.
8	Data Storage	Page table stores the page data.	Segmentation table stores the segmentation data.

4. Explain about Demand paging with example.

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

Advantages

Following are the advantages of Demand Paging −

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques

5. Explain various page replacement algorithms with examples.

Page Replacement Algorithms

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

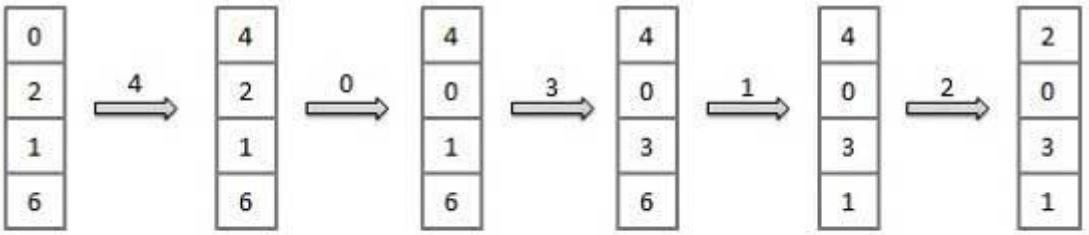
- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses − 123,215,600,1234,76,96
- If page size is 100, then the reference string is 1,2,6,12,0,0

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x x



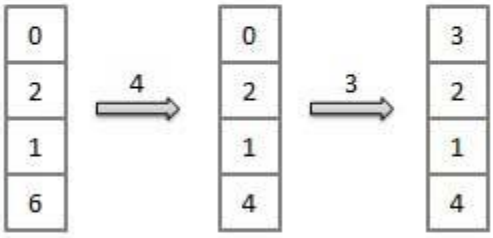
Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x



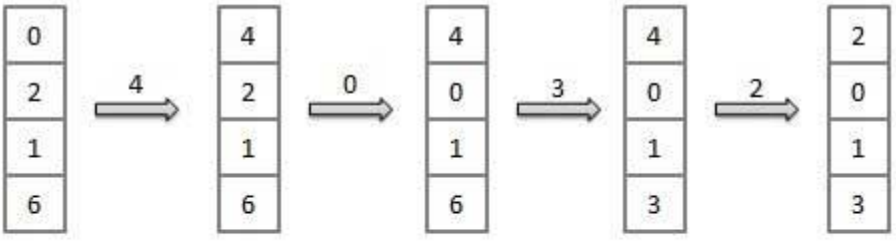
Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used(MFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

UNIT-V

1. Discuss about the concepts of file in detailed.

File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.

