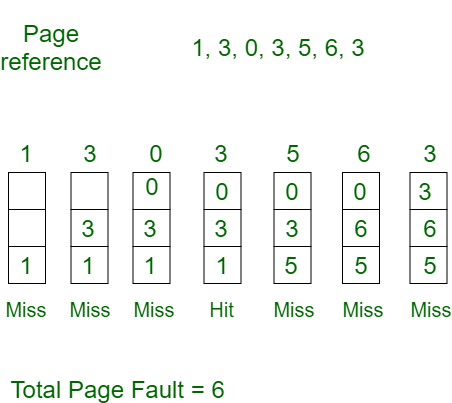
[**Other QB Click**](https://github.com/neeraj46665/BTech-CS-Notes/tree/main/BTech-2nd-year/sem-4)

1. a) FIFO replacement  
   b) LRU replacement

c)Optimal Page Replacement.-10

**1. First In First Out (FIFO):**This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

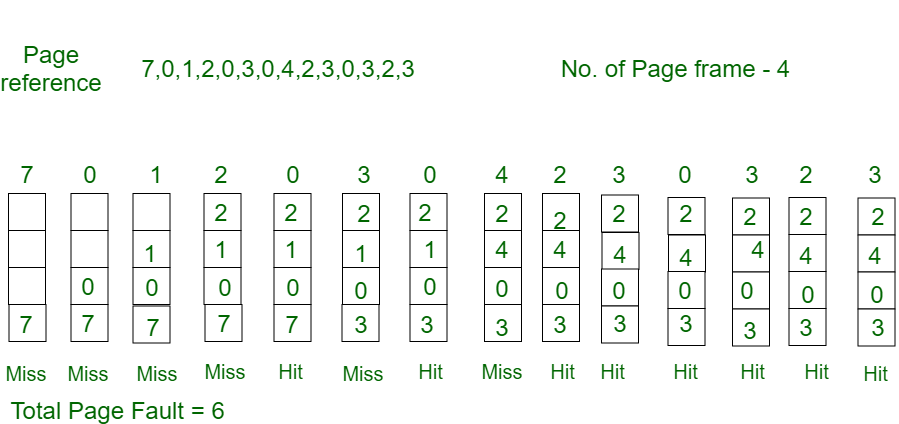
**Example 1:**Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames.Find the number of page faults.



Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**   
when 3 comes, it is already in memory so —> **0 Page Faults.** Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.** 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**Finally, when 3 come it is not available so it replaces 0 **1 page fault.**

**2. Optimal Page replacement:**In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

**Example-2:**Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

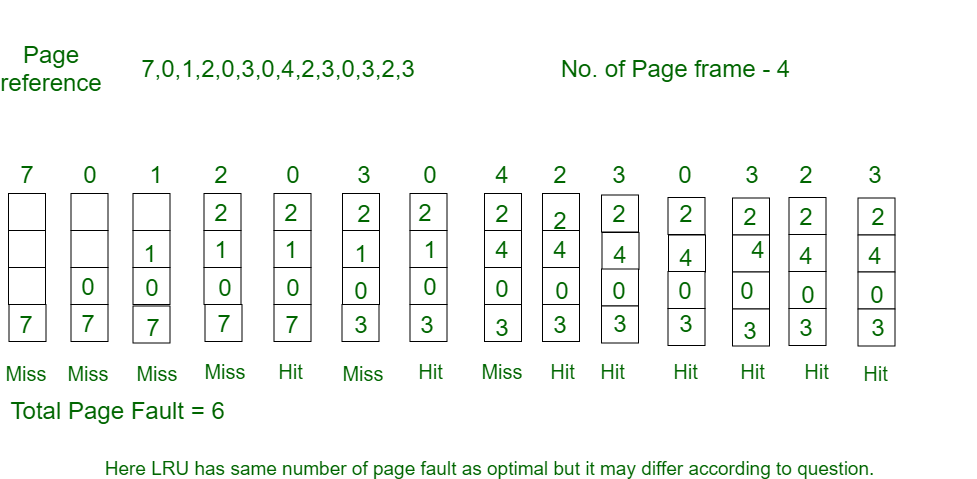


Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —>**4 Page faults**   
0 is already there so —> **0 Page fault.** when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.** 0 is already there so —>**0 Page fault.**4 will takes place of 1 —>**1 Page Fault.**

Now for the further page reference string —>**0 Page fault** because they are already available in the memory.   
Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

**3. Least Recently Used:**In this algorithm, page will be replaced which is least recently used.

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

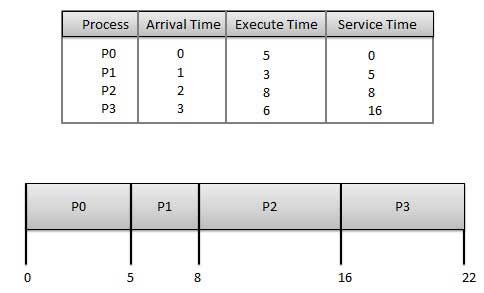


Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —>**4 Page faults**   
0 is already their so —> **0 Page fault.** when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**   
0 is already in memory so —>**0 Page fault**.   
4 will takes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

1. FCFS, SJF, RR-15

First Come First Serve (FCFS)

* Jobs are executed on first come, first serve basis.
* It is a non-preemptive, pre-emptive scheduling algorithm.
* Easy to understand and implement.
* Its implementation is based on FIFO queue.
* Poor in performance as average wait time is high.



**Wait time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time - Arrival Time** |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 8 - 2 = 6 |
| P3 | 16 - 3 = 13 |

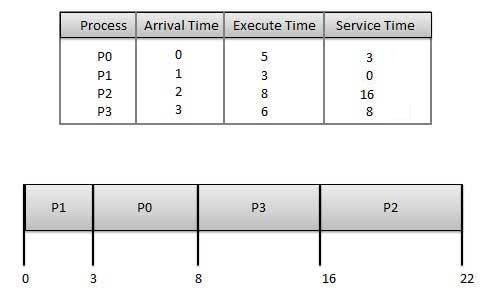
Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

* This is also known as **shortest job first**, or SJF
* This is a non-preemptive, pre-emptive scheduling algorithm.
* Best approach to minimize waiting time.
* Easy to implement in Batch systems where required CPU time is known in advance.
* Impossible to implement in interactive systems where required CPU time is not known.
* The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Execution Time** | **Service Time** |
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P3 | 3 | 6 | 8 |



**Waiting time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Waiting Time** |
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 14 - 2 = 12 |
| P3 | 8 - 3 = 5 |

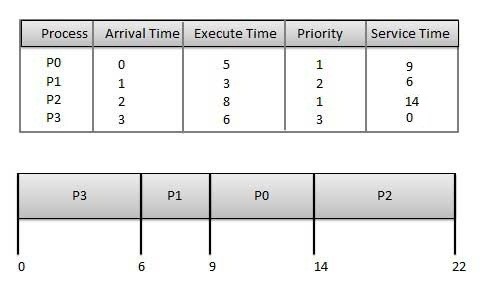
Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

* Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
* Each process is assigned a priority. Process with highest priority is to be executed first and so on.
* Processes with same priority are executed on first come first served basis.
* Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Execution Time** | **Priority** | **Service Time** |
| P0 | 0 | 5 | 1 | 0 |
| P1 | 1 | 3 | 2 | 11 |
| P2 | 2 | 8 | 1 | 14 |
| P3 | 3 | 6 | 3 | 5 |



**Waiting time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Waiting Time** |
| P0 | 0 - 0 = 0 |
| P1 | 11 - 1 = 10 |
| P2 | 14 - 2 = 12 |
| P3 | 5 - 3 = 2 |

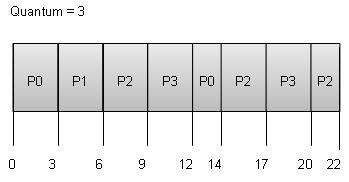
Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Shortest Remaining Time

* Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
* The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
* Impossible to implement in interactive systems where required CPU time is not known.
* It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

* Round Robin is the preemptive process scheduling algorithm.
* Each process is provided a fix time to execute, it is called a **quantum**.
* Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
* Context switching is used to save states of preempted processes.



