Paging

One of the techniques which are used for memory management is paging. In paging, processes are divided into pages and main memory is divided into frames. Pages of a process are loaded into frames of main memory when required.

Page Replacement Algorithm is used when a page fault occurs. Page Fault means the page referenced by the CPU is not present in the main memory.

Different Page Replacement Algorithms suggest different ways to decide which page is to be replaced. The main objective of these algorithms is to reduce the number of page faults.

Page Replacement Algorithms

First In First Out (FIFO) -This algorithm is similar to the operations of the queue. All the pages are stored in the queue in the order they are allocated frames in the main memory. The one which is allocated first stays in the front of the queue. The one which is allocated the memory first is replaced first. The one which is at the front of the queue is removed at the time of replacement.

Optimal Page Replacement - In this algorithm, the page which would be used after the longest interval is replaced. In other words, the page which is farthest to come in the upcoming sequence is replaced.

Least Recently Used - This algorithm works on previous data. The page which is used the earliest is replaced or which appears the earliest in the sequence is replaced.

Exercise :1

The library's virtual system is divided into pages of 2 KB each. The system can address up to 6 KB of memory using logical addresses and has a physical memory capacity of 4 KB. The page table maps logical pages to physical frames.

Tasks:

1. Calculate no.of frames and no.of pages
2. Determine the page number and offset for each logical address.
3. Use the page table to find the corresponding frame number.

Note: Exercise 2 & 3 are to be executed using WSL Terminal not on Tessellator

Exercise :2

Alex Works as a systems administrator at DataSecure Inc., a company that deals with sensitive customer data. Recently, there have been concerns about unauthorized access to critical system files and directories. To address this, Alex has decided to implement a monitoring solution that tracks the following information for each file:

1. File type
2. Number of links
3. Read, write, and execute permissions
4. Time of last access

Write a C implementation for the same by accepting one or more file or directory names as command line input .

Exercise:3

Alice is a system administrator at a large company. Recently, she has noticed that the server's storage is filling up quickly, and she suspects there might be some large files hidden deep within the directory structure that are taking up a lot of space. To identify these files, Alice wants to list all files along with their inode numbers and file names. She decides to write a C program for this purpose.

Practice-1

You are a systems engineer at MemoryOptimize Inc., a company specializing in developing high-efficiency memory management systems for operating systems. Your team has been testing different page replacement algorithms to determine the best one for the upcoming OS release.The system has 3 page frames available. Here is the page reference string that was used in the test:

1, 2, 3, 4, 2, 1, 5, 1, 2, 3, 4, 5

Your colleague provides the following description of the page replacement behavior:

1. The first three page requests (1, 2, 3) fill the available frames.
2. When the fourth page (4) is requested, it replaces the page (1) that was loaded into memory.
3. Subsequent requests for pages that are already in memory (2, 1) do not cause any replacements.
4. When a new page (5) is requested, it replaces page (2) that was loaded into memory

Based on the described behavior, determine which page replacement algorithm was used and analyze how many pages are being replaced in memory.

Practice-2

Imagine you are using a web browser that supports multiple tabs, and you have several tabs open simultaneously. Each tab represents a page in memory that you've accessed recently. If you open more tabs than your browser can display at once (exceeding the available memory), the browser needs to manage which tabs to keep in memory. Determine which page replacement algorithm was used and analyze how many pages are being replaced in memory.

Practice-3

Imagine you are managing a library with limited shelf space for books ,each capable of holding a fixed number of books.There are more books in the library than there are shelves, necessitating a system to decide which books to keep on the shelves for quick access. When a new book needs to be placed on the shelf (cache miss), algorithm would replace the book that is least likely to be requested again, based on perfect knowledge of future borrowing patterns. When a new book needs to be placed on the shelf (cache miss), the algorithm would replace the book that is least likely to be requested again in the future, based on perfect knowledge of future borrowing patterns.

Unit V

Topic-1 : Mass Storage Structure

Systems designed to store enormous volumes of data are referred to as mass storage devices. The basic idea of Mass Storage is to create a Data Backup or Data Recovery System.

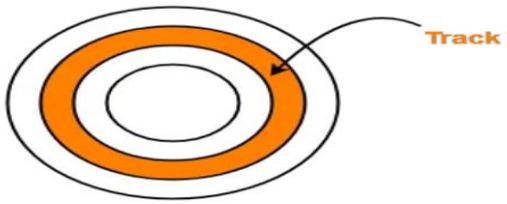
The Mass Storage Structure Devices are:

1. Magnetic Disks
2. Solid State Disks
3. Magnetic Tapes

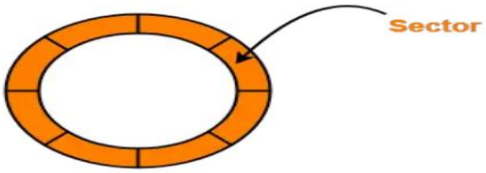
Magnetic Disk

Magnetic disk is a storage device that is used to write, rewrite and access data. It uses a magnetization process.