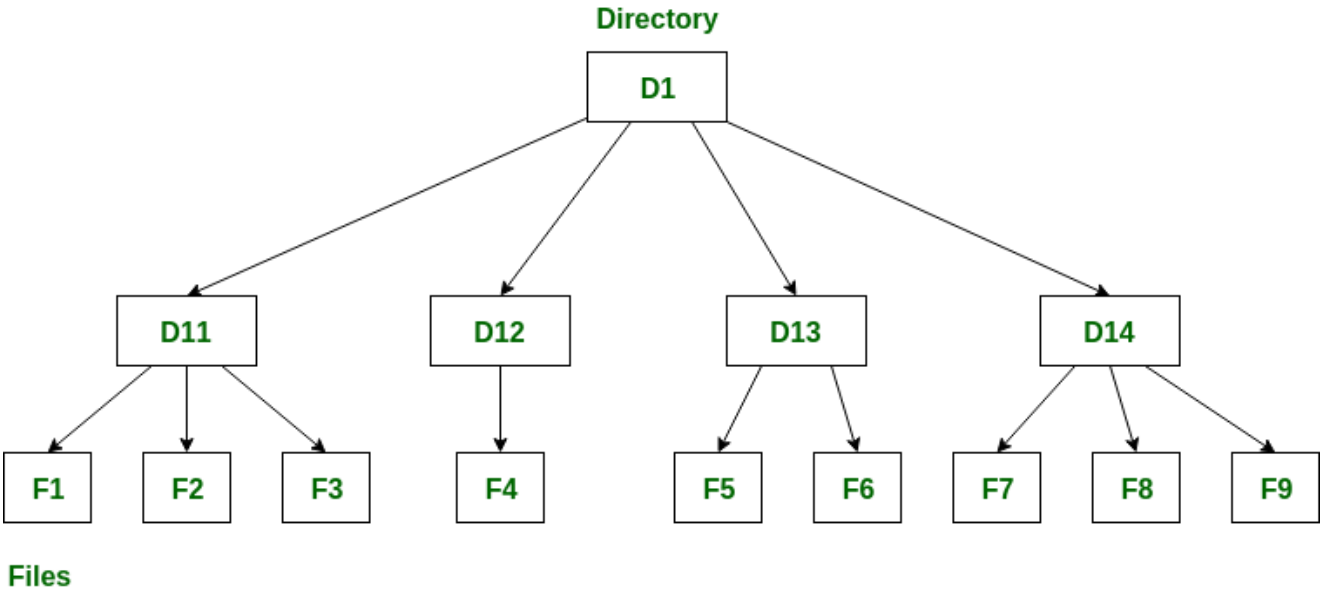
Indexed Allocation

- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.

- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files.

2 .Explain about Directory structure and its types.

A **directory** is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner.

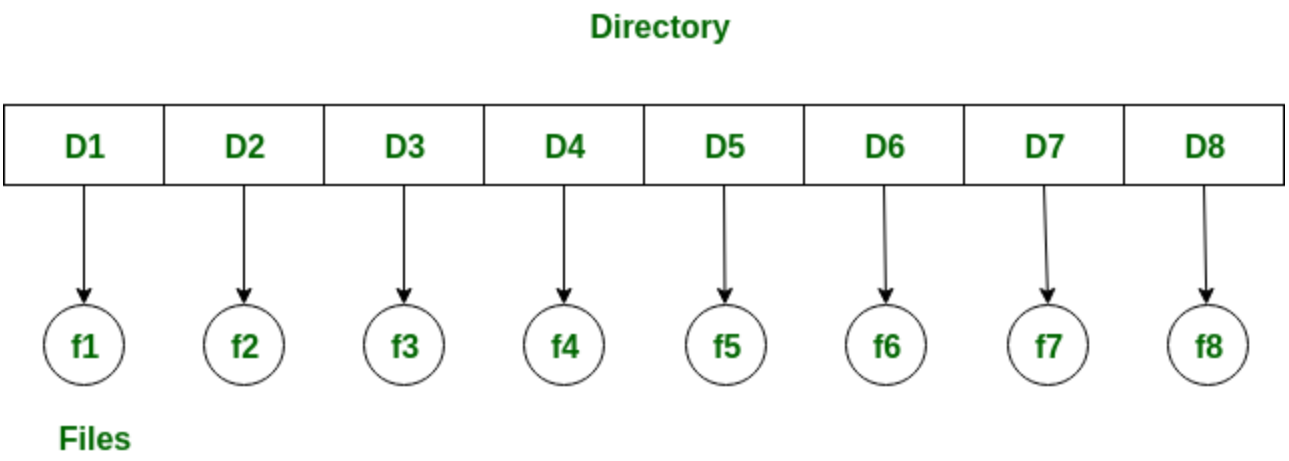


There are several logical structures of a directory, these are given below.

- **Single-level-directory** –

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

  A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a unique name. if two users call their dataset test, then the unique name rule violated.



Advantages:

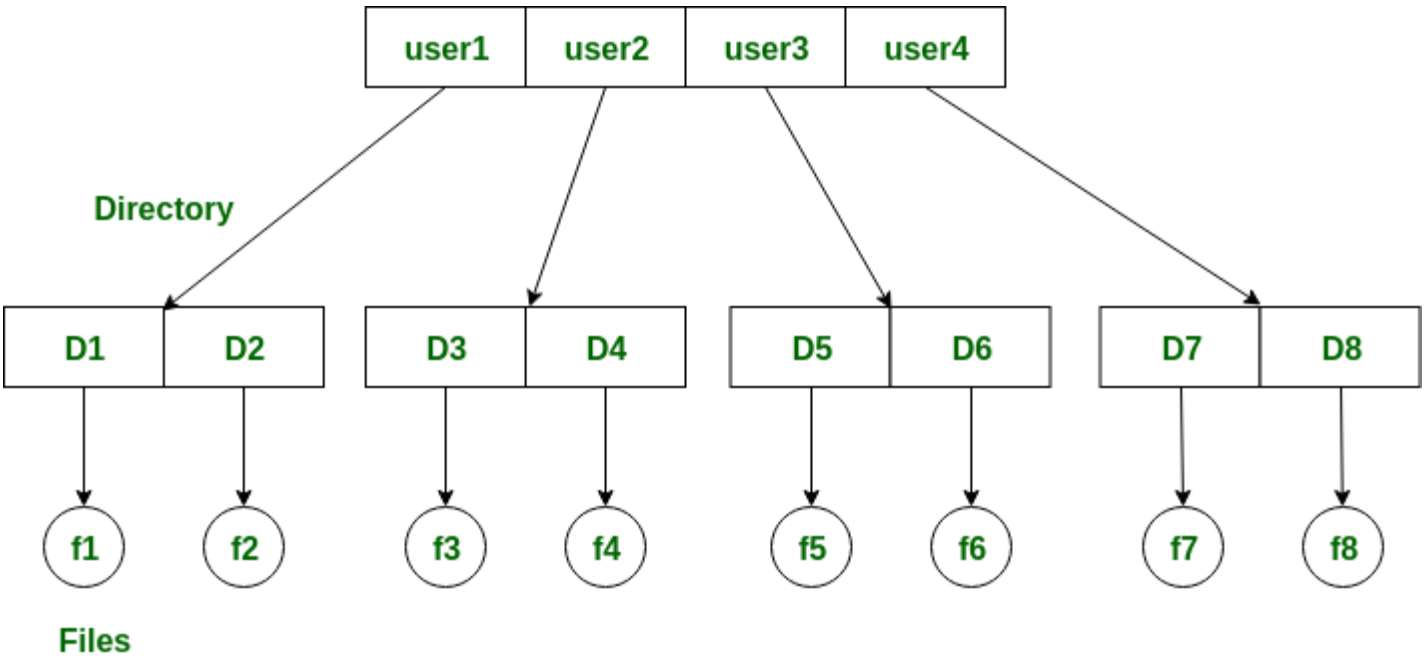
- Since it is a single directory, so its implementation is very easy.
- If the files are smaller in size, searching will become faster.
- The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

Disadvantages:

- There may chance of name collision because two files can not have the same name.
- Searching will become time taking if the directory is large.
- This can not group the same type of files together.
- **Two-level-directory** –

  As we have seen, a single level directory often leads to confusion of files names among different users. the solution to this problem is to create a separate directory for each user.

  In the two-level directory structure, each user has their own *user files directory (UFD)*. The UFDs have similar structures, but each lists only the files of a single user. system's *master file directory (MFD)* is searches whenever a new user id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.



Advantages:

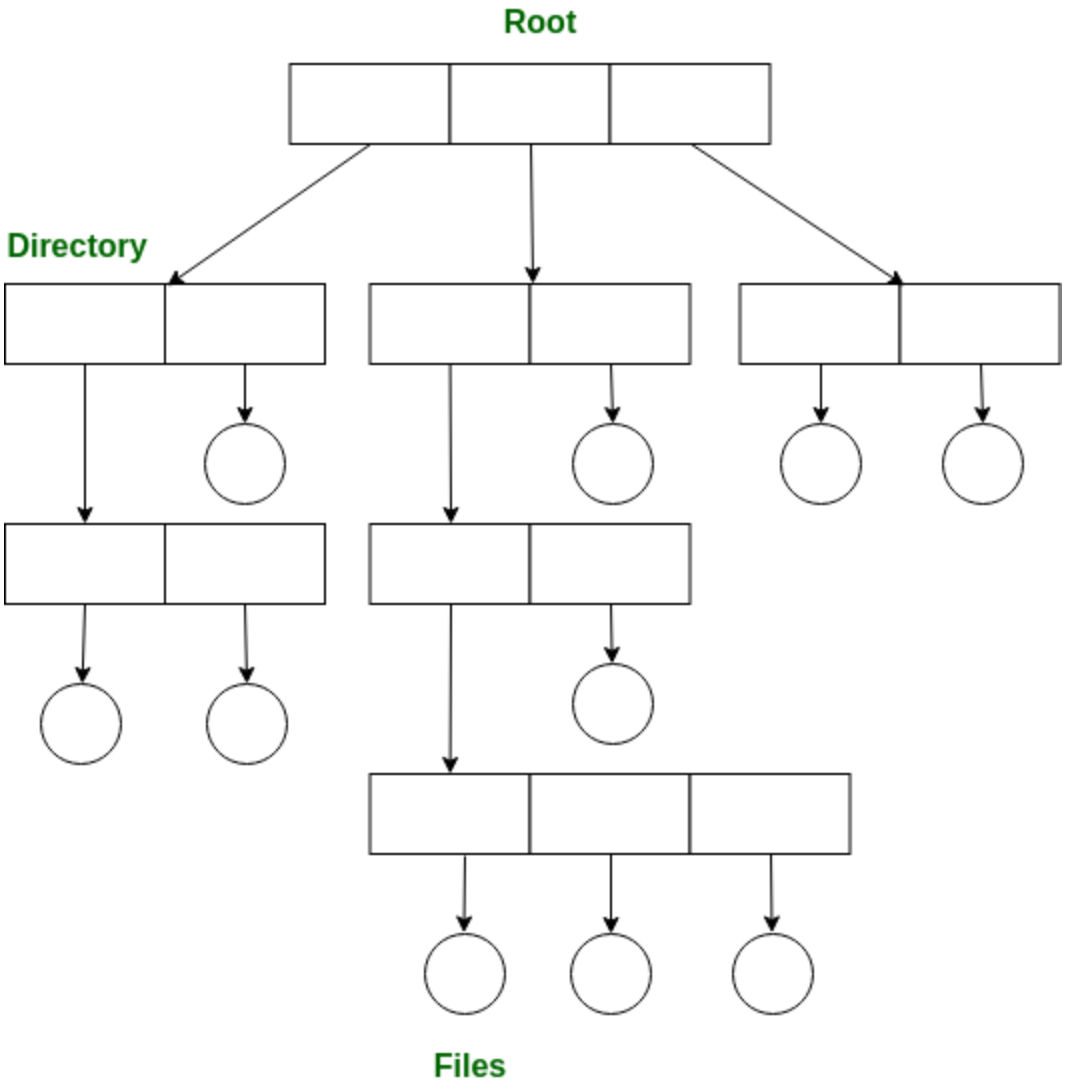
- We can give full path like /User-name/directory-name/.
- Different users can have the same directory as well as the file name.
- Searching of files becomes easier due to pathname and user-grouping.

Disadvantages:

- A user is not allowed to share files with other users.
- Still, it not very scalable, two files of the same type cannot be grouped together in the sameuser.

- **Tree-structured-directory** –

  Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height. This generalization allows the user to create their own subdirectories and to organize their files accordingly.



A tree structure is the most common directory structure. The tree has a root directory, and every file in the system has a unique path.

Advantages:

- Very general, since full pathname can be given.
- Very scalable, the probability of name collision is less.
- Searching becomes very easy, we can use both absolute paths as well as relative.

Disadvantages:

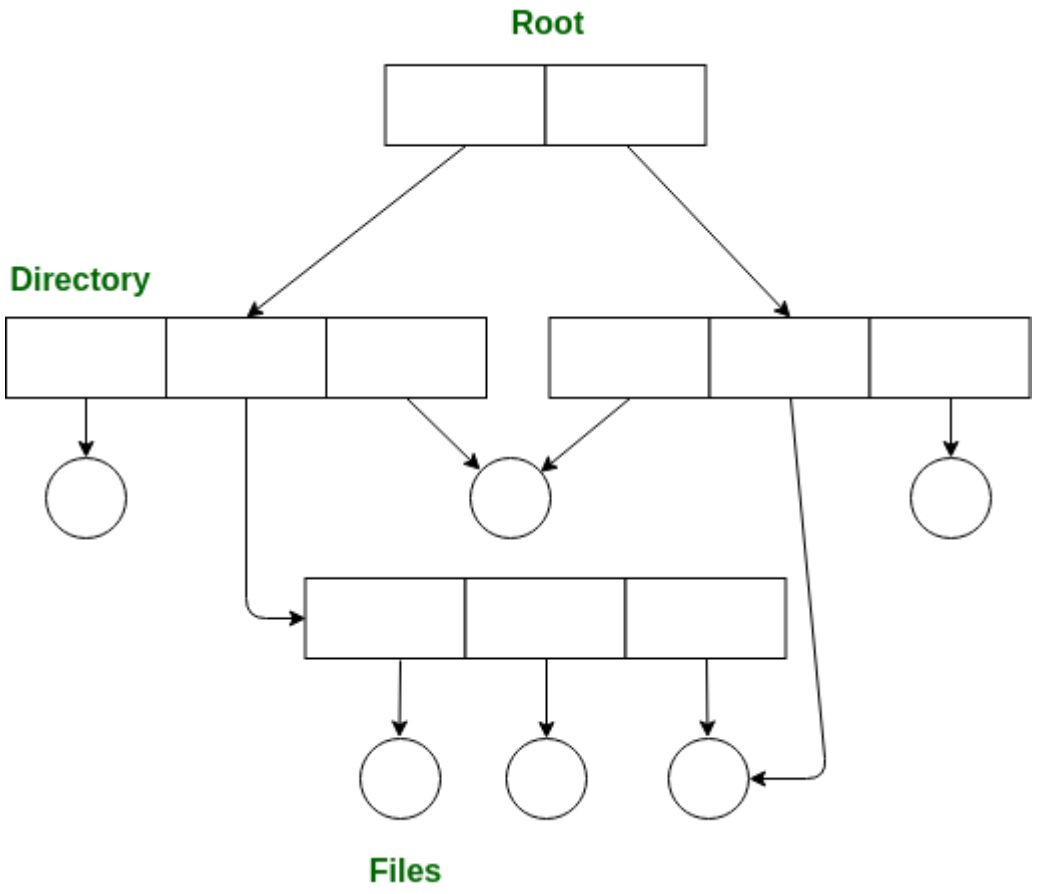
- Every file does not fit into the hierarchical model, files may be saved into multiple directories.
- We can not share files.
- It is inefficient, because accessing a file may go under multiple directories.

- **Acyclic-graph-directory** –

An acyclic graph is a graph with no cycle and allows us to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.

It is used in the situation like when two programmers are working on a joint project and they need to access files. The associated files are stored in a subdirectory, separating them from other projects and files of other programmers since they are working on a joint project so they want the subdirectories to be into their own directories. The common subdirectories should be shared. So here we use Acyclic directories.

It is the point to note that the shared file is not the same as the copy file. If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.



Advantages:

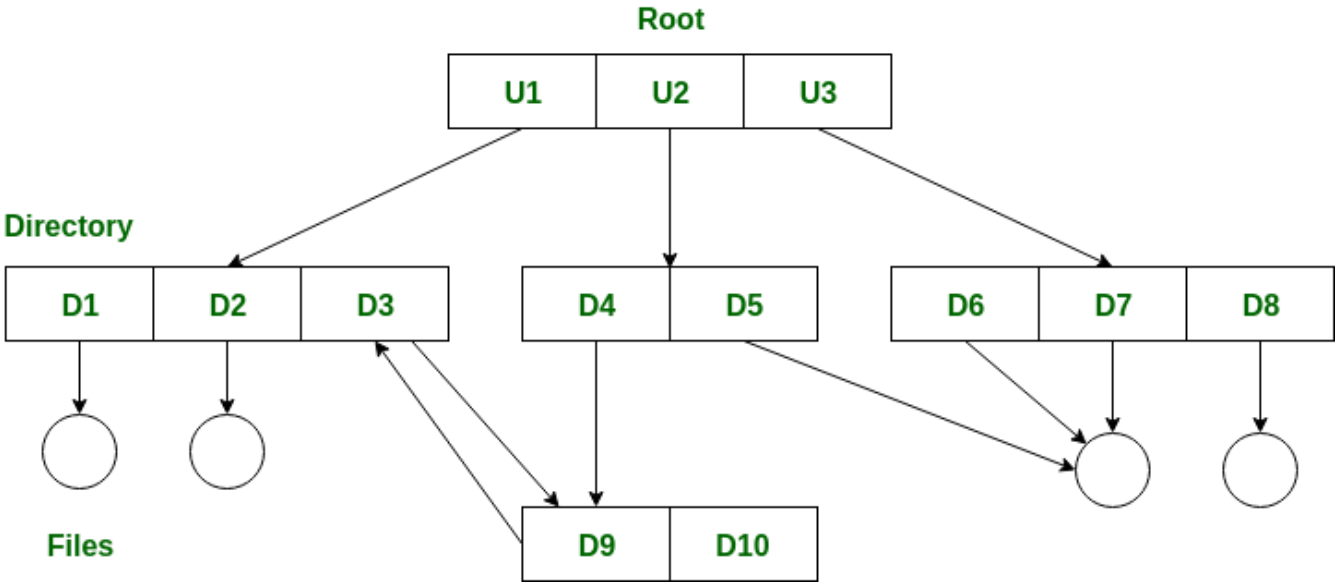
- We can share files.
- Searching is easy due to different-different paths.