**Wait time** of each process is as follows −

|  |  |
| --- | --- |
| **Process** | **Wait Time : Service Time - Arrival Time** |
| P0 | (0 - 0) + (12 - 3) = 9 |
| P1 | (3 - 1) = 2 |
| P2 | (6 - 2) + (14 - 9) + (20 - 17) = 12 |
| P3 | (9 - 3) + (17 - 12) = 11 |

Average Wait Time: (9+2+12+11) / 4 = 8.5

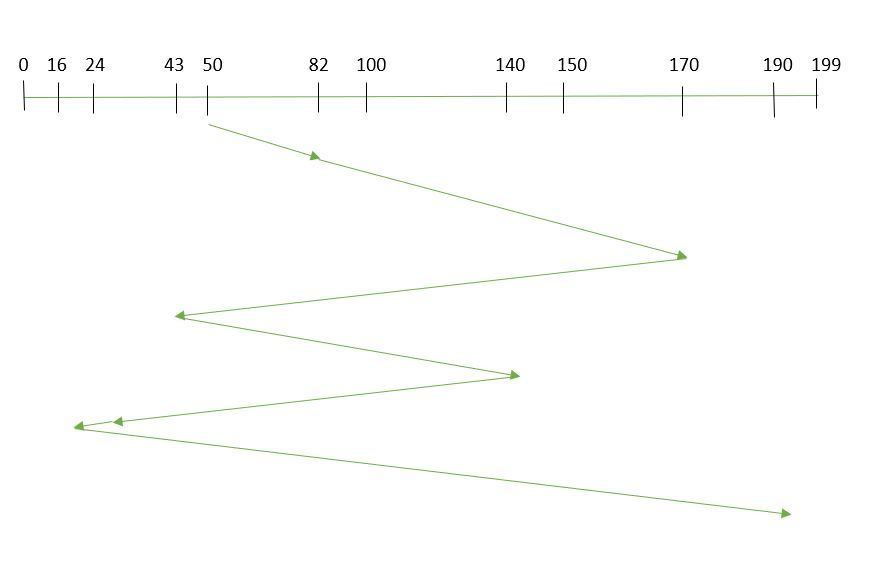
1. FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK-15

**Disk Scheduling Algorithms**

1. **FCFS:**FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.Let us understand this with the help of an example.

**Example:**

Suppose the order of request is- (82,170,43,140,24,16,190)  
And current position of Read/Write head is: 50



So, total overhead movement  (total distance covered by the disk arm) : =(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16) =642

Advantages:

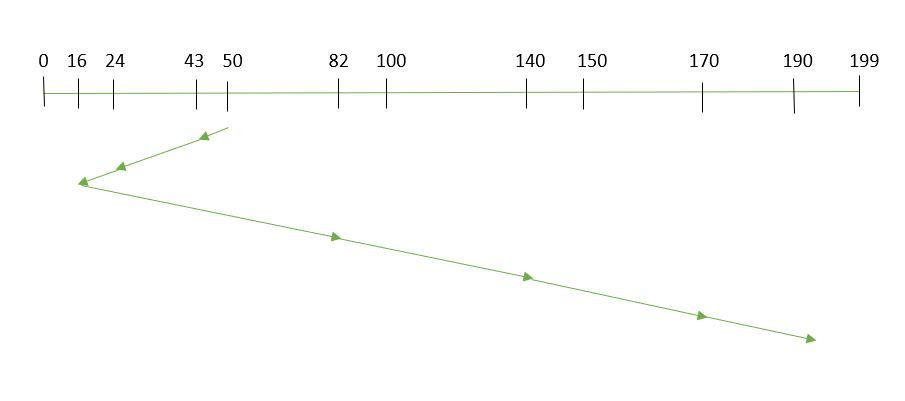
* Every request gets a fair chance
* No indefinite postponement

Disadvantages:

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

1. **SSTF:** In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.Let us understand this with the help of an example.

#### **Example:**

1. Suppose the order of request is- (82,170,43,140,24,16,190)  
   And current position of Read/Write head is : 50 

So,

total overhead movement (total distance covered by the disk arm) =(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170) =208

Advantages:

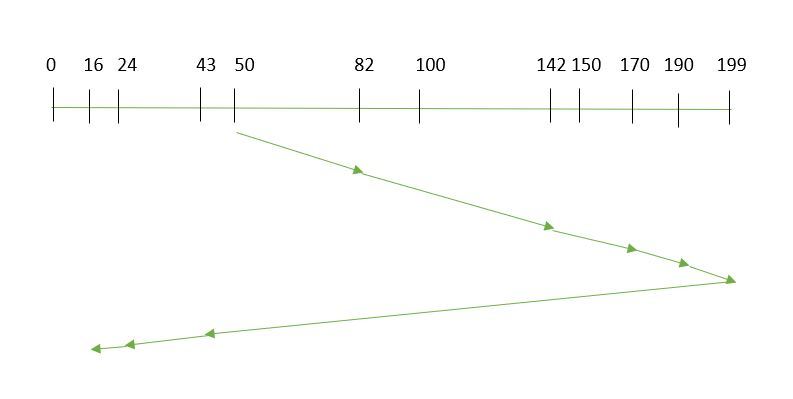
* Average Response Time decreases
* Throughput increases

Disadvantages:

* Overhead to calculate seek time in advance
* Can cause Starvation for a request if it has a higher seek time as compared to incoming requests
* High variance of response time as SSTF favors only some requests

**SCAN:**In SCAN algorithm the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and is hence also known as an **elevator algorithm.**As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

**Example:**

1. Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.** 

Therefore, the total overhead movement  (total distance covered by the disk arm)  is calculated as:

1. =(199-50)+(199-16) =332

Advantages:

* High throughput
* Low variance of response time
* Average response time

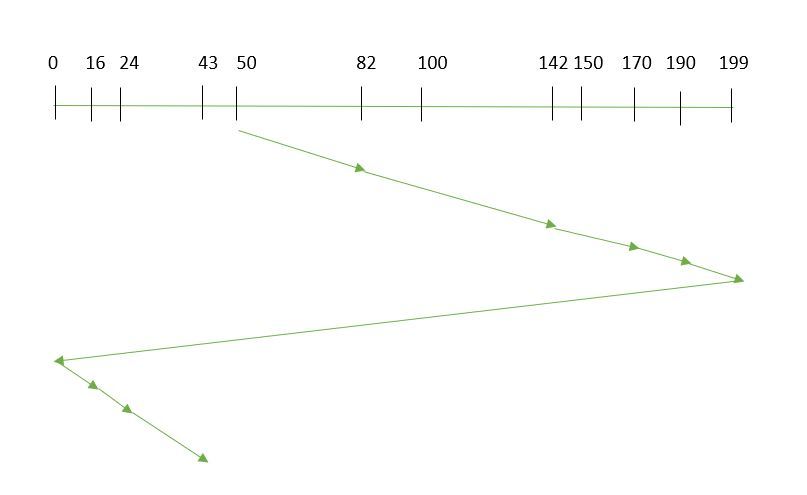
Disadvantages:

* Long waiting time for requests for locations just visited by disk arm

1. **CSCAN**: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in *CSCAN*algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

#### **Example:**

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.** 

so, the total overhead movement  (total distance covered by the disk arm) is calculated as:

