Notes

Operating System :- Section – A

- Operating System is a collection of set of programs which manages all the resources of the computer system.
- It is an intermediator between the user and hardware.

User ↔ O.S ↔ Compiler

Advantages

1. Operating System provides a GUI interface for the users in the form of menu, icons and buttons.
2. Operating System allow us to sharing the resources with other users.
3. It helps user to understand the functions of a computer.
4. It is very easy to use.
5. It can be easily updated.

Disadvantages

1. If the operating system is corrupted then it will affect entire system & the computer system will not work.
2. Only same task run at a time.

Types of Operating System

1. Batch Operating System :-

- The batch operating system will work to submit similar kinds of jobs together.
- In this operating system, user do not interact directly with own computer system.

Advantages

- It is useful when we working with large files which can take more time to execute.

Disadvantages

- Once the job is submitted, the user did not have any interaction with it.

2. Time Sharing Operating System :-

- The time sharing operating system works with time sharing concept.
- Here the CPU will provide a same time period to each and every process to complete its task as soon as possible whether it is a long process or short process.

Advantages

- If the process of a task is completed then the time between the other tasks increased.

Disadvantages

- Sometimes in this operating system there may be a problem of data communication.

3. Distributed Operating System :-

- When many computers are interconnected to each other through a network for the purpose of sharing their task then it is called distributed operating system.

Advantages

- If a node is over used, the distributed operating system shares that load to other node of a network.

Disadvantage

- If the network is spread, communication of all computers is broken.

4. Network Operating System :-

- Network operating system have a server that connects many other client computers.
- So, we can easily share our files, resources and may move from the server machine to all the machine which one is connected to a server.

Advantage

- Network operating system provides a security & we need to upgrade only server machine.

Disadvantage

- If the server machine is crashed then automatically the entire network will be failed.

---

5. Real Time Operating System :-

- Real time operating system is very useful where we required a quick response.
- For example – missile systems, where the missile has to be launched at certain time, hospital etc.
- In this operating system CPU provides maximum offers to its task with quick response.

Advantages

- Real time operating system provide a quick response hence, generally used in Scientific engineering NASA & many more organisation.

Disadvantage

- These operating system is very costly.

Types

1. Hard Real time OS
2. Soft Real time OS

---

6. Simple Batch Operating System :-

- A simple batch operating system is one of the earliest types of operating systems.
- It was developed to improve CPU utilization by reducing idle time.
- In this system jobs are grouped into batches and executed one after another without any user interaction during execution.

Working of Simple batch operating system

- In a simple batch OS, users submits their programs (jobs) along with data to the computer operator.
- These jobs are collected and stored on secondary storage usually in the form of a batch.
- The OS then selects jobs from the batch and executes them sequentially.
- Once a job starts executing, it runs to completion without interruption.
- After the job finishes, the next job from the batch is loaded and executed. Users receive the output only after the entire job execution is completed.

---

7. Parallel Operating System

- A parallel operating system is designed to work with multiple processors that operate simultaneously.
- The main objective of this system is to increase computational speed, throughput, reliability by executing tasks in parallel.

Working of Parallel Operating System

- In a parallel operating system, a single task or multiple tasks are divided into smaller subtasks.
- These subtasks are distributed among multiple processors, which execute them simultaneously.
- The operating system coordinates and synchronizes the processors to ensure correct execution.
- Parallel systems may use shared memory architecture, where all processors share a common memory, or distributed memory architecture, where each processor has its own local memory.

System Calls

- System calls are the mechanism through which a user program requests services from the operating system.
- Since user programs cannot directly access hardware or critical system resources, system calls provide a controlled and safe interface between user level processes and the kernel of the operating system.

Working

- When a user program needs a service such as reading a file or creating a process, it invokes a system call. The execution of a system call involves the following steps:
- First the user program issues a system call request.
- Control of the CPU is then transferred from user mode to kernel mode, allowing the operating system to perform the requested operation securely.
- The operating system executes the required service, such as accessing a file or allocating memory.
- After completion, control is returned back to the user program, and execution continues in user mode.

Types

1. File Manipulation System Calls :-

- File manipulation system calls are used to manage files and directories stored on secondary storage.
- Through these calls, a user program can open an existing file, create a new file, read data from a file, write data to a file, or close a file after use.

- The operating system performs all disks related operations internally, ensuring data consistency and protection.

---

2. Process Control System Calls :-

- Process control system calls are used to manage the execution of processes.
- These calls allow the creation of new processes and termination of existing ones.
- They also support loading and executing programs, allocating memory to processes, and synchronizing processes using wait and signal operation.

---

3. Device management System calls :-

- Device management system calls enable interaction with input / output devices.
- Using these calls, a process can request or release a device, perform read or write operations, and obtain or modify device attributes.

---

4. Communication System calls :-

- Communication system calls are used for inter process communication.
- These calls allow processes to send and receive message, establish or remove communication links and exchange data.

Services of Operating System :-

1. Program Execution

- The OS provides an environment in which users can execute programs.
- When a user runs a program, the OS loads the program into memory, allocates necessary resources such as CPU and memory, and starts execution.
- After execution is complete, the OS ensures proper transition of the program and releases the allocated resources. Users are not required to manage these low level details.

---

2. I/O Operations

- Programs frequently require input from input devices or output to output devices. The operating system handles all I/O operations on behalf of users.

- Users cannot directly access hardware devices; instead the OS performs all I/O operations using system calls and device drivers. This ensures safety, consistency and device interdependence.

---

3. File System Manipulation

- The OS provides services for managing files and directories. Users can create, delete, read, write and modify files without worrying about how data is stored on disk.
- The OS also manages directory structures and ensures protection and access control for files.

---

4. Communication

- Processes often need to exchange data with one another. The operating system provides communication mechanisms such as message passing and shared memory.
- This service is essential for multitasking and distributed systems.

---

5. Errors detection

- Errors may occur in CPU, memory, I/O devices, or user programs. The operating system continually monitors the system to detect such errors.
- When an error is detected, the OS takes appropriate action to ensure correct and consistent system operation.

---

6. Accounting

- The OS keeps track of how much and which type of resources each user or process uses.
- This information is useful for billing, system optimization, and performance evaluation.

---

7. Protection & Security

- The OS provides services to control access to system resources and protect data from unauthorized access. This includes password protection, access control, and encryption.

Components of OS :-

1. Process Management

- Process management is one of the most important components of an operating system.
- A process is a program in execution. Since multiple programs may run simultaneously in a multiprogramming environment, the OS must manage these processes efficiently.

- Process management also includes CPU scheduling, where the operating system decides which process should get the CPU and for how long.
- To ensure correct execution, the OS provides process synchronization mechanisms such as semaphores and mutexes.
- Additionally, the OS handles interprocess communication and provides mechanisms to detect and resolve deadlocks, where processes wait indefinitely for resources.
- Thus process management ensures efficient CPU utilization and smooth execution of multiple processes.

---

2. Main Memory management :-

- Main memory management is responsible for managing the system primary memory (RAM).
- Memory is a critical resource because all programs and data must be loaded into main memory before execution.
- The operating system keeps tracks of which parts of memory are currently in use and which are free.
- When a new process is created, the OS decides where to load the process in memory and allocated the required memory space. When the process finishes execution, the OS deallocates the memory so it can be reused.
- In a multiprogramming environment, several processes may reside in memory at the same time.
- Therefore, the operating system must ensure memory protection so that one process do not access the memory of another process itself.

---

3. Secondary Storage Management :-

- Main memory is limited in size and volatile in nature. Therefore, the OS uses secondary storage devices such as hard disks and SSDs for permanent data storage.
- Secondary storage management involves managing disk space efficiently. The OS performs free space management to keep track of unused disk blocks.
- It also handles storage allocation, deciding where to store files and programs on the disk.
- Another important function is disk scheduling where the OS decides the order in which disk I/O requests are serviced to minimize seek time and improve performance.