Disadvantages:

- We share the files via linking, in case deleting it may create the problem,
- If the link is a soft link then after deleting the file we left with a dangling pointer.

In the case of a hard link, to delete a file we have to delete all the references associated with it.

- **General graph directory structure –**
  In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.
  The main problem with this kind of directory structure is to calculate the total size or space that has been taken by the files and directories.



Advantages:
- It allows cycles.
- It is more flexible than other directories structure.

Disadvantages:
- It is more costly than others.
- It needs garbage collection.

3. Write about protection and security in file system interface.

Protection is one of the major role in the OS and file system interface. The protection can be provided in many ways. For example, these are represented as follows.

Types of access: it provides a LIMITEED FILE ACCESS TO THE USER. Several types of operations may be controlled by file system interface is as follows • Read: read from the file • Write: write or rewrite the file • Execute: load the file into memory and execute • Append: write the new information at the end of the file • Delete: delete the file and releases the memory • List: list the name and attributes of the file Access list and groups: In the file system interface, every file may requires 3 persons. They are 1.owner 2.group 3.others Owner – the user who created the file is the owner Group – a set of users who are sharing the file with in the work group Universe/others – all other users in the file system interface

In the MS-Dos operating system the access rights are applied though “attriB command is as follows C:\\ Attrib +751<fIle name> In the UNIX operating system the access rightes are applied though “chmod” command is as follows

Chmod +751<file name>

➔The attributes read contains 4,write contains 2,and execute contains 1 ➔Hence in the above example the owner contains read and execute permission, group contains read and execute permissions and others contain execute permission only.

Security:

Security requires not only an adequate protection system, but also consideration of the external environment within which the system operates. The system protect it from

i) unauthorized access. ii) Malicious (harmful) modification or destruction iii) accidental introduction of inconsistency.

It is easier to protect against accidental than malicious misuse.

Authentication:

➔A major security problem for operating systems is authentication. Generally, authentication is based on one or more of three items: user possession (a key or card), user knowledge (a user_id and password) or user attributes (fingerprint, retina pattern or signature). ➔¨ Authentication means Verifying the identity of another entity – Computer authenticating to another computer

– Person authenticating to a local computer – Person authenticating to a remote computer

Passwords:

➔ The most common authentication of a user’s identity is via a user password. ➔ In password authentication, users are authenticated by the password. - User has a secret password. - System checks it to authenticate the user. - Vulnerable to eavesdropping when password is communicated from user to system. - The big problem with password-based authentication is eavesdropping. Vulnerabilities : 1) External Disclosure – Password becomes known to an unauthorized person by a means outside normal

network or system operation. Includes storing passwords in unprotected files or posting it in an unsecured place.

2) Password Guessing – Results from very short passwords, not changing passwords often, allowing passwords which are common words or proper names, and using default passwords. 3) Live Eavesdropping – Results from allowing passwords to transmit in an unprotected manner.

4. Discuss about file system structure.

• The functionality of the file is used to store large amount of information.

• In the file system implementation, the layered approach is represented as follows.

Application programs

1. Logical file system
2. File organisation module
3. Basic file system
4. I/O control | Devices

➢The lowest level contains the I/O control consists of device drivers and interrupt handlers to transfer the information between the memory and disk system.

➢The file organisation module defines the information about the files, logical blocks as well as the physical blocks. It also include free space manager.

➢The logical file system uses the directory structure to provide the file organisation module. It is also responsible for protection and security.

➢Application programs are written by the user with the help of languages like c ,c++,java,.., e.t.c. in the application programs, they are various types of files. For example, they are text files, binary files, indexed files, random files, executable files, e.t.c..

5.List out various operations in files. Explain it briefly.

1.Create

Creation of the file is the most important operation on the file. Different types of files are created by different methods for example text editors are used to create a text file, word processors are used to create a word file and Image editors are used to create the image files.

2.Write

Writing the file is different from creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the data needs to be written.