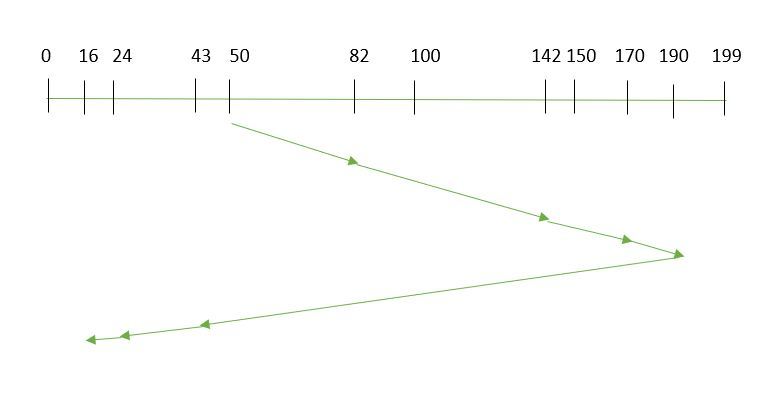
=(199-50)+(199-0)+(43-0) =391 Advantages:

* Provides more uniform wait time compared to SCAN

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

#### **Example:**

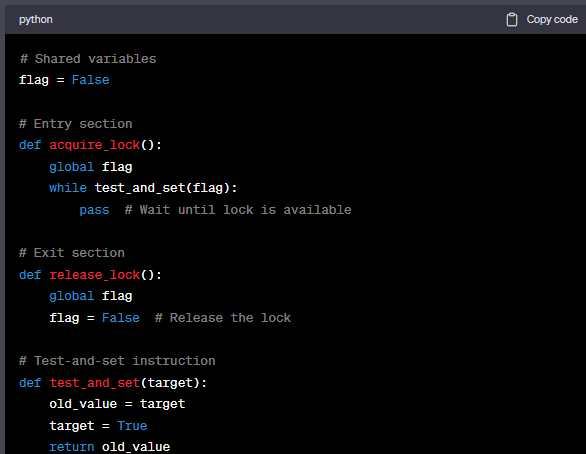
1. Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **“towards the larger value”.**



So, the total overhead movement  (total distance covered by the disk arm) is calculated as:

1. =(190-50)+(190-16) =314
2. **CLOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end’s last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.
3. Mutual-exclusion implementation with test and set() instruction.-15

The test-and-set instruction is a synchronization primitive that can be used to implement mutual exclusion in multi-threaded or multi-process environments. It provides an atomic read-modify-write operation that sets a flag while returning its previous value. Here's a simple implementation of mutual exclusion using the test-and-set instruction:



In the above example, the acquire\_lock() function is called by a thread or process when it wants to enter the critical section. It repeatedly calls the test\_and\_set() function until it successfully sets the flag to True, indicating that the lock has been acquired.

The release\_lock() function is called by the thread or process when it wants to exit the critical section. It simply sets the flag to False, releasing the lock for other threads or processes to acquire.

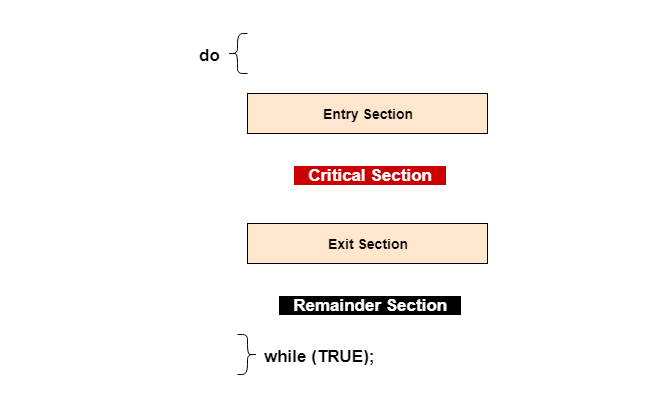
The test\_and\_set() function implements the test-and-set instruction. It atomically sets the target variable to True and returns its previous value. This ensures that only one thread or process can set the flag to True at a time, allowing mutual exclusion.

It's worth noting that this implementation assumes that the test-and-set instruction itself is atomic. In practice, atomic instructions are typically provided by the underlying hardware or operating system. The above code demonstrates the basic idea, but the actual implementation may vary depending on the specific programming language or platform being used.

1. Critical Section problem. Illustrate the software-based solution to the Critical Section problem.15

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.

1. The concept of Thrashing. What is the cause of Thrashing? How does the system detect Thrashing.-15

Thrashing is a phenomenon that occurs in computer systems when the system's performance drastically decreases due to excessive and inefficient use of system resources, particularly virtual memory. It typically happens when a system is overwhelmed with a high demand for memory but cannot efficiently allocate it to meet the demand. Instead, the system spends a significant amount of time and resources swapping data between physical memory (RAM) and the disk, resulting in a severe degradation of performance.

The primary cause of thrashing is an excessive number of page faults. A page fault happens when a program requests data that is not currently in physical memory and must be loaded from the disk. If a system is constantly experiencing page faults and continuously swapping pages in and out of the disk, it leads to thrashing.

Thrashing can occur due to several reasons:

Insufficient memory: When a system does not have enough physical memory to hold all the actively used data and programs, it relies heavily on swapping pages between the disk and RAM, leading to thrashing.

Overloaded system: If the system is running too many processes simultaneously, each requiring a significant amount of memory, it can cause excessive demand for memory resources and result in thrashing.

Poor memory management: Inefficient memory management algorithms or policies can contribute to thrashing. For example, if the system is using a paging algorithm that does not effectively prioritize frequently accessed pages, it may waste resources on unnecessary disk swaps, exacerbating thrashing.

Detecting thrashing can be challenging, but some common indicators include:

High page fault rate: Monitoring the number of page faults per unit of time can provide insights into thrashing. A significant increase in page faults might indicate that the system is struggling to keep up with memory demands.

Low CPU utilization: In a thrashing system, the CPU may be mostly idle because it spends most of its time waiting for page swaps to complete. If the CPU utilization remains consistently low despite the system being busy, it can be a sign of thrashing.

Excessive disk activity: Thrashing involves frequent swapping of pages between RAM and disk. Therefore, monitoring disk activity can reveal high disk usage due to constant page swapping.

Slow response time: If the system becomes unresponsive or experiences significant delays in executing tasks, it can be an indication of thrashing. This slowdown occurs because the system spends excessive time on disk I/O operations rather than processing tasks.

1. File Directories and their operation types.-10

File directories are a fundamental component of file systems that organize and manage files and folders within a storage system. They provide a hierarchical structure that allows users to locate and access files efficiently.

Here are some common directory operations:

Creating Directories:

mkdir: Creates a new directory in the file system.

mkdir -p: Creates parent directories recursively if they don't exist.

Listing Directory Contents:

ls: Lists the files and directories in a given directory.

ls -l: Provides a detailed listing, including file permissions, sizes, and timestamps.

Changing Directories:

cd: Changes the current working directory to the specified directory.

cd ..: Moves up one level in the directory hierarchy.

cd ~: Moves to the user's home directory.

Removing Directories:

rmdir: Deletes an empty directory.

rm -r: Removes a directory and its contents recursively.

Navigating Directories:

pwd: Prints the current working directory.

Renaming or Moving Directories:

mv: Renames or moves a directory to a new location.

Copying Directories:

cp -r: Copies a directory and its contents recursively to a new location.

Checking Directory Information:

stat: Displays detailed information about a directory, including size, permissions, and timestamps.

Changing Directory Permissions:

chmod: Modifies the permissions of a directory.

These operations help users manage and organize files and directories within a file system, enabling efficient storage and retrieval of data. The specific commands and syntax may vary depending on the operating system or file system being used.