---

4. File Management :-

- File management is responsible for handling files, which are collections of related information stored on secondary storage.
- The OS provides a logical view of files so that users do not need to know how data is physically stored on the disk.
- The OS allows users to create, delete, read, write, and modify files. It also supports directory structures which helps organize files for easy access.
- File management includes mapping files onto disk blocks and maintaining file metadata such as size, type, and access permissions.
- Additionally, the OS provides backup and recovery mechanism to protect files against accidental loss.

Protection and Security :-

- Protection and security are essential components of an operating system, especially in multi user environments.
- Protection ensures that processes do not interfere with each other or with the OS.
- The OS controls access to system resources such as CPU, memory, files and devices. It ensures that only authorized users and processes can access certain resources.
- Security mechanism include authentication (user login), authorization, and protection against unauthorized access.
- The OS also safeguards the system from both internal threats and external threats.

Process Management Section - A

- A process is the instance of a computer program that is being executed by one or many threads.
- In the operating system, a process is something that is currently under execution. So an active program can be called a process.
- In most systems, a process is the unit of work done.
- A process needs contain resources such as CPU time, memory, files, and I/O devices to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

Abstract Machine Environment (OS)

For example :-
When you want to search something on web then you start a browser. So this can be a process.

⇒ Process States / Life Cycle of Process

- During the execution of a process, it undergoes a no. of states.
- These steps may differ in different operating systems and the names of these stages are also not standardized.
- As a process executes it changes state. The state of a process is defined by the current activity of that process.

1. New :-

- A process that has just been created but has not get been admitted to the pool of executable processes by the operating system.

2. Ready :-

- After the creation of the process, when the process is ready for its execution then it goes in the ready state.
- In a ready state, the process is ready for its execution by the CPU but is waiting for its turn to come.

3. Running :-

- The process whose instructions are being executed is called running process. A running process posseses all resources needed for its execution including the processor.
- A running process may go into blocked state or ready state or terminated state.

4. Waiting / Blocked :-

- The process is waiting for some event to occur or is blocked until some event occurs such as the completion of I/O operation.
- Such a process cannot execute even if a CPU is available.

5. Terminated :-

- The process has finished its execution. All the tasks in a process are completed.
- After the complete execution of the process, the process comes into the terminated state and the information related to their process is deleted.

⇒ Operations on Processes :-

- The execution of a program is called process. The operating system manages these process and perform several operations on them.

1. Process Creation :-

The first operation on a process is process creation. A new process is create when:

1. When a system boot up
2. A user start a program (opening application)
3. A running process created a child process.

- The newly created process get it own process control block (PCB).

---

2. Process Termination / Destroy a process :-

- Process termination occur when a process finishes its execution or is explicitly killed.
- There are two main reasons for termination.

1. Normal completion :-

- The process finishes its task and exist cleanly.

2. Forced termination :-

- The operating system or user forcefully kills the process if its misbehaving or hangs.

⇒ Upon termination, the OS deallocates the resources used by the process and removes it process control block.

---

3. Process Scheduling :-

- The operating system uses process scheduling to determine which process run on the CPU at the given time.
- The operating system use scheduling algorithm like Round Robin, priority scheduling.

Process scheduling consists of the following sub functions :-

1. Scheduling :-

- Selecting the process to be executed next on CPU is called scheduling. In this function a process is taken out from a pool of ready processes and is assigned to CPU.
- This task is done by a component of OS called scheduler.

2. Dispatching :-

- Setting up the execution of the selected process on the CPU is called dispatching.
- It is done by a component of OS called dispatcher. Thus a dispatcher is a program responsible for assigning the CPU to the process, that has been selected by the scheduler.

3. Context Save :-

- Saving the status of a running process when its execution is to be suspended is known as context save.

---

4. Process Synchronization :-

- Process synchronization ensure that multiple processes executes in a way that they don’t interfere each other.

Example :- In a banking system, two process updating the same bank amount balance need to be synchronized to avoid inconsistent result.

---

5. Inter Process Communication :-

- Some time process need to communicate with each other and this where Inter process communication come.
- IPC allows process to share data and coordinate their action.

Shared memory :- multiple processes share the same memory segment to communicate.

Message passing :- process send & receive message via mechanism like message queue and pipes.

Process Suspension and Resumption :-

- A process can be suspended temporarily if it’s not ready to continue, usually because it’s waiting for some event (I/O) etc.
- Once the event occur, the process can be resumed and placed back in ready state to continue execution.

⇒ Threads :-

- The thread is the smallest unit of execution within a process.
- A thread is a light weight process and the smallest unit of CPU utilization.
- Thus thread is like a little mini process.
- Each thread has a thread id, a program counter, a register set and a stack.
- A process can have single thread or multithreads.

Example :-
Process – Hotel
Thread – Waiters

⇒ Complete Hotel is manage by waiters

- Each thread share the code section, Data Section, OS resource with the others and process.

⇒ Multithreading :-

- A process can have single thread of control or multiple threads of control.
- If a process has a single thread of control, it can perform only one task at a time.
- If a process has multiple thread of control, it can perform multiple tasks at a time.

⇒ Types of Threads :-

1. User level threads :-

- User level thread is created by user, not by system calls.
- The kernels has no work in management of user level threads.
- Switching between threads is very fast.

Problem :- If one thread is block, the whole process blocks.
Example :- Java green threads

Advantages

1. User level threads are fast in creation as kernel intervention is not required.
2. Low overhead as no kernel data structures are required for thread management.
3. Portable across different operating systems since thread management is done in user space.
4. Efficient for applications with frequent thread switching.
5. Custom scheduling is possible applications can implement their own scheduling algorithms.

Disadvantages

1. Less suitable for modern multi core systems.
2. Poor performance for I/O – bound applications.

3. Kernel cannot preempt individual threads, reducing fairness.
4. If one thread performs a blocking system call, the entire process gets blocked.
5. No true parallelism on multi core processors because the kernel schedules only the process, not individual threads.

---

2. Kernel level threads :-

- Managed & directly by the operating system kernel.
- Kernel schedules thread individually.
- Blocking of one thread does not effect others.
- More powerful but slower (Kernel Involvement).
- Kernel threads are slower to create and manage as operating system manages them.

Example – Windows, Linux support kernel level threads.

Advantages

1. Supports true parallel execution on multicore and multiprocessor systems.
2. If one thread blocks other thread continue execution.
3. Provides preemptive multitasking, improving responsiveness.
4. Better CPU utilization for concurrent application.
5. Suitable for I/O intensive and high performance systems.

Disadvantages

1. High overhead due to kernel involvement in thread management.
2. Thread creation and context switching are slower than user thread (level).
3. Increased system complexity.
4. Less portable, as implementation depends on OS support.

Hybrid Approach :-

In Hybrid Approach both Kernel level threads and user level threads are implemented.

(User level)
(Kernel level)
Hybrid

P – Process

⇒ Comparison between process and threads :-

Process	Thread
1. A process is a unit of resource ownership.	1. A thread is a unit of dispatching
2. A process is heavy weight.	2. A thread is light weight.
3. Process are independent of one another.	3. Threads are not independent of one another.
4. Process require more system resources then threads.	4. Threads require less system resources than process.
5. Each process has its own distinct address space in memory.	5. All threads in a process share the same address space.
6. A process does not depend on thread's state.	6. A thread is dependent on the process state (running, ready etc).
7. More complications arise when many processes run concurrently.	7. Multiple threads may run concurrently with ease
8. It takes more time to switch between two processes.	8. It takes less time to switch between two threads of a same process.

CPU Scheduling :-

Section - A

- CPU scheduling is the process of deciding which task get to use the CPU at any given time.
- CPU can execute only one process at a time, so it needs a way to choose which process run first.
- CPU scheduling is the basis of multiprogrammed operating system.