- The entire disk is divided into **platters**.
- Each platter consists of concentric circles called as **tracks**.
- These tracks are further divided into **sectors** which are the smallest divisions in the disk.

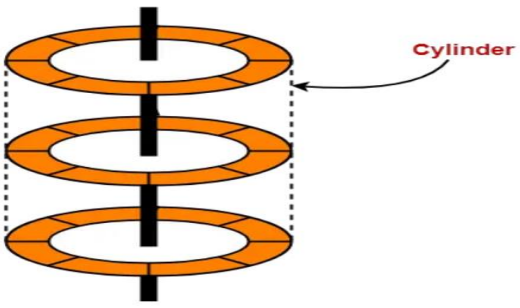


Disk divided into tracks

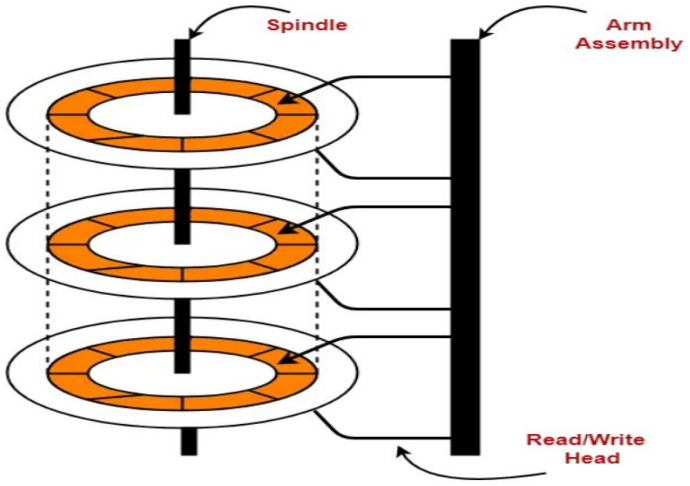


Track divided into sectors

- A **cylinder** is formed by combining the tracks at a given radius of a disk pack.



- There exists a mechanical arm called as **Read / Write head**.
- It is used to read from and write to the disk.
- Head has to reach at a particular track and then wait for the rotation of the platter.
- The rotation causes the required sector of the track to come under the head.
- Each platter has 2 surfaces- top and bottom and both the surfaces are used to store the data.
- Each surface has its own read / write head.



The access time of a record on a disk includes three components such as seek time, latency time, and data transfer time.

Seek time − The time required to arrange the read/write head at the desired track is called seek time. For example, suppose that the read/write head is on track 2 and the record to be read is on track 5, then the read/write head must move from track 2 to track 5. The average seeks time on a modern disk is 8 to 12 ms.

Rotational delay or latency time − The time required to position the read/write head on a specific sector when the head has already been placed on the desired track is called rotational delay. The rotational delay is based on the speed of rotation of the disk.

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

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

Solid-State Disks – New

• Solid-state drive (SSD) is a solid-state storage device that uses integrated circuit assemblies as memory to store data. SSD is also known as a solid-state disk although SSDs do not have physical disks. There are no moving mechanical components in SSD. This makes them different from conventional electromechanical drives such as Hard Disk Drives (HDDs) or floppy disks, which contain movable read/write heads and spinning disks. SSDs are typically more resistant to physical shock, run silently, and have quicker access time, and lower latency compared to electromechanical devices.It is a type of **non-volatile** memory that retains data even when power is lost. SSDs may be constructed from random-access memory (RAM) for applications requiring fast access but not necessarily data persistence after power loss. Batteries can be employed as integrated power sources in such devices to retain data for a certain amount of time after external power is lost.

Magnetic Tapes

Prior to the advent of hard disk drives, magnetic tapes were frequently utilized for secondary storage; today, they are mostly used for backups. It might take a while to get to a specific location on a magnetic tape, but once reading or writing starts, access rates are on par with disk drives.

Disk Structure

Tape drive capacities may be anywhere from 20 and 200 GB, and compression can increase that capacity by double. Modern disk drives use logical block addressing (LBA) to manage data storage. This system treats the disk as a large one-dimensional array of logical blocks, which are the smallest units of data transfer. Typically, a logical block is 512 bytes, but other sizes like 1,024 bytes can also be used. Logical blocks are sequentially mapped onto the disk's physical sectors. Here's the typical mapping process:

1. **Start at Sector 0**: The first sector of the first track on the outermost cylinder.
2. **Proceed Sequentially**: Mapping continues through the sectors of that track, then through other tracks in the cylinder, and finally through other cylinders from the outermost to the innermost.