1. similarities and dissimilarities (differences) between process and thread.10

## Difference between Process and Thread

The following table highlights the major differences between a process and a thread −

|  |  |  |
| --- | --- | --- |
| **Comparison Basis** | **Process** | **Thread** |
| Definition | A process is a program under execution i.e. an active program. | A thread is a lightweight process that can be managed independently by a scheduler |
| Context switching time | Processes require more time for context switching as they are heavier. | Threads require less time for context switching as they are lighter than processes. |
| Memory Sharing | Processes are totally independent and don’t share memory. | A thread may share some memory with its peer threads. |
| Communication | Communication between processes requires more time than between threads. | Communication between threads requires less time than between processes. |
| Blocked | If a process gets blocked, remaining processes can continue execution. | If a user level thread gets blocked, all of its peer threads also get blocked. |
| Resource Consumption | Processes require more resources than threads. | Threads generally need less resources than processes. |
| Dependency | Individual processes are independent of each other. | Threads are parts of a process and so are dependent. |
| Data and Code sharing | Processes have independent data and code segments. | A thread shares the data segment, code segment, files etc. with its peer threads. |
| Treatment by OS | All the different processes are treated separately by the operating system. | All user level peer threads are treated as a single task by the operating system. |
| Time for creation | Processes require more time for creation. | Threads require less time for creation. |
| Time for termination | Processes require more time for termination. | Threads require less time for termination. |

1. types of thread and their advantages, and disadvantages-10

Threads are lightweight units of execution within a process that enable concurrent execution of multiple tasks. They provide several advantages and disadvantages based on their types and usage. Here are some common types of threads along with their advantages and disadvantages:

Kernel Threads:

Advantages:

Kernel threads are managed and scheduled by the operating system.

They can run in parallel on multiple processors or processor cores.

Kernel threads provide strong isolation between threads, ensuring that one thread's issues do not affect others.

They can take advantage of the full range of operating system features and services.

Disadvantages:

Creation and management of kernel threads typically involve system calls, which can be more time-consuming than user-level threads.

Switching between kernel threads often incurs higher overhead due to the involvement of the operating system.

The number of kernel threads that can be created is typically limited by the operating system.

User-Level Threads:

Advantages:

User-level threads are managed by a user-level thread library or runtime system, which provides flexibility and control over scheduling and thread management.

Context switching between user-level threads is typically faster than kernel threads since it doesn't involve the operating system.

User-level threads can be customized to meet specific application requirements.

They can run on any operating system that supports the execution of a single thread.

Disadvantages:

User-level threads are typically not well-suited for parallel execution on multiple processors or processor cores since the operating system schedules them on a single kernel thread.

Blocking system calls made by one user-level thread can block the entire process and all its threads.

User-level threads do not provide strong isolation, and issues in one thread can affect the entire process.

They cannot take full advantage of operating system features or services.

Hybrid Threads:

Advantages:

Hybrid threads combine the advantages of both kernel threads and user-level threads.

They allow multiple user-level threads to be associated with a single kernel thread, enabling parallel execution on multiple processors or cores.

Hybrid threads provide a balance between control and efficiency by utilizing both user-level and kernel-level mechanisms.

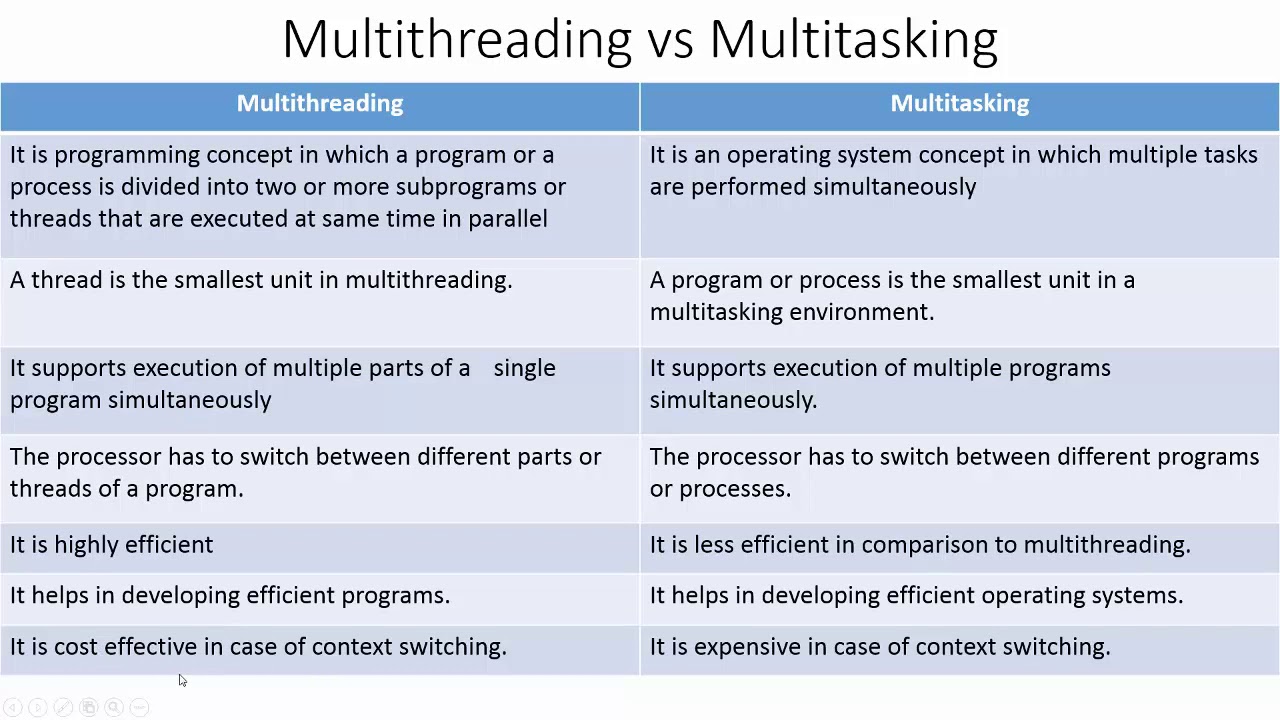
Disadvantages:

Hybrid thread implementations may require more complex programming models and management strategies.

Context switching between hybrid threads can involve both user-level and kernel-level operations, which may introduce additional overhead.

The effectiveness of hybrid threads depends on the underlying thread library and the level of integration with the operating system.

1. Multithreading and Multitasking.10



1. Multi-Programming and Multi-tasking systems.-10

| **Sr.no** | **Multiprogramming** | **Multi-tasking** |
| --- | --- | --- |
| 1. | It includes the single CPU to execute the program. | It uses multiple tasks for the task allocation. |
| 2. | Concept of Context Switching is used. | Concept of Context Switching and Time Sharing is used. |
| 3. | In multiprogrammed system, the operating system simply switches to, and executes, another job when current job needs to wait. | The processor is typically used in time sharing mode. Switching happens when either allowed time expires or where there other reason for current process needs to wait (example process needs to do IO). |
| 4. | Multi-programming increases CPU utilization by organizing jobs . | In multi-tasking also increases CPU utilization, it also increases responsiveness. |
| 5. | The idea is to reduce the CPU idle time for as long as possible. | The idea is to further extend the CPU Utilization concept by increasing responsiveness Time Sharing. |
| 6. | It uses job scheduling algorithms so that more than one program can run at the same time. | Time sharing mechanism is used so that multiple tasks can run at the same time. |
| 7. | In community edition, personalized shopping experiences is not created. | Promotions, personalized shopping experiences can be displayed in enterprise edition products. |
| 8. | Execution of process takes more time. | Execution of process takes less time. |

1. If there are 100 units of resource R in the system and each process in the system requires 4 units of resource R, then test how many processes can be present at maximum so that no deadlock will occur.-10

To calculate the maximum number of processes that can be present without causing a deadlock, we divide the total number of units of resource R by the resource requirement of each process:

Maximum number of processes = Total units of resource R / Resource requirement per process

= 100 units / 4 units per process

= 25 processes

Therefore, a maximum of 25 processes can be present in the system without causing a deadlock, given the available 100 units of resource R and the resource requirement of 4 units per process.