Example – Bank
X → Manager
A B C D E

Why CPU scheduling is important :-

- To keep the CPU busy all the time.
- To ensure faster task completion.
- To make sure process get a fair chance to execute.

How CPU scheduling Work :-

1. The OS keeps a list of all waiting task in a ready queue.
2. The CPU scheduler selects one process based on a scheduling algorithm.
3. The selected process runs for some time, then either completes or move back.
4. The next process is selected, ensuring fair execution.

Real Life Examples :- (imp)

1. **A Bank Queue : (FCFS)**

- The first person in the line get served first just like how CPU executes the process first.

2. **A Hospital Emergency Room :-**
   A → Serious Injured
   B → minor fever
3. Playing music songs in a playlist

⇒ CPU Scheduling Criteria :-

- Scheduling Criteria are parameters used to evaluate and compare CPU scheduling algorithms.
- The commonly used criteria can be grouped into two categories : user oriented and system oriented criteria.

User oriented criteria relate to the behaviour of the system as perceived by individual user or process.
For example – response time, turn around time.

System oriented criteria relate to the effective and efficient use of processor.
For example – throughput, processor utilization, fairness.

The various scheduling criteria for evaluating an algorithm are:

1. CPU Utilization :-

- CPU utilization refers to how busy the CPU is.
- It is the percentage of time CPU is doing useful work.
- CPU utilization should be as high as possible.
- The level of CPU utilization depends on the load on the system.
- CPU utilization may range from 0 to 100 percent.

- In a real system, it should range from 40 percent (for light loaded system) to 90 percent (for high loaded system).

---

2. Turn around Time :-

- Turn around time is the total time taken by a process from submission to completion.
- Lower turn around time indicates faster execution of process.
- Turn around time is inversely proportional to throughput.
- It is the metrics used for batch jobs.

---

3. Throughput :-

- Throughput is the number of process completed per unit time.
- For long process this rate may be 1 process per hour, for short processes, throughput may be 10 processes per second.
- Thus evaluation of throughput depends on the average length of a process.
- Higher throughput means better system performance.

---

4. Waiting Time :-

- Waiting time is the total time a process waits in the ready queue.
- It is the average period of time a process spends waiting.
- It is a time spent in waiting for a resource allocation.
  Therefore, waiting time is the penalty imposed for sharing resources with others.

Formula : Waiting time = Turnaround – Burst time

---

5. Burst Time :-

- Burst time is the CPU execution time required by a process.
- It does not include waiting time.

---

6. Arrival Time :-

- Arrival time is the time at which a process enters the ready queue.

---

7. Response Time :-

- Response time is defined as the time interval between the job submission and the first response produced by the job.
- Response time is an important metrics used in the interactive system.
- In order to obtain better performance response time should be low.

⇒ Scheduling Algorithms :-

CPU scheduling algorithms deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.

Algorithms are :-

1. First comes First served (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Multi level Queue (MLQ) scheduling.

---

1. FCFS Scheduling :-

- first come first served scheduling is the simplest CPU scheduling algorithm.
- The process that arrives first in the ready queue gets the CPU first.

Here P1 enters first in ready queue get the CPU first.

How FCFS Scheduling Works

Step-1) Process enter's the ready queue.
Step-2) CPU select the process with earliest 'arrival time'.
Step-3) That process runs until it finishes execution.
Step-4) Next process in the queue get the CPU.

Gantt chart :-

Important term used in FCFS :-

1. Arrival time : Time when process enter in ready queue
2. Burst Time : CPU execution time required
3. Completion Time : Time when process finish execution.
4. Turn around Time : Time from submission to execution.

TAT = Completion time – Arrival time

5. Waiting time :
   WT = TAT – Burst time

Ques 1 – Given the Arrival time and Burst time of processes, draw the Gantt chart using FCFS scheduling and calculate waiting time & turn around time.

Given :-

Process	Arrival time	Burst time
P1	0	4
P2	1	3
P3	2	2

Gantt chart →
P1 P2 P3
0 4 7 9

Let’s find more criterias :-

1. Completion time :
   P1 = 4
   P2 = 7
   P3 = 9
2. Turnaround time : TAT = CT – AT

P1 = 4 – 0 = 4
P2 = 7 – 1 = 6
P3 = 9 – 2 = 7

3. Burst time : Burst time = Turnaround time – waiting time
4. Waiting time : WT = TAT – BT

P1 = 4 – 4 = 0
P2 = 6 – 3 = 3
P3 = 7 – 2 = 5

Advantages :-

- Very simple and easy to implement
- No starvation every process get CPU.
- Suitable for batch OS.

Disadvantages :-

- In FCFS scheduling, waiting time can be large if short requests wait behind the long ones.
- A proper mix of jobs is needed to achieve good results from FCFS scheduling.
- FCFS is not suited for time sharing systems where each user needs to get the CPU for an equal amount of time interval.

---

2. SJF Scheduling :-

(Shortest Job First Scheduling)

- SJF stands for shortest job first algorithm and is fastest than FCFS.
- SJF is a CPU scheduling Algo in which the process with the smallest burst time gets the CPU first.
- If two processes have the same length of next CPU burst, FCFS scheduling algorithm is used to break the tie.
- The time taken by CPU to execute a process is called burst time.
- SJF algorithm can be preemptive or non - preemptive.

Non - Preemptive SJF Scheduling :-

- Once a process start execution, it runs till completion.
- CPU can not be taken back forcibly.

Preemptive SJF Scheduling :-

- A running process can be interrupted if a shorter job arrive.
- The preemptive version of SJF scheduling is known as shortest Remaining Time scheduling (SRT).

- SRT scheduling is useful in timesharing system.

How SJF Scheduling Works :-

Step1) Ready process arrives in ready queue.
Step2) All ready process compared based on burst time.
Step3) The process with the smallest burst time get the CPU.

Advantages of SJF Scheduling :-

- Minimum average waiting time
- Efficient CPU utilization
- Better than FCFS in performance

Disadvantages of SJF Scheduling :-

- Starvation of long processes.
- Burst time predict is difficult
- Not suitable for real time system.

---

3. Round Robin (RR) Scheduling :-

- Round Robing (RR) is a preemptive CPU scheduling algorithm.
- Each process gets CPU for a fixed time slice, called Time Quantum.

After time quantum expires, CPU is taken back and given to the next process.

P1 P2 P3 P4 P5 CPU

Quantum = 5 second

⇒ What is Time Quantum ?

- Time Quantum is maximum time a process can use the CPU at once.
- It decide by the operating system.
- If a process finishes before time Quantum - CPU moves to next process.
- If not process goes back to ready Queue.

⇒ How Round Robing Scheduling Works ?

- All process are placed in ready Queue.
- CPU is allocated to each process for given time Quantum.
- If a process is not finished, it is preempted and added to the end of the queue.
- The Cycles continues until all processes completed.

⇒ Advantages of Round Robin Scheduling :-

- Fair chance scheduling (no starvation)
- Better response time
- Suitable for time sharing system

⇒ Disadvantages :-

- Too small time Quantum → too many context switch
- Too large time Quantum → behave like FCFS

---

4. Multi level Queue Scheduling :-

- Multi level Queue scheduling is a CPU scheduling algorithm in which the ready queue is divided into multiple separate queues each for a different type of process.
- Each queue has its own scheduling algorithm, and processes are permanently assigned to a queue.

Why it is used

1. Different processes have different priorities.
2. Improve system efficiency.
3. Separates system tasks from user tasks.
4. Suitable for systems with mixed workloads.

Structure of Multi level Queue

The ready queue is divided based on process type, for example

Queue No	Process type	Scheduling Algorithm
Queue 1	System processes	Priority Scheduling
Queue 2	Interactive processes	Round Robin
Queue 3	Batch processes	FCFS

CPU Selection b/w Queues

There are two main ways CPU is allocated among queues :

1. Fixed Priority Scheduling

- Each queue has a fixed priority.
- Higher priority queue executes first.
- Lower priority queues may starve.