3.Read

Every file is opened in three different modes : Read, Write and append. A Read pointer is maintained by the OS, pointing to the position up to which, the data has been read.

4.Re-position

Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as seeking.

5.Delete

Deleting the file will not only delete all the data stored inside the file, It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.

6.Truncate

Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file get replaced.

UNIT WISE UNIVERSITY PREVIOUS QUESTION PAPER QUESTIONS

UNIT –I PREVIOUSLY ASKED ESSAY QUESTIONS

PART –C
PREVIOUS YEAR QUESTION PAPERS
UNIT-I

1. What is operating systems?. Explain its various functions?
2. Explain in detailed about role of operating systems as resource manager.
3. What are the different types of os?. Explain them in detail.
4. Discuss about Distributed systems and various failures in Distributed systems.
5.What are the various characteristics of real time systems?
6. Discuss the essential properties of Time-Shared and Distributed systems.
7. Write short notes on different types of system calls.
8.Compare multiprogramming and multitasking.
9. What are the various services provide by an operating systems?
10. Discuss different level of view of operating systems.

UNIT-II

1.Define process. How many different states a process has?. Explain it.
2. Discuss about thread creation. What are the various types of threads?
3.Explain about Inter Process Communication and its roles.
4. Discuss about RR scheduling with own example.
5. What are the various system calls for process management?
6. Discuss about various operations on process.
7. What are the different types of schedulers? Explain each of them.
8. Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	2
P2	3	1
P3	5	6

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

9. Consider the set of 6 processes whose arrival time and burst time are given below

Process Id	Arrival time	Burst time
P1	5	5
P2	4	6
P3	3	7
P4	1	9
P5	2	2
P6	6	3

If the CPU scheduling policy is Round Robin with time quantum = 3, calculate the average waiting time and average turn around time.

10 Discuss multiple-processor scheduling.

UNIT-III

1. What is deadlock? Write the characterization of deadlock.

2. Explain Semaphore and its types.

3. Discuss about Deadlock prevention and its importance.

4. A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST?

1. P0
2. P1

3. P2

4.None of the above since the system is in a deadlock

	Alloc			Request		
	X	**Y**	**Z**	**X**	**Y**	**Z**
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

5.Discuss about the uses of RAG and its components.

6. Explain how Dining-Philosopher problem solved by using monitors?

7. Discuss abou IPC mechanism with its various stages.

8. What is pipes? Explain properties and limitations of pipes.

9. What are the various issues in IPC by message passing?

UNIT-IV

1. What do you mean by internal and external fragmentation?

2. What are the disadvantages of single continuous memory allocation?

3. Consider a swapping system which memory consist of the following hole sizes in memory order **10kB, 4kB, 20kB,18kB, 7kB,9kB,12kB and 15kB**.which hole is taken for successive segment request of 1. 12kB 2. 10kB 3. 9kB for first fit, bestfit and worstfit approaches.

4. Explain the importance of memory protection in multiprogramming systems.

5. Define segmentation and its importance.

6. Explain the common technique for structuring page table.

7. Write a detailed note on virtual memory**.**

8. A system uses 3 page frames for storing process pages in main memory. It uses the First in First out (FIFO) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below-

4 , 7, 6, 1, 7, 6, 1, 2, 7, 2

Also calculate the hit ratio and miss ratio.

9. A system uses 3 page frames for storing process pages in main memory. It uses the Optimal page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below-

4 , 7, 6, 1, 7, 6, 1, 2, 7, 2

Also calculate the hit ratio and miss ratio.

10. Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3. Find out the number of page faults respective to:

1. Optimal Page Replacement Algorithm
2. FIFO Page Replacement Algorithm
3. LRU Page Replacement Algorithm

11. Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. which one of the following is true with respect to page replacement policies First-In-First-out (FIFO) and Least Recently Used (LRU)?

A. Both incur the same number of page faults

B. FIFO incurs 2 more page faults than LRU

C. LRU incurs 2 more page faults than FIFO

D. FIFO incurs 1 more page faults than LRU

UNIT-V

1. Explain various directory structure and its uses.
2. What is meant by file? What is its access method of file?
3. Discuss about file allocation table.
4. Distinguish between shared and exclusive locks.
5. Explain various operations in files.
6. What is free space management?
7. Explain the importance of file protection.
8. Write a syntax of create and open system call of file.