1. 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. LRU Page Replacement Algorithm  
   Which algorithm is better, according to you.-15
2. first fit, best fit and worst fit algorithms.-10
3. a) File attributes  
   b) File operations  
   c) File types  
   d) Internal file structure.-10

a) File attributes: File attributes refer to the characteristics or properties associated with a file. These attributes provide information about the file's type, permissions, size, creation/modification timestamps, ownership, and other metadata. Common file attributes include:

* File name: The name of the file, which identifies it within the file system.
* File extension: A part of the file name that indicates the file's type or format.
* File size: The size of the file in bytes or another appropriate unit.
* File permissions: Access permissions that determine who can read, write, or execute the file.
* File timestamps: Creation, modification, and access timestamps that record when the file was created, last modified, and last accessed.
* File ownership: The user and group ownership of the file.
* File attributes may also include additional information specific to the file system or operating system, such as file flags, encryption status, and file versioning.

b) File operations: File operations refer to the actions or tasks that can be performed on files within a file system. Common file operations include:

* Create: Create a new file in the file system.
* Open: Open an existing file for reading, writing, or both.
* Read: Retrieve the content of a file.
* Write: Modify or append data to a file.
* Close: Release the resources associated with an open file.
* Delete: Remove a file from the file system.
* Rename: Change the name of a file.
* Copy: Create a duplicate of a file in a different location.
* Move: Move a file from one location to another.
* Seek: Change the position within a file for reading or writing.
* Truncate: Resize a file to a specified length.
* Lock: Apply locks to a file to prevent concurrent access.
* Unlock: Release locks applied to a file.

c) File types: File types refer to the categorization or classification of files based on their content, format, or purpose. Some common file types include:

* Text files: Files containing plain text without any specific formatting or binary data.
* Binary files: Files containing non-textual data, such as executable programs or multimedia files.
* Document files: Files created and used by word processors or document editing software.
* Spreadsheet files: Files containing tabular data used by spreadsheet applications.
* Image files: Files storing graphical images in various formats (e.g., JPEG, PNG, GIF).
* Audio files: Files containing audio data (e.g., MP3, WAV).
* Video files: Files containing video data (e.g., MP4, AVI).
* Archive files: Files that contain compressed or multiple files bundled together (e.g., ZIP, TAR).
* Configuration files: Files containing settings and configurations for applications or systems (e.g., XML, INI).
* Database files: Files used to store structured data in a database format (e.g., SQLite, MySQL).

d) Internal file structure: The internal file structure refers to how the contents of a file are organized and stored within the file system. The internal structure can vary based on the file system implementation and file type. Some common internal file structures include:

* Sequential file structure: Data is stored sequentially in the order it is written. It allows efficient reading of data in a sequential manner.
* Random access file structure: Data is organized in a way that allows direct access to any part of the file. Random access enables efficient reading and writing of data at specific positions within the file.
* Indexed file structure: A separate index or lookup table is maintained alongside the file, allowing efficient access to data based on a key or identifier. This structure is commonly used in database systems.
* Linked file structure: Data is stored in linked blocks or nodes, forming a linked list-like structure. It

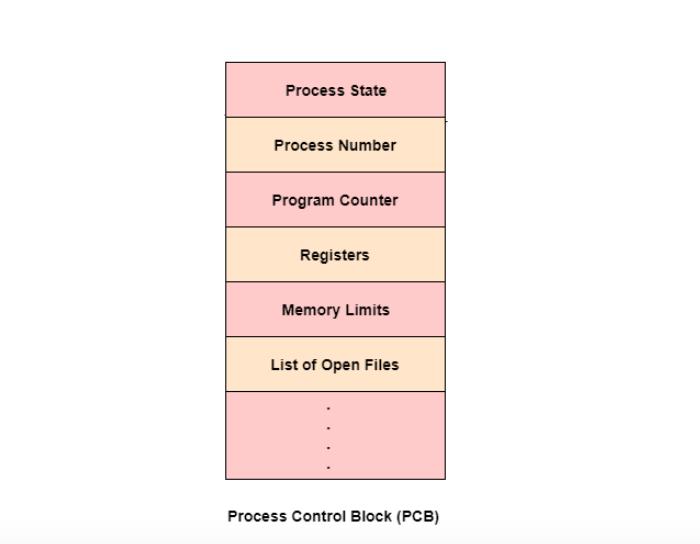
1. Process Control Block.-10

Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

Structure of the Process Control Block

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram –



Process Control Block in Operating System

The following are the data items −

Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

This shows the number of the particular process.

Program Counter

This contains the address of the next instruction that needs to be executed in the process.

Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

List of Open Files

These are the different files that are associated with the process

CPU Scheduling Information

The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

Memory Management Information

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

I/O Status Information

This information includes the list of I/O devices used by the process, the list of files etc.

Accounting information

The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

Location of the Process Control Block

The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

1. System Components.-10

There are various components of an Operating System to perform well defined tasks. Though most of the Operating Systems differ in structure but logically they have similar components. Each component must be a well-defined portion of a system that appropriately describes the functions, inputs, and outputs.

There are following 8-components of an Operating System:

1. Process Management
2. I/O Device Management
3. File Management
4. Network Management
5. Main Memory Management
6. Secondary Storage Management
7. Security Management
8. Command Interpreter System

Following section explains all the above components in more detail:

Process Management

A process is program or a fraction of a program that is loaded in main memory. A process needs certain resources including CPU time, Memory, Files, and I/O devices to accomplish its task. The process management component manages the multiple processes running simultaneously on the Operating System.

A program in running state is called a process.

The operating system is responsible for the following activities in connection with process management:

* Create, load, execute, suspend, resume, and terminate processes.
* Switch system among multiple processes in main memory.
* Provides communication mechanisms so that processes can communicate with each others
* Provides synchronization mechanisms to control concurrent access to shared data to keep shared data consistent.
* Allocate/de-allocate resources properly to prevent or avoid deadlock situation.

I/O Device Management

One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. I/O Device Management provides an abstract level of H/W devices and keep the details from applications to ensure proper use of devices, to prevent errors, and to provide users with convenient and efficient programming environment.

Following are the tasks of I/O Device Management component:

* Hide the details of H/W devices
* Manage main memory for the devices using cache, buffer, and spooling
* Maintain and provide custom drivers for each device.

File Management

File management is one of the most visible services of an operating system. Computers can store information in several different physical forms; magnetic tape, disk, and drum are the most common forms.

A file is defined as a set of correlated information and it is defined by the creator of the file. Mostly files represent data, source and object forms, and programs. Data files can be of any type like alphabetic, numeric, and alphanumeric.

A files is a sequence of bits, bytes, lines or records whose meaning is defined by its creator and user.

The operating system implements the abstract concept of the file by managing mass storage device, such as types and disks. Also files are normally organized into directories to ease their use. These directories may contain files and other directories and so on.

The operating system is responsible for the following activities in connection with file management:

* File creation and deletion
* Directory creation and deletion
* The support of primitives for manipulating files and directories
* Mapping files onto secondary storage
* File backup on stable (nonvolatile) storage media

Network Management

The definition of network management is often broad, as network management involves several different components. Network management is the process of managing and administering a computer network. A computer network is a collection of various types of computers connected with each other.

Network management comprises fault analysis, maintaining the quality of service, provisioning of networks, and performance management.

Network management is the process of keeping your network healthy for an efficient communication between different computers.

Following are the features of network management:

* Network administration
* Network maintenance
* Network operation
* Network provisioning
* Network security

Main Memory Management

Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.

Main memory is a volatile storage device which means it loses its contents in the case of system failure or as soon as system power goes down.

The main motivation behind Memory Management is to maximize memory utilization on the computer system.

The operating system is responsible for the following activities in connections with memory management:

* Keep track of which parts of memory are currently being used and by whom.
* Decide which processes to load when memory space becomes available.
* Allocate and deallocate memory space as needed.

Secondary Storage Management

The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in main memory during execution. Since the main memory is too small to permanently accommodate all data and program, the computer system must provide secondary storage to backup main memory.

Most modern computer systems use disks as the principle on-line storage medium, for both programs and data. Most programs, like compilers, assemblers, sort routines, editors, formatters, and so on, are stored on the disk until loaded into memory, and then use the disk as both the source and destination of their processing.