Example
System queue > Interactive queue > Batch queue

---

2. Time sharing b/w Queues

- Each queue gets a fixed time slice
- CPU is shared fairly among queues.

Example

- System Queue → 50% CPU time
- Interactive Queue → 30%
- Batch Queue → 20%

Deadlock

Section – B

- A deadlock is a situation where two or more processes are waiting forever, and none of them can proceed.
- In simple words everyone is waiting for something that another process is holding.
- Due to locking of resource, each process has to wait for resource that is locked by another process and as a result none of the transaction can finish.

Example

Imagine two people trying to eat with two spoons :

- Person A picks up the left spoon
- Person B picks up the right spoon
  Now both are waiting for the other spoon to eat : this creates a deadlock

Explain using diagram :-

Shared Resource – Processes

- Process P1 hold resource R1 and waits for R2.
- Process P2 hold resource R2 and wait for R3
- Similarly P3 and P4 take resource R3 and R4 and wait for R1 and R2

Now all the processes stucks forever – none can move more forward.

⇒ Necessary condition for deadlock (Coffman's condition) :-

1) Mutual exclusion :-

Means a resource can be used by only one process at a time.

Example :- A printer can print only one document at a time.
If P1 is printing, P2 must wait.

---

2) Hold and wait :-

A process is holding one resource and waiting for another.

Example :-
P1 holds the printer and waiting for scanner.
P2 hold the scanner and wait for printer.

---

3) No preemption :-

Once a process get a resource, it can't be taken away forcibly, it must release it voluntary.

If a process is using the printer OS can not just snatch it.
This rules increase safety but also make deadlock possible.
To avoid this OS can allow preemption in some cases.

---

4) Circular wait :-

There exist a circular chain of processes where each is waiting for a resource held by the next.

⇒ Resource Allocation Graph (RAG) :-

- Resource Allocation graph is a graphical method used to represent the allocation of resource to process.
- It helps the operating system to detect deadlock.

- Simply : representing the allocation of resource to process and the request of resource by process on graph.

Components of Resource Allocation graph :-

a) **Process Node :** Represent by Circle 'P'
→ Denote as P1, P2, P3 etc

b) **Resource Node :** Represent by square
'R'
→ Denote as R1, R2, R3 etc

Edges in RAG

(a) Request Edge → Process to resource
(b) Allocation Graph → Resource to process

How resource allocation graph works :-

Step1 Process request resource for execution
Step2 Resource are allocated by the OS
Step3 These request and allocation are shown using edges in the graph
Step4 The graph help to analyze whether a deadlock situation exists or not

---

Cycle in RAG :-

- A cycle is closed path
- If a cycle exist, then system is in deadlock
- If there is no cycle, then the system is safe.

⇒ Deadlock Prevention

- Deadlock prevention is a technique in which the OS make sure that deadlock never happen by denying at least one of the necessary condition of deadlock.

1. Prevent mutual exclusion :-

- Resource can be used by only one process at a time.
- We can prevent this condition by making a resource shareable.
- Allows resource to be shared if possible like read only files.

Files can be shared.

- A read only file can be used in shareable mode if several processes attempt to open a read only file at the same time, they can be granted simultaneous access.
- A never needs to wait for shareable resource.
- Problem : most resource cannot be shared (printer, scanner etc)

---

2. Preventing Hold & wait :-

- Hold & wait : A process hold some resource and wait for other.
- A process must request all require resource at the beginning.
- If we can prevent processes that hold resources from waiting more resources, we can eliminate deadlock.
- Thus, in this we must ensure that whenever a process requests a resource, it should not hold any other resources.

Example – Before running, a process must ask for
CPU → memory → I/O device → file access

Problem : process wait longer even if some resource are free.

---

3. Preventing No preemption :-

No preemption : Resource cannot be forcibly taken from a process.

- If a process requested a resource that is not available
  • All resource currently held are taken away (preempted)
  • Process is restart when all resource are available.
- Preemption can be applied to those resources whose state can easily saved and later restored. For example, CPU registers and memory space.

---

4. Preventing Circular Wait :-

- This method ensures that a circular chain of processes is not formed in a system.
- The circular wait can be eliminated in several ways.
- Process must order in increasing order of resources.
- One way is to use a rule : A process is entitled only to a single resource at any moment. If it needs a second one, it must release the first one.

⇒ Deadlock avoidance :-

- Deadlock avoidance is a technique in operating system that decide whether to allocate a resource or not.
- It means the system check every request before giving a resource.
- The OS gives resource only if after allocation, the system remains safe.
- A deadlock avoidance algorithm dynamically examines the resource allocation state to ensure that a circular wait condition can never exist.

Basic idea :-

Every process must try :
→ How much maximum resource it might need

OS check :-
→ If granted the resources keep the system safe / unsafe
→ If becomes unsafe → Deny / Wait

---

1. Safe state and Unsafe state :-

a) **Safe state :** A state where process can complete in safe order.
→ At least one safe sequence exist

A system is said to be in a safe state if there is a sequence (safe execution)

Example → P1 → P3 → P2

b) **Unsafe state :** A state where deadlock may occur.
→ Resource not allocated in this state.

---

2. Resource Allocation Graph Algorithm :-

- Resource Allocation graph Algorithm helps in deadlock avoidance in system having only one instance of each resource type.
- This algorithm uses the resource (algorithm) allocation graph to determine whether the allocation of a particular resource leads to the formation of cycles in graph or not.
- If no cycles exits then the allocation of the resource leaves the system in the safe state.

Resource allocation graph for deadlock avoidance
An unsafe state in a resource allocation graph

---

3. Banker’s Algorithm :-

- Banker’s Algorithm is used to avoid deadlock.
- It is mainly used to check the safe sequence.
- If allocation makes the system unsafe → request is not granted.

The name banker’s algorithm depicts the similarity of the concept used in the field of banking.

A banker never allocates the cash available in such a manner that the banker cannot fulfill needs of its clients.

The banker’s algorithm has two parts : Safety algorithm and resource request algorithm.

The safety algorithm is applied by an operating system to determine whether or not a system in safe state.

The resource request algorithm is applied by an operating system to determine whether or not a request forwarded by a process for acquiring a resource would lead the system to an unsafe state.

Banker Algorithm Dry Run :-

- There are 4 Process P0 P1 P2 P3
- There are 3 Resource A B C

Total Resource in System :-
A = 9 B = 7 C = 6

Process	Max Need (A B C)	Allocated (A B C)	Need (A B C)	Available (A B C)	Sequence
P0	7 5 3	0 1 0	7 4 3	2 5 3	(4)
P1	3 2 2	2 0 0	1 2 2	4 5 3	(1)
P2	9 0 2	3 0 2	6 0 0	6 6 4	(3)
P3	4 3 3	2 1 1	2 2 2	9 6 6	(2)

- Need = Max – Allocated
- Available = Total – Σ (Allocation)

Find safe sequence

1. check P0 (Need ≤ Available) (7,4,3 ≤ 2,5,3) ? → No
   check P1 (Need ≤ Available) (1,2,2 ≤ 2,5,3) ? → Yes choose P1

- Mark P1 finish → Available + Allocated (2,5,3 + 2,0,0)
  Net Available = (4,5,3)

2. Available (4,5,3)

check P0 : (7,4,3 ≤ 4,5,3) ? → No
check P2 : (6,0,0 ≤ 4,5,3) ? → No
check P3 : (2,2,2 ≤ 4,5,3) ? → yes choose P3

- Mark P3 finished (Available + Allocated) (4,5,3 + 2,1,1)
  Net Available → (6,6,4)

Sequence : P1 → P3

3. Available (6,6,4)

   check P0 : (7,4,3 ≤ 6,6,4) → No

check P2 : (6,0,0) ≤ (6,6,4) → Yes choose P2

- Mark P2 finished (Available + Allocated) (6,6,4) + (3,0,2)
  Net Available = (9,6,6)

Sequence : P1 → P3 → P2

---

4. Available : (9,6,6)

Now only P0 left check (7,4,3) ≤ (9,6,6) yes

- Mark P0 finish (9,6,6) + (0,1,0) = (9,7,6) → Total available

Sequence : P1 → P3 → P2 → P0 (Safe Sequence)

⇒ Dealing with deadlocks / methods for handling deadlocks :-

1. Deadlock Prevention :-

- Measures are taken so that the system does not go into deadlock condition in the first place.
- Deadlock prevention techniques prevent the occurrence of at least one of the four conditions that cause deadlock.

---

2. Deadlock avoidance :-

- In deadlock avoidance, before a process is allocated required resources it is made sure that the request will not lead to deadlock.

- In order to decide whether the current request can be satisfied the system should consider :
  • the resources currently available
  • the resources currently allocated to each process
  • the future requests and releases of each process

---

3. Deadlock recovery :-

- It means, let the deadlock occur in the system and then detect it and recover the system from deadlock.
- In this environment, the system can provide an algorithm that examine the state of a system to determine whether a deadlock has occured and can then use an algorithm to recover from the deadlock.

---

4. Deadlock Ignorance :-

- An operating can assume that deadlock will never happen and fully ignore it.
- This approach is also known as no policy approach.
- The advantage of this approach is that it saves CPU time and memory space for deadlock management.

File Management

Section - B

- Allocation methods refers to the techniques used by an operating system to allocate disk blocks (secondary storage space) to files so that data can be stored, retrieved, and managed efficiently.
- The allocation method determines:
  • How disk blocks are assigned to a file
  • How files are accessed (sequential or direct)
  • How disk space is utilized.
- A good allocation method should:
  • Minimize disk access time
  • Use disk space efficiently
  • Allow files to grow easily

⇒ Types of Allocation methods :-

There are three main allocation methods used by operating systems:

---

1. Contiguous Allocation :-

- Contiguous allocation is an allocation method in which all the disk blocks required by a file are allocated consecutively (one after another) on the disk at the time of file creation.
- Each file occupies a continuous sequence of disk blocks.

Working

- When a file is created, the operating system:
  (i) Searches for a free contiguous set of disk blocks
  (ii) Allocates all required blocks together.
- The directory entry for the file contains:
  (i) Starting block address
  (ii) Length of the file (no of blocks)

If a file starts at block **b** and has length **n**, then the blocks used are:

$b, b+1, b+2, \dots, b+n-1$

Example

Suppose:

- Disk block size = 1 KB
- File size = 5 KB

Then:

- The file needs 5 contiguous blocks
- If starting block = 10
- The file occupies blocks:

10, 11, 12, 13, 14

To access the 3rd block of the file:

Block no = Starting block + offset
= 10 + 2
= 12

So access is very fast

Advantages

1. **Simple implementation :-**
   - The operating system only needs to store the starting block and file length, making management easy.
2. **Fast access time :-**
   - Both sequential access and direct access are very efficient because blocks are stored next to each other.
3. **Minimum disk head movement :-**
   - Since blocks are contiguous, the disk head does not need to move frequently, improving performance.
4. **Best suited for sequential files :-**
   - Large files such as videos, audio files, and logs can be read very efficiently.

Disadvantages

1. **External fragmentation**
   - Free disk space gets broken into small pieces, making it difficult to find large contiguous space.
2. The operating system must know the file size at creation time.
3. If a file grows beyond its allocated space, it must be reallocated.

---

2. Linked Allocation

- Linked allocation is an allocation method in which a file is stored as a linked list of disk blocks, and the blocks may be scattered anywhere on the disk.
- Each block contains:
  • Data
  • A pointer to the next block

Working

1. The directory entry stores:
   • Address of the first block
2. Each disk block contains:
   • Data part
   • Pointer to the next block
3. The last block contains a NULL pointer, indicating end of file.

The file is accessed by following pointers from one block to the next.

Example

Suppose a file occupies blocks:

5 → 18 → 2 → 30 → 9 → NULL

These blocks are located at different positions on the disk.

To read the file:

- OS starts from block 5
- Reads data
- Follows pointer to block 18
- Continues until NULL is reached

Advantages

1. Any free block can be used, so disk space utilization is efficient.
2. Files can grow dynamically as long as free blocks are available.
3. Reading a file sequentially is simple and effective.
4. Since blocks are scattered, compaction is unnecessary.

Disadvantages

1. Each block requires space to store a pointer, reducing usable disk space.
2. If a pointer is lost or damaged, the remaining part of the file becomes inaccessible.
3. Each pointer traversal may require an additional disk access.

---

3. Indexed Allocation :-

- Indexed allocation is an allocation method in which all the block addresses of a file are stored in a separate block called the index block.
- Each file has:
  • One index block
  • The index block contains pointers to all data blocks of the file.

Working

1. When a file is created:
    - An index block is allocated.

- All entries in the index block are initialized to NULL.

2. When data is written:
    - Free disk blocks are allocated
    - Their addresses are stored in the index block
3. The directory entry stores:
   • Address of the index block.

To access any block:

- OS first reads the index block
- Then accesses the required data block directly

Example

Suppose a file uses blocks:

4, 7, 19, 25

Index block contents:

[4 | 7 | 19 | 25]

To access the 3rd block:

- OS reads index block
- Fetches address at position 3 → block 19

Advantages

1. Any block can be accessed directly using the index block.
2. Disk blocks can be allocated anywhere.
3. Random access is faster compared to linked allocation.
4. New blocks can be added by updating the index block.

Disadvantages

- Each file requires an additional index block, even for small files.
- File size is limited by the number of pointers that can be stored in the index block.

Disk management

Section - B

⇒ Disk Scheduling

- Disk scheduling refers to the technique used by the operating system to decide the order in which disk I/O requests are serviced so that disk performance is improved.
- In multi programming operating system, multiple processes may request disk I/O operations at the same time.
- These requests are stored in disk request queue.
- Since disk operations are much slower than CPU operations, improper servicing of disk requests can seriously degrade system performance.
- Disk scheduling is required because:
  • Disk access time is very high compared to CPU time
  • Disk head movement is expensive
  • Multiple I/O requests compete for disk access
  • Poor scheduling increases waiting time and response time.

Disk access time components

1. **Seek time :-** Time taken to move the disk head to the required track.
2. **Rotational latency :-** Time taken for the desired sector to come under the read/write head.
3. **Transfer time :-** Time taken to transfer data.

→ Disk scheduling mainly focuses on reducing seek time.

- The operating system takes out the I/O requests from the queue and process them one by one. The algorithm used to select which I/O request is going to be satisfied first is called disk scheduling algorithm.

There are many disk scheduling algorithms are:-

---

1. First come first served (FCFS / FIFO) :-

- First come first served is a disk scheduling algorithm in which disk I/O are serviced in the exact order in which they arrive in the queue without any reordering.
- FCFS follows a simple queue structure.
- The disk head moves from its current position to the track requested by the first process, then to the second, and so on.
- The operating system does not consider the distance between tracks, which may result in large head movements.

Working

1. Disk requests arrive and are placed in a queue.
2. Requests are serviced sequentially
3. Disk head moves according to arrival order
4. Each request is fully serviced before moving to the next

Example

Assume:
• Disk size = 200 tracks (0 – 199)
• Request queue = 23, 89, 132, 42, 187, 60
• Initial head position = 100

Service order :

100 → 23 → 89 → 132 → 42 → 187 → 60

The disk head moves back and forth unnecessarily, resulting in very high total head movements (548) tracks.