The operating system is responsible for the following activities in connection with disk management:

* Free space management
* Storage allocation

Disk scheduling

Security Management

The operating system is primarily responsible for all task and activities happen in the computer system. The various processes in an operating system must be protected from each other’s activities. For that purpose, various mechanisms which can be used to ensure that the files, memory segment, cpu and other resources can be operated on only by those processes that have gained proper authorization from the operating system.

Security Management refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls to be imposed, together with some means of enforcement.

For example, memory addressing hardware ensure that a process can only execute within its own address space. The timer ensure that no process can gain control of the CPU without relinquishing it. Finally, no process is allowed to do it’s own I/O, to protect the integrity of the various peripheral devices.

Command Interpreter System

One of the most important component of an operating system is its command interpreter. The command interpreter is the primary interface between the user and the rest of the system.

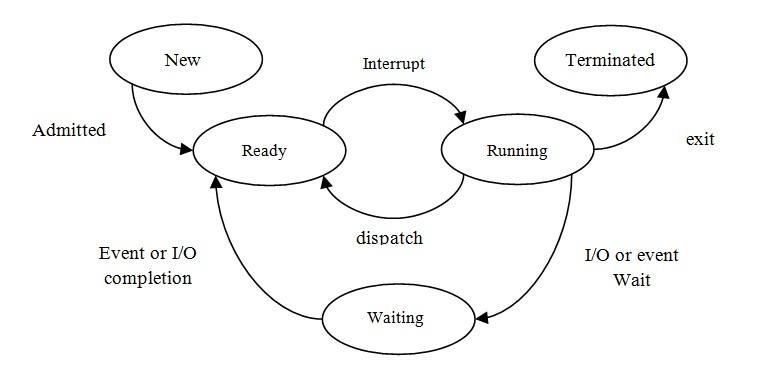
Command Interpreter System executes a user command by calling one or more number of underlying system programs or system calls.

Command Interpreter System allows human users to interact with the Operating System and provides convenient programming environment to the users.

Many commands are given to the operating system by control statements. A program which reads and interprets control statements is automatically executed. This program is called the shell and few examples are Windows DOS command window, Bash of Unix/Linux or C-Shell of Unix/Linux.

1. Banker’s algorithm for deadlock avoidance.-10
2. process state. Explain the state transition diagram.-10

**shows the state transition diagram for the process states defined above:**

Logically, the first two state are similar. In both case the process is willing to run, but in the ready state there is no CPU temporarily available for it.

**(I) Running to ready state:**

* A process in the running state has all of the resources that it needs for further execution, including a processor.
* The long term scheduler picks up a new process from second memory and loads it into the main memory when there are sufficient resources available.
* The process is now in ready state, waiting for its execution.

**(II) waiting to ready:**

* Process waiting for some event such as completion of I/O operation, synchronization signal, etc.
* A process moves from waiting state to ready state if the event the  
  process has been waiting for, occurs.
* The process is now ready for execution.

**(III) Running to waiting:**

* The process in the main memory that is waiting for some event.
* A process is put in the waiting state if it must wait for some event. For example, the process may request some resources or memory which might not be available.
* The process may be waiting for an I/O operation or it may be waiting for some other process to finish before it can continue execution.

**(IV) blocked to ready:**

* The process is in secondary memory but not yet ready for execution.
* The process moves from Blocked to Ready state if the event, the process has been waiting for occurs.

**(v) Running to terminated:**

* The process has finished execution.
* The OS moves a process from running state to terminated state if the process finishes execution or if it aborts.
* Whenever the execution of a process is completed in running state, it will exit to terminate state, which is the completion of process.

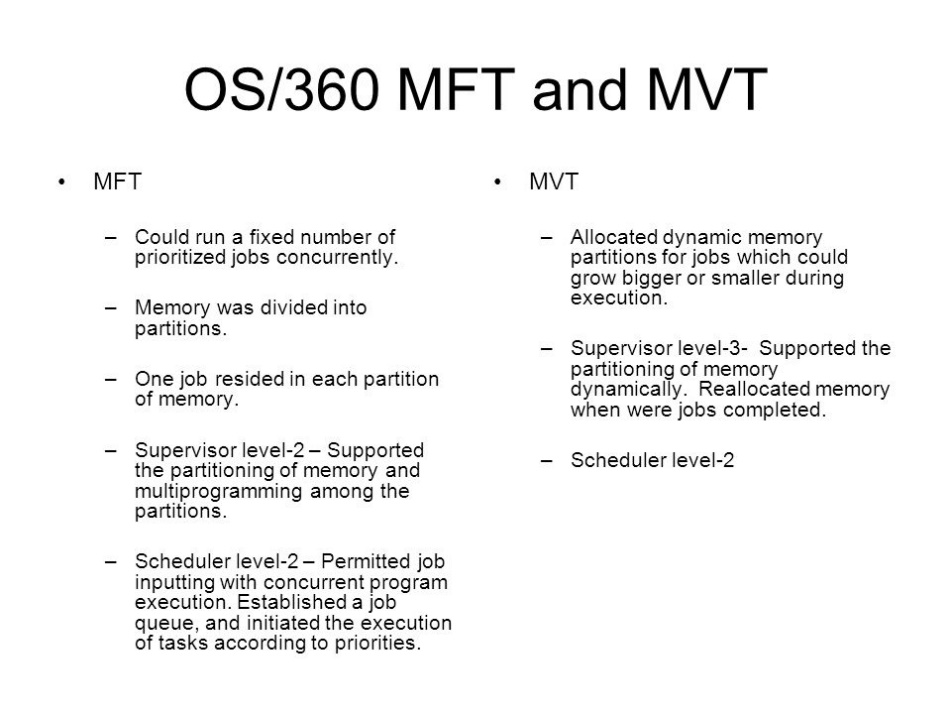
1. MFT and MVT.-5

1)MFT or fixed partitioning Scheme:

* In fixed scheme, the OS will be divided into fixed sized blocks. It takes place at the time of installation.
* At compile time, we can bind only addresses
* Degree of multiprogramming is not flexible. This is because the number of blocks is fixed resulting in memory wastage due to fragmentation.

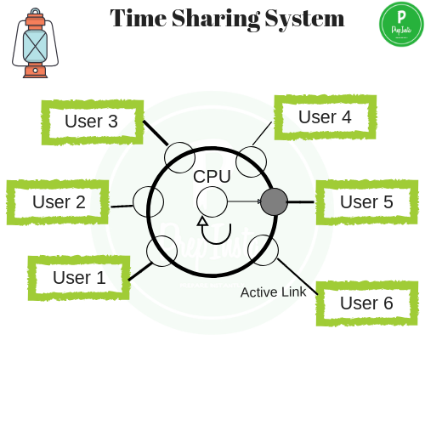
2)MVT OR variable partitioning Scheme:

* In variable partitioning scheme there are no partitions at the beginning.
* There is only the OS area and the rest of the available RAM.
* The memory is allocated to the processes as they enter.
* This method is more flexible as there is no internal fragmentation and there is no size limitation.
* Compile time address binding is impossible because of external fragmentation.



1. features of the Time-Sharing System.-5

* Every user gets a dedicated time for the operation.
* Simultaneous tasks are carried out at once.
* Tasks no longer have to wait for the previous task to finish to get the processor.
* Quick processing of multiple tasks.
* Equal time given to all the processes so that they operate smoothly without any significant delay.



### **Advantages**

Some of the advantages of the time-sharing system are as follows:

* Quick response to the users.
* No duplication of data.
* No duplication of software applications.
* Reduces CPU idle time.