---

2. Shortest Seek Time First (SSTF) :-

- Shortest seek time first is a disk scheduling algorithm that selects the disk request requiring the minimum seek time from the current head position.
- Instead of servicing requests in arrival order, SSTF chooses the request closest to the current disk head position.
- This reduces unnecessary head movement and improves performance compared to FCFS.

Working

1. Disk head checks all pending requests.
2. Distance from current head position is calculated.
3. The request with minimum seek distance is selected.
4. Process repeats after servicing each request.

Example

Initial head = 100
Requests = 23, 89, 132, 42, 187, 60

Service order:

100 → 89 → 60 → 42 → 23 → 132 → 187

Total head movement = 241 tracks, which is much less than FCFS

Advantages

1. Throughput is higher as compared to FIFO
2. It provides better performance than FIFO and produces less no of head movements.

Disadvantages

1. Requests that are far from the current head position may wait indefinitely if closer requests keep arriving.
2. User may experience inconsistent response times.

---

3. SCAN Scheduling :-

- Scan scheduling moves the disk head in one direction, servicing all requests along the way until it reaches the end of the disk then reverses its direction.
- The disk head behaves like an elevator in a building.
- It services all requests while moving in one direction, then reverses and services in the opposite direction (requests).

Working

1. Disk head starts moving in one direction
2. Services all requests in that direction
3. Reaches the last track
4. Reverses direction and continues servicing

Example

Initial head = 100
Requests = 23, 89, 132, 42, 187, 60

Service order -

100 → 89 → 60 → 42 → 23 → 0 → 132 → 187

Total head movement = 287 tracks

Advantages

1. Throughput is better than FIFO.
2. It also considers the starvation of request. In this regard it is better than SSTF.

Disadvantages

1. **Unnecessary movement**
   The disk head moves to the extreme ends even when no request exists there.

---

4. Circular Scan (C-Scan)

- Circular Scan services disk requests only in one direction. After reaching the last track, the disk head returns directly to the first track without servicing any request.
- C Scan treats the disk as a circular list. This ensures uniform waiting time for all requests.

Example

Service order :

100 → 89 → 60 → 42 → 23 → 0 → 199 → 187 → 132

Total head movement = 311 tracks

Advantages

1. **Uniform waiting time**
   All requests are treated equally regardless of position
2. **No bias toward middle tracks**
   Prevents unfair advantage to central cylinders.

Disadvantages

1. **More head movement**
   The jump from last to first track increase seek time.
2. **Lower efficiency than LOOK**
   Some movements do not service any request.

LOOK Scheduling

- Look scheduling is an optimized version of SCAN in which the disk head moves only as far as the last request in each direction instead of going to the disk end.
- The disk head looks ahead to see if further requests exist before continuing. This avoids unnecessary movements.

Example

Service order :
$100 \rightarrow 89 \rightarrow 60 \rightarrow 42 \rightarrow 23 \rightarrow 132 \rightarrow 187$

$= |100 - 89| + |89 - 60| + |60 - 42| + |42 - 23| + |23 - 132| + |132 - 187|$

$= 11 + 29 + 18 + 19 + 109 + 55$

= **241 tracks [Total head movements]**

Memory Management

Section – B

⇒ Logical and physical address space :-

Logical address ⇒

- It is also known as a virtual address
- It is a address generated by CPU during program execution
- This address is used as a reference to access the physical memory located by CPU.

Physical address ⇒

- It is the actual address in main memory where data is stored.
- Physical address are used by the memory management unit (MMU) to translate logical address to physical addresses.

⇒ Difference between Logical & Physical address

Logical address Physical address

1. It is generated by CPU. | 1. It is access the physical memory location by CPU.
2. It is set of all logical address generated by CPU in reference to a program. | 2. It is set of all physical address mapped to the corresponding logical address.
3. User can view the logical address of a program. | 3. User can never view physical address of a program.
4. User can use the logical address to access the physical address. | 4. The user can indirectly access physical address but not directly.
5. Logical address can be changed. | 5. Physical address will not change.
6. It is also called virtual address. | 6. It is also called real address.

⇒ Swapping :-

- Swapping is a memory management technique in which a process is temporarily removed from main memory (RAM) and stored in secondary storage (usually hard disk) and later brought back into main memory for continued execution.
- In simple words, swapping allows the operating system to free main memory space by moving inactive or less priority processes to disk and loading other processes into memory.

Swapping is required because :

- Main memory (RAM) is limited in size
- In a multiprogramming system, many processes compete for memory
- Sometimes all active processes cannot fit in RAM
- Swapping helps the system execute more processes than the memory can normally hold.

⇒ How Swapping Works

1. Initially, several processes are loaded into main memory.
2. When a new process arrives and insufficient free memory is available :
   - One existing process is selected by the OS.
3. That selected process is moved from main memory to secondary storage.
   - This operation is called **Swap Out**.
4. After freeing memory, the new process is loaded into main memory.
5. When the swapped out process is needed again :
   - It is brought back from secondary storage to main memory.
   - This operation is called **Swap In**.

⇒ Swap In and Swap Out

- **Swap Out** – The process of moving a process from main memory to secondary storage.
- **Swap In** – The process of moving a process from secondary storage back to main memory.

⇒ Swap Space :-

- The area on the disk where swapped out processes are stored is called swap space.

⇒ In a multiprogramming system swapping :

(i) More processes are kept ready to run.
(ii) Swapping helps increase CPU utilization.
(iii) While one process is waiting, another process can be loaded.

⇒ Swapping and CPU Scheduling

1. Round Robin Scheduling

- When a process's time quantum expires, it may be swapped out.
- Another process is swapped in.
- This ensures fairness and better CPU usage.

2. Priority Scheduling (Roll Out , Roll In)

- A low priority process may be swapped out.
- A high priority process is swapped immediately.
- After completion, the low priority process is swapped back in.
- This is called Roll Out – Roll In.

Example of Swapping

Suppose main memory contains processes A, B and C.
A new process D arrives, but there is no free memory.

- Process A is swapped out to disk.
- Process D is loaded into memory.
- Later, when A is required again, D or another process is swapped out and A is swapped in.

⇒ Contiguous memory allocation

- Contiguous memory allocation is a memory management technique in which each process is allocated a single continuous block of memory.
- In this method, the entire program (data + instructions) is stored in adjacent memory locations.

In Simple :

- Memory is divided into partitions
- Each process occupies one complete partition
- No process is split across different memory locations

Types

1. Simple Partition Allocation

- Memory is divided into two parts operating system and one user process
- Only one process at a time can execute.
- Used in early operating systems

---

2. Multiple Partition Allocation

(a) **Fixed Partition Allocation**

- Memory is divided into fixed size partitions
- Each partition holds exactly one process
- Leads to internal fragmentation

(b) **Variable Partition Allocation**

- Memory partitions are created dynamically
- Size depends on process requirement
- Leads to external fragmentation

Allocation process in contiguous allocation

1. OS searches for a free contiguous block
2. Block size must be greater than or equal to process size
3. Process is loaded into that block
4. Memory remains occupied until the process terminates

Example

If a process requires 25 KB of memory and block size is 1 KB, then 25 consecutive blocks are allocated to that process.

- The process can not use memory outside these blocks.

⇒ Paging :-

- Paging is a memory management technique in which the logical address space of a process is divided into fixed size blocks called pages, and the physical memory is divided into fixed size blocks called frames.
- In paging, pages of a process are loaded into any available frames in physical memory, so the memory allocation is non-contiguous.

Paging is required to overcome the limitations of contiguous memory allocation such as :

- External fragmentation
- Difficulty in allocating large contiguous memory blocks
- Inefficient memory utilization

Paging allows :

- Better utilization of main memory
- Higher degree of multiprogramming
- Flexible memory allocation

⇒ Logical address in Paging

A logical address generated by the CPU consists of two parts :

1. **Page Number (p)**
   - Identifies which page of the process is required.
   - Used as an index into the page table.
2. **Page offset (d)**
   - Specifies the exact location within the page

So, Logical address = (Page Number , Page offset)

⇒ Physical address in Paging

A physical address also consists of two parts :

1. **Frame Number (f)**
    - Identifies the frame in physical memory
2. **Frame offset (d)**
    - Same offset as page offset (because page size = frame size)

(I) Page Table :-

- A page table is a data structure maintained by the operating system that stores the mapping b/w page numbers and frame numbers.