1. The difference between Process and programme.-5

|  |  |
| --- | --- |
| **Program** | **Process** |
| It is a set of instructions that has been designed to complete a certain task. | It is an instance of a program that is being currently executed. |
| It is a passive entity. | It is an active entity. |
| It resides in the secondary memory of the system. | It is created when a program is in execution and is being loaded into the main memory. |
| It exists in a single place and continues to exist until it has been explicitly deleted. | It exists for a limited amount of time and it gets terminated once the task has been completed. |
| It is considered as a static entity. | It is considered as a dynamic entity. |
| It doesn't have a resource requirement. | It has a high resource requirement. |
| It requires memory space to store instructions. | It requires resources such as CPU, memory address, I/O during its working. |
| It doesn't have a control block. | It has its own control block, which is known as Process Control Block. |

1. generations of operating system detail.-5

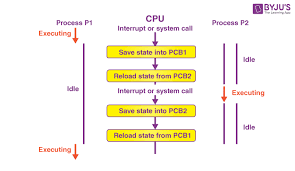
* First Generation (1951-1959): These were the earliest operating systems, and they were mainly used to manage batch processing systems.
* Second Generation (1959-1965): This generation of operating systems added support for time-sharing, which allowed multiple users to access the system simultaneously.
* Third Generation (1965-1971): This generation of operating systems introduced the concept of virtual memory, which allowed programs to run larger and more complex applications.
* Fourth Generation (1971-1980): This generation saw the emergence of personal computers and the development of the first personal computer operating systems, such as CP/M.
* Fifth Generation (1980-present): This generation of operating systems includes modern operating systems, such as Windows, macOS, and various distributions of Linux, which provide a graphical user interface, support for multi-tasking, and a wide range of advanced features.

1. Real-Time Operating System.-5

A real-time operating system (RTOS) is an operating system specifically designed for applications that require precise and deterministic timing and responsiveness. It is used in systems where the execution of tasks must meet strict deadlines and where failure to meet these deadlines can have serious consequences, such as in industrial control systems, medical devices, aerospace systems, and automotive applications.

Key features of a real-time operating system include:

1. Deterministic scheduling: RTOSs provide deterministic scheduling algorithms that guarantee a task will run within a specified time frame or deadline. This ensures that critical tasks are executed in a timely manner.
2. Priority-based scheduling: RTOSs typically use priority-based scheduling to determine the order in which tasks are executed. Higher-priority tasks preempt lower-priority tasks, ensuring that critical tasks receive immediate attention.
3. Fast context switching: RTOSs have efficient context switching mechanisms that allow tasks to quickly transition from one task to another. This enables the system to respond rapidly to real-time events.
4. Interrupt handling: RTOSs have efficient interrupt handling mechanisms to respond to external events or interrupts in a timely manner. Interrupt service routines (ISRs) can preempt the currently executing task to handle the urgent event.
5. Resource management: RTOSs provide mechanisms for managing system resources, such as memory, CPU time, and I/O devices. These mechanisms ensure that tasks are allocated the necessary resources and prevent resource contention.
6. Deterministic I/O operations: RTOSs often provide specialized I/O mechanisms that allow for predictable and time-bound communication with external devices. This is crucial for real-time systems that require precise control over I/O operations.
7. Minimal latency: RTOSs strive to minimize interrupt and task switching latency to ensure that critical tasks can meet their deadlines. This involves reducing the time between an event occurring and the system responding to it.
8. Small footprint: RTOSs are typically designed to have a small memory footprint and low overhead to maximize the available resources for application tasks.
9. context switching with a diagram.-5



The Context switching is a technique or method used by the operating system to switch a process from one state to another to execute its function using CPUs in the system. When switching perform in the system, it stores the old running process's status in the form of registers and assigns the [CPU](https://www.javatpoint.com/cpu-full-form) to a new process to execute its tasks. While a new process is running in the system, the previous process must wait in a ready queue. The execution of the old process starts at that point where another process stopped it. It defines the characteristics of a multitasking operating system in which multiple processes shared the same [CPU](https://www.javatpoint.com/central-processing-unit) to perform multiple tasks without the need for additional processors in the system.

A context switching helps to share a single CPU across all processes to complete its execution and store the system's tasks status. When the process reloads in the system, the execution of the process starts at the same point where there is conflicting.

The following steps are taken when switching Process P1 to Process 2:

1. First, thes context switching needs to save the state of process P1 in the form of the program counter and the registers to the PCB (Program Counter Block), which is in the running state.
2. Now update PCB1 to process P1 and moves the process to the appropriate queue, such as the ready queue, I/O queue and waiting queue.
3. After that, another process gets into the running state, or we can select a new process from the ready state, which is to be executed, or the process has a high priority to execute its task.
4. Now, we have to update the PCB (Process Control Block) for the selected process P2. It includes switching the process state from ready to running state or from another state like blocked, exit, or suspend.
5. If the CPU already executes process P2, we need to get the status of process P2 to resume its execution at the same time point where the system interrupt occurs.
6. Disk Arm Scheduling Algorithm.-5

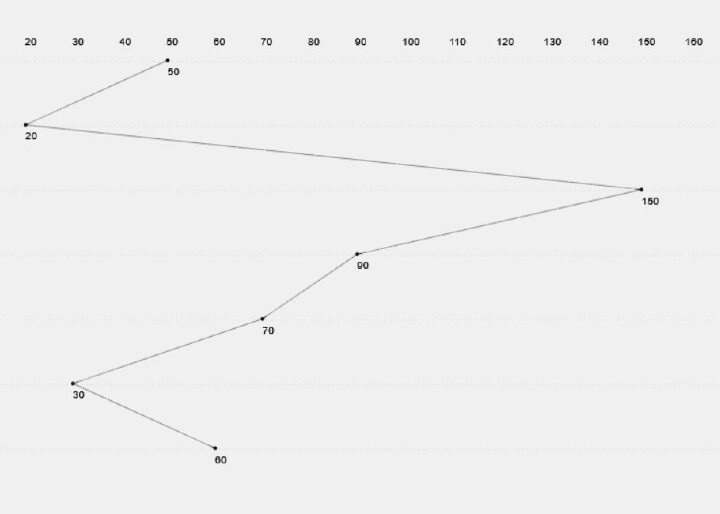
Disc scheduling is an important process in operating systems that determines the order in which disk access requests are serviced. The objective of disc scheduling is to minimize the time it takes to access data on the disk and to minimize the time it takes to complete a disk access request. Disk access time is determined by two factors: seek time and rotational latency. Seek time is the time it takes for the disk head to move to the desired location on the disk, while rotational latency is the time taken by the disk to rotate the desired data sector under the disk head. Disk scheduling algorithms are an essential component of modern operating systems and are responsible for determining the order in which disk access requests are serviced. The primary goal of these algorithms is to minimize disk access time and improve overall system performance.

## First-Come-First-Serve

The First-Come-First-Served (FCFS) disk scheduling algorithm is one of the simplest and most straightforward disk scheduling algorithms used in modern operating systems. It operates on the principle of servicing disk access requests in the order in which they are received. In the FCFS algorithm, the disk head is positioned at the first request in the queue and the request is serviced. The disk head then moves to the next request in the queue and services that request. This process continues until all requests have been serviced.

### **Example**

Suppose we have an order of disk access requests: 20 150 90 70 30 60. The disk head is −



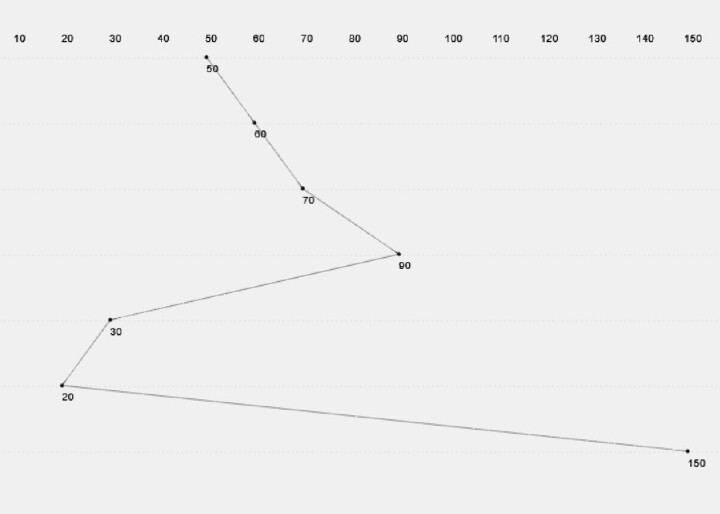
currently located at track 50.