Structure of a page table

Each entry in the page table contains :

- Page number (implicitly by index)
- Corresponding frame number
- Control bits (valid / invalid , protection bits)

Function of page table

When a page number is given, the page table tells :

- Whether the page is present in memory.
- If present, which frame contains that page.

Address translation in Paging

The method of converting a logical address into its corresponding physical address is known as address translation.

---

1. CPU generates a logical address (p , d)
2. Page number p is sent to the page table
3. Page table returns the frame number f
4. Frame number f is combined with offset d
5. Physical address (f , d) is formed
6. Required memory location is accessed.

Page Size and address bits

- Page size is always a power of 2.
- This makes division of address into page number and offset easier.

If :

- Logical address size = 2^m
- Page size = 2^n

Then :

- Page number = m – n bits
- Page offset = n bits

Example

If :

- Logical address size = 2^13 (13 bits)
- Page size = 2^10 (1 KB)

Then :

- Page number = 13 – 10
  = 3 bits
- Offset bits = 10 bits

II) Implementation of page table

1. Using Registers

- Page table is stored in a set of high speed registers.
- These registers are built with very high speed logic that effectively translate logical address into physical address.
- These registers are loaded by CPU dispatcher, just as it reloads the other registers.
- Registers provide very fast access.

---

2. Using Page table base register

- This method is used when page table contains millions of entries.
- Page table is stored in the main memory.
- Page table base register holds the base address of the page table.

- On context switching, only PTBR needs to be updated.

---

3. Using associative memory or translation look-aside buffer

- In order to overcome the problem of longer duration of access time, we can use high speed associative memory.
- Translation look aside buffer (TLB) is a small, fast cache that stores recent page frame mappings.

Working of TLB

1. Page number is searched in TLB.

2. If found → TLB hit
   • Frame number obtained immediately
3. If not found → TLB miss
   • Page table is accessed
   • Entry is added to TLB

Hit Ratio

Hit ratio is the percentage of times a page number is found in the TLB.

Higher hit ratio → better performance

If the page number is present in the TLB it is known as TLB hit.

Effective memory access time

• TLB Hit → Fast access
• TLB Miss → Slower access

Effective access time is calculated using :
• Hit ratio
• TLB access time
• Memory access time

(Usually asked as numerical problem in exams)

III. Memory protection in paging

Protection bits

Each page table entry contains protection bits such as :

• Read only
• Read Write

If a process tries to :
• Write to a read only page → trap to OS

Valid / Invalid bit

• Valid bit = 1 → Page is part of the process address space.
• Invalid bit = 0 → Page is not part of process address space

---

Purpose

• Prevents illegal memory access
• Ensures memory protection.

If an invalid page is accessed :
• Hardware trap is generated
• Control is transferred to OS

IV. Sharing in Paging

- When paging is implemented, we can share the common program code i.e. the pages containing common code can be shared by no. of processes.
- For a code to be shared, it is important that it should be reentrant. A reentrant code is also known as pure code.
- Thus, a reentrant code does not change during the execution of a program. As a result it can be shared.
- Two or more processes can execute the same code at the same time.

Virtual Memory
Section – B

- Virtual memory is a memory management technique in which a program can execute even if the entire program is not present in main memory (RAM).
- Only the required portions of a program (pages or segments) are loaded into main memory, while the remaining parts are kept in secondary storage (disk).
- In simple words, virtual memory creates an illusion of a very large memory by using secondary storage along with main memory.

- Virtual memory is required because:

1. Physical memory is limited and cannot hold all programs completely.
2. Modern programs are very large in size.
3. Multiple programs run simultaneously in a multiprogramming environment.
4. Efficient utilization of memory is required to improve CPU performance.

- A process is divided into pages. Only those pages which are currently required for execution are loaded into main memory.
- Remaining pages stay on secondary storage disk.
- When a required page is not present in memory, a page fault occurs. The OS then loads the required page into memory.

Thus, a program can be larger than physical memory.

Working of Virtual Memory

1. The CPU generates a logical address.
2. The memory management unit checks whether the required page is present in main memory.
3. If present → execution continues normally.
4. If not present → page fault occurs.
5. The operating system brings the required page from secondary storage into main memory.
6. Page table is updated and execution resumes.

Advantages

1. Large Address Space

- Virtual memory allows a program to use a large logical address space even when physical memory is small.
- This simplifies programming because the programmer does not have to worry about memory limitations.

2. Increased degree of multiprogramming

- Since only required pages are loaded, more processes can reside in memory at the same time.
- This increases CPU utilization and system throughput.

3. Reduced I/O operations

- Entire programs are not loaded at once. Only necessary pages are transferred, reducing disk I/O.

4. Faster Program Execution

- Programs start execution quickly because only a few pages are loaded initially.

Disadvantages

1. Additional hardware requirement

- Virtual memory requires special hardware like MMU, page tables, and TLB, increasing system cost.

2. Page Fault Overhead

- Frequent page faults increase execution time because disk access is much slower than memory access.

3. Thrashing

- If too many page faults occur, the system spends most of its time swapping pages instead of executing instructions.

Implementation of Virtual Memory by Demand Paging

- Demand paging is a virtual memory technique in which pages are loaded into main memory only when they are required, not in advance.
- A page is brought into memory on demand, i.e., when it is referenced for the first time.

Why it is called demand paging?

- Pages are not loaded initially.
- They are loaded only when demanded by the CPU.

- In demand paging, only those pages of the process are brought into memory that are required.
- All other pages reside on secondary storage and are brought into memory when needed.
- This task of bringing in and out the various pages of a process is done by a pager.

- The basic components used in demand paging are page table, valid/invalid bit, secondary storage (disk), memory management unit.
- Valid bit = 1 → Page is present in main memory.
- Invalid bit = 0 → Page is not present in main memory (on disk or illegal).

This bit helps in detecting page faults.

- A page fault occurs when a process tries to access a page that is not present in main memory.

Steps involved in handling a page fault

1. CPU generates a logical address.
2. MMU checks the page table.
3. Valid/Invalid bit is checked.
4. If the page is invalid → page fault occurs.
5. Control is transferred to the operating system.
6. OS checks whether the reference is valid.
7. If valid, OS finds a free frame in main memory.
8. Required page is brought from secondary storage to main memory.
9. Page table is updated.
10. Execution resumes from the interrupted instructions.

Pure Demand Paging

- Pure demand paging is a form of demand paging in which not even a single page of a process is loaded initially.
- The very first instruction causes a page fault.

Characteristics of pure demand paging

1. Pages are loaded when referenced only.
2. Initial page faults are very high.
3. Memory utilization is highly efficient.
4. Performance may degrade if page fault rate is high.

- Effective access time represents the average time taken to access memory, considering both normal access and page fault cases.

Effective access time = $(1 - p) \times ma + p \times$ page fault time

---

Page Replacement Algorithms

- In a virtual memory system, programs are divided into fixed-size blocks called pages and main memory is divided into frames.
- During execution, if a required page is not present in main memory, a page fault occurs.
- When a page fault occurs and no free frame is available, the operating system must remove one of the existing pages from memory to make space for the incoming page.
- The decision of which page should be removed is taken by a page replacement algorithm.
- Thus, a page replacement algorithm is a policy used by the operating system to select a page to be swapped out from main memory when a page fault occurs.