The total seek time = (50-20) + (150-20) + (150-90) + (90-70) + (70-30) + (60-30) = 310

## Shortest-Seek-Time-First

Shortest Seek Time First (SSTF) is a disk scheduling algorithm used in operating systems to efficiently manage disk I/O operations. The goal of SSTF is to minimize the total seek time required to service all the disk access requests. In SSTF, the disk head moves to the request with the shortest seek time from its current position, services it, and then repeats this process until all requests have been serviced. The algorithm prioritizes disk access requests based on their proximity to the current position of the disk head, ensuring that the disk head moves the shortest possible distance to service each request.

### **Example**

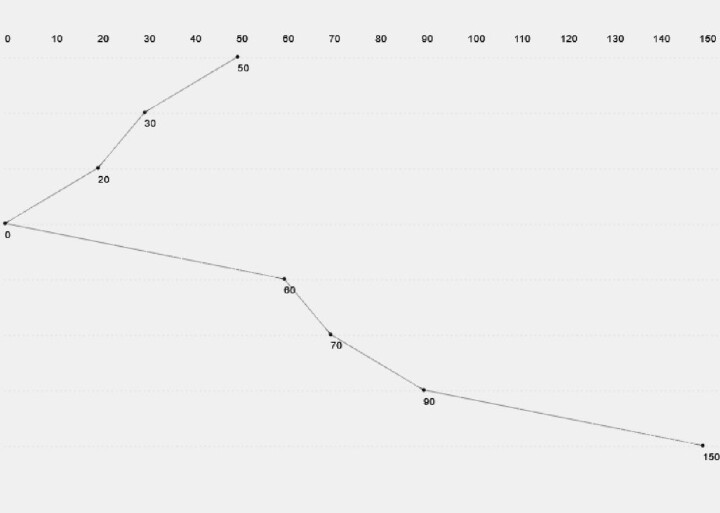


In this case, for the same order of success request, the total seek time = (60-50) + (70-60) + (90-70) + (90-30) + (30-20) + (150-20) = 240

## SCAN

SCAN (Scanning) is a disk scheduling algorithm used in operating systems to manage disk I/O operations. The SCAN algorithm moves the disk head in a single direction and services all requests until it reaches the end of the disk, and then it reverses direction and services all the remaining requests. In SCAN, the disk head starts at one end of the disk, moves toward the other end, and services all requests that lie in its path. Once the disk head reaches the other end, it reverses direction and services all requests that it missed on the way. This continues until all requests have been serviced.

### **Example**



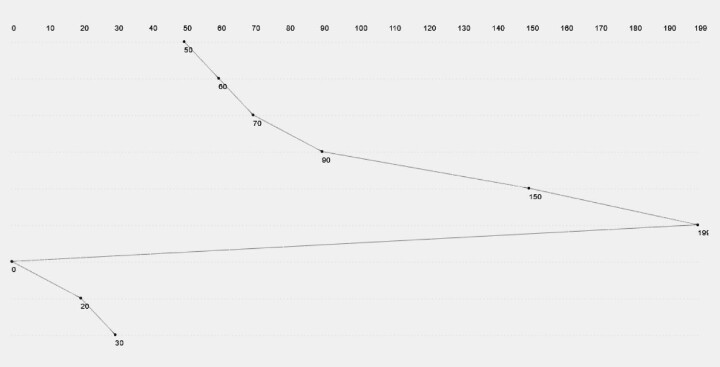
If we consider that the head direction is left in case of SCAN, the total seek time = (50-30) + (30-20) + (20-0) + (60-0) + (60-70) + (90-70) + (90-150) = 200

## C-SCAN

The C-SCAN (Circular SCAN) algorithm operates similarly to the SCAN algorithm, but it does not reverse direction at the end of the disk. Instead, the disk head wraps around to the other end of the disk and continues to service requests. This algorithm can reduce the total distance the disk head must travel, improving disk access time. However, this algorithm can lead to long wait times for requests that are made near the end of the disk, as they must wait for the disk head to wrap around to the other end of the disk before they can be serviced. The C-SCAN algorithm is often used in modern operating systems due to its ability to reduce disk access time and improve overall system performance.

### **Example**

For C-SCAN, the total seek time = (60-50) + (70-60) + (90-70) + (150-90) + (199-150) + (199-0) + (20-0) + (30-20) = 378

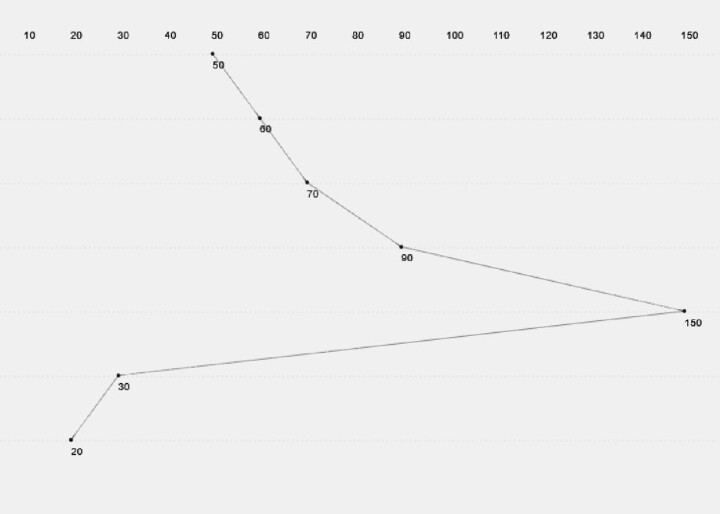


## LOOK

The LOOK algorithm is similar to the SCAN algorithm but stops servicing requests as soon as it reaches the end of the disk. This algorithm can reduce the total distance the disk head must travel, improving disk access time. However, this algorithm can lead to long wait times for requests that are made near the end of the disk, as they must wait for the disk head to wrap around to the other end of the disk before they can be serviced. The LOOK algorithm is often used in modern operating systems due to its ability to reduce disk access time and improve overall system performance.

### **Example**

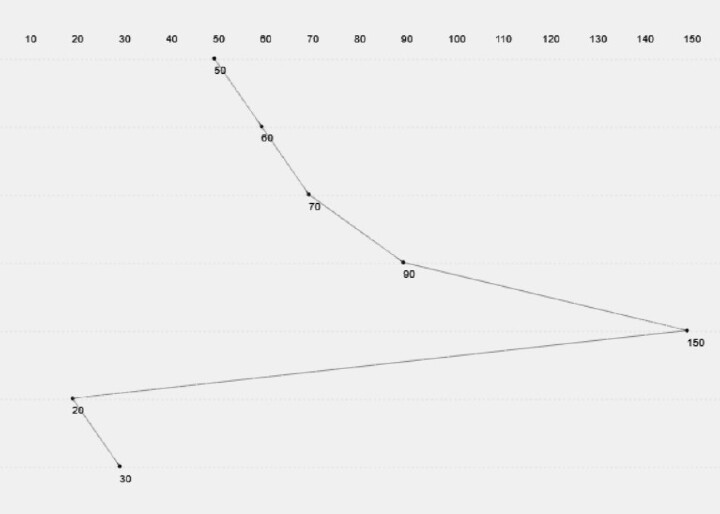
Considering the head direction is right, in this case, the total seek time = (60-50) + (70-60) + (90-70) + (150-90) + (150-30) + (30-20) = 230



## C-LOOK

C-LOOK is similar to the C-SCAN disk scheduling algorithm. In this algorithm, goes only to the last request to be serviced in front of the head in spite of the disc arm going to the end, and then from there it goes to the other end’s last request. Thus, it also prevents the extra delay which might occur due to unnecessary traversal to the end of the disk.

### **Example**



For the C-LOOK algorithm, the total seek time = (60-50) + (70-60) + (90-70) + (150-90) + (150-20) + (30-20) = 240