Objectives

- To minimize the number of page faults.
- To improve CPU utilization.
- To reduce disk I/O operations.
- To improve overall system performance.
- A reference string is a sequence of memory references (page numbers) generated by a process during execution.
- Page replacement algorithms are analyzed and compared by applying them to the same reference string and counting the number of page faults produced.

I. Optimal Page Replacement Algorithm

- The optimal page replacement algorithm replaces the page that will not be used for the longest period of time in the future.
- This algorithm assumes that the operating system has complete knowledge of future page references, making it theoretical in nature.
- This algorithm is also called OPT / MIN.

Working

1. The operating system examines the future page reference string.
2. It determines which page currently in memory will not be used for the longest duration.
3. That page is selected for replacement.
4. The required page is then loaded into the freed frame.

Thus, the algorithm always makes the best possible replacement decision.

Characteristics

- Produces the minimum possible number of page faults.
- Used as a benchmark to evaluate other algorithms.
- Based on the strategy of looking forward in time.

Advantages

1. Improves system performance by reducing unnecessary swapping.
2. Gives the lowest page fault rate among all algorithms.

Disadvantages

1. Impossible to implement practically since future references are unknown.
2. Requires perfect prediction of program behaviour.

II. First In First Out (FIFO) Page Replacement Algorithm

- FIFO replaces the page that was brought into memory first.
- The page that has been in memory for the longest time is selected for replacement.
- It is implemented by creating a FIFO queue that holds all pages in memory.

Working of FIFO

1. Pages are maintained in a queue structure.
2. When a page is loaded into memory, it is inserted at the rear of the queue.
3. When a page fault occurs and replacement is required:
   - The page at the front of the queue (oldest page) is removed.
4. The new page is inserted at the rear of the queue.

Thus, pages are replaced strictly based on arrival time without considering their usage.

Advantages

- FIFO is very simple and easy to understand.
- Easy to implement using a queue.
- Requires minimal overhead.

Disadvantages

- FIFO may replace frequently used pages, leading to poor performance.
- It does not consider page usage patterns.
- FIFO suffers from **Belady's anomaly**, where increasing frames can increase page faults.

Belady's Anomaly

- A situation in which increasing the number of frames results in an increase in the number of page faults.
- Normally, increasing frames should reduce page faults, but FIFO sometimes shows opposite behaviour.
- This anomaly was discovered by Belady, Nelson, and Shedler in 1970.

III. Least Recently Used (LRU) Page Replacement Algorithm

- LRU replaces the page that has not been used for the longest period of time in the past.
- This algorithm assumes that pages used recently will likely be used again soon.

Working of LRU

1. The operating system keeps track of the time of last use of each page.
2. When a page fault occurs:
    - The least recently accessed page is identified.
3. That page is replaced with the required page.

Thus, LRU uses past behaviour to predict future usage.

Implementation of LRU Page Replacement

1. Using Counters

- Each page table entry has a counter storing time of last reference.
- Whenever a page is accessed, the counter is updated.
- On page fault, the page with the smallest counter value is replaced.

Drawback: Updating counters on every access increases overhead.

2. Using Matrix

- An n × n matrix is used for n page frames.
- When page k is accessed:
  (i) All bits in row k are set to 1.
  (ii) All bits in column k are set to 0.
- The page with the lowest binary row value is least recently used.

Drawback: Requires large memory and hardware support.

3. Using Linked List

- Pages are stored in a linked list.
- Most recently used page is kept at the head.
- Least recently used page is at the tail.
- On replacement, the tail page is removed.

Drawback: Time-consuming operations.

Thrashing

- Thrashing is a condition in which a system spends more time handling page faults and swapping pages between main memory and secondary storage than executing actual CPU instructions.
- In a thrashing state, the CPU remains mostly idle while the disk is heavily utilized.
- Thrashing occurs when the working set of active processes cannot be fully accommodated in main memory.
- As a result, processes continuously request pages that are not present in memory, leading to frequent page faults.
- Each page fault requires:
  - Suspension of the current process
  - Disk access to fetch the required page
  - Possible replacement of an existing page

When this happens repeatedly, system performance degrades drastically.

Step-by-step process of Thrashing

1. The system increases the degree of multiprogramming by loading more processes into memory.
2. Each process is allocated fewer frames than required.
3. Processes begin to generate frequent page faults.
4. CPU utilization decreases because processes are waiting for I/O operations.
5. The operating system misinterprets low CPU utilization as under-utilization.
6. OS increases the degree of multiprogramming further.
7. This causes even more page faults.
8. Eventually, the system enters a thrashing state, where most of the time is spent on paging.

Causes of Thrashing

1. High degree of Multiprogramming

- When too many processes are loaded simultaneously, available memory frames are divided among them, leading to insufficient frames per process.

2. Insufficient Main memory

- If main memory is too small to hold the working sets of all active processes, page faults increase rapidly.

3. Global Page Replacement Algorithm

- In global replacement, a process can replace pages belonging to other processes. This causes:
    - Chain reaction of page faults.
    - One process affecting the performance of others.

4. Poor memory management policy

- Inefficient allocation of frames and poor page replacement strategies increase the probability of thrashing.

Thrashing and Degree of Multiprogramming

- Thrashing is directly related to the degree of multiprogramming.
- Initially, increasing multiprogramming improves CPU utilization.
- Beyond an optimal point, further increase causes thrashing.
- CPU utilization then decreases rapidly.

Graph explanation

- X-axis: Degree of multiprogramming
- Y-axis: CPU utilization
- Curve rises initially, reaches an optimal point, and then falls due to thrashing.

Methods to handle Thrashing

- Thrashing severely degrades system performance by increasing page faults and disk I/O.
- To prevent or control thrashing, the operating system uses several techniques aimed at reducing excessive paging and ensuring processes get enough memory frames.

---

1. Reducing the degree of multiprogramming

- The degree of multiprogramming refers to the number of processes residing in main memory at the same time.
- When too many processes are loaded simultaneously, the available memory frames are divided among them, resulting in insufficient frames per process.
- This shortage of frames causes frequent page faults, which leads to thrashing.

How the OS reduces multiprogramming

- The operating system suspends or swaps out some processes from memory.
- Suspended processes are moved to secondary storage.

- The remaining active processes receive more memory frames.

---

2. Using Local Page Replacement Algorithm

- In a local page replacement algorithm, each process is allowed to replace only its own pages and cannot take frames from other processes.
- This prevents a situation where one process causes page faults in other processes.

Why global replacement causes thrashing

- A process with high page faults may steal frames from others.
- Other processes then start faulting.
- A chain reaction of page faults occurs.
- This leads to system-wide thrashing.

Advantages

- Faults of one process do not affect others.
- System becomes more stable.
- Thrashing is localized and easier to control.

Limitation

- If a process is given too few frames, it may still thrash internally.

---

3. Working Set Model

- The working set of a process is the collection of pages that have been referenced during a recent time interval.
- It is based on the principle of locality of reference.

Explanation of working set model

- Each process requires a minimum number of frames equal to its working set size.
- The operating system calculates the working set using a window size Δ (delta).
- The working set is represented as $W(t, \Delta)$, where:
    - t is current time
    - Δ is window size

Relation to Thrashing

- If the total working set of all processes fits into available memory, the system runs efficiently.
- If the total working set exceeds available frames, thrashing occurs.

How it prevents thrashing

- OS admits a process into memory only if its working set can be fully allocated.
- Otherwise, the process is delayed or suspended.

---

4. Page Fault Frequency (PFF) Control

- Page Fault Frequency control is a dynamic technique used by the operating system to monitor and control the page fault rate of each process.

Working of PFF

- The OS defines an acceptable upper and lower limit for page fault frequency.
- If page fault rate exceeds the upper limit:
  - The process is allocated additional frames.
- If page fault rate falls below the lower limit:
  - Some frames may be removed and given to other processes.

Handling Thrashing

- If sufficient frames are not available:
  - The OS suspends one or more processes.
- This reduces memory load and page faults.