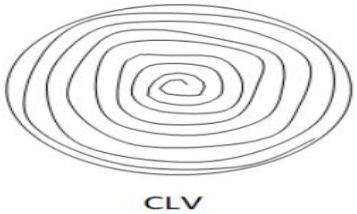
Using this mapping, a logical block number can theoretically be translated into a physical disk address consisting of:

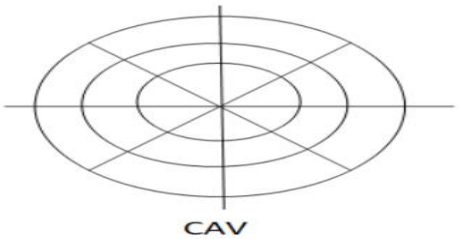
- **Cylinder Number**
- **Track Number within that Cylinder**
- **Sector Number within that Track**

The disk structure (architecture) can be of two types – Constant Linear Velocity (CLV) , Constant Angular Velocity (CAV)

CLV – The density of bits per track is uniform. The farther a track is from the center of the disk, the greater its length, so the more sectors it can hold. As we move from outer zones to inner zones, the number of sectors per track decreases. This architecture is used in CD-ROM and DVD-ROM.

CAV – There is same number of sectors in each track. The sectors are densely packed in the inner tracks. The density of bits decreases from inner tracks to outer tracks to keep the data rate constant.





Disk Attachment

Disk attachment refers to the process of physically connecting a storage device, such as a hard disk drive or solid-state drive, to a computer system. The purpose of disk attachment is to enable the computer system to read and write data to the storage device.

Host-Attached Storage

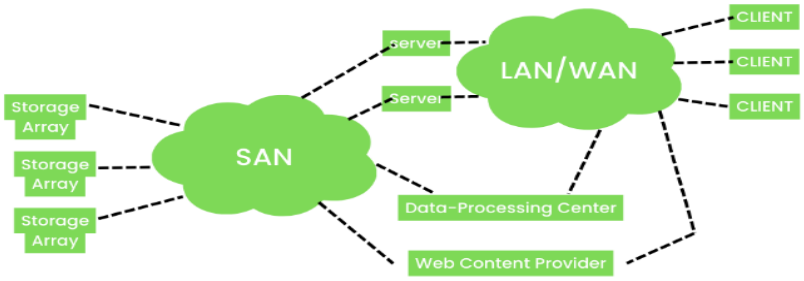
- Host-attached storage (HAS) is a form of internal computer storage that can be attached to a host computer, such as a PC or server. Host-attached devices are often used for backup purposes and can include tape drives, optical drives, **hard disk drives** (HDDs), **solid state drives** (SSDs), **USB flash drives,** and other similar media.
- A common example of host-attached storage is the use of an external USB flash drive for data transfer between computers.
- Host-attached storage is usually very fast, which makes it a popular choice for high-performance applications. It's also relatively cheap compared to network-attached storage (NAS), which is another form of internal computer storage that can be accessed by multiple systems at once.
- Host-attached storage systems are often used in data centers because of their high performance and reliability. These devices are usually more expensive than NAS devices, but they offer many benefits over other types of internal computer storage.

Network-Attached Storage (NAS)

- A Network-Attached Storage (NAS) is a computer that is connected to a network and shared by multiple users. NAS can also be called a file server, or storage appliance. The term "storage" refers to any device that stores data; this includes hard drives and flashes memory modules.
- NAS has its own processor and memory so it can perform many tasks simultaneously without slowing down other devices on the network; this makes it an ideal solution for businesses who need high availability but don't have enough CPU power available at their desktops or laptops.
- NAS systems are often used as backup solutions for PCs because they allow users to access files from anywhere in the world through remote access services provided by vendors such as **Symantec Remote Access Server** (RAS).

Storage Area Network (SAN)

- Storage area networks (SAN) are a network of storage devices that can be used to store and retrieve data from a shared central repository. A SAN is usually used for large data storage and retrieval, which requires high availability, scalability, reliability, and performance. The most common type of SAN uses fiber channel adapters to connect the host with its disk arrays.
- A block-based storage protocol like **Fibre Channel Protocol (FCP)** allows multiple devices within the same fabric to communicate with each other in order to share resources such as disks or tape drives. In contrast to other protocols such as **Network File System (NFS)**, FCP does not require any special software drivers on either end because all communication takes place using standard protocols like TCP/IP.
- Another common type of SAN uses a network called **InfiniBand (IB)**, which is a high-speed serial interconnect that provides better performance than traditional Ethernet networks. When used in conjunction with FCP, IB allows data to be transferred faster and more efficiently between two computers.
- A SAN can be implemented in two ways: as an external or internal device. An external SAN is used to connect storage devices to a network, while an internal SAN is integrated into the operating system of a computer. A disk is basically a device that communicates with the host to fetch data or store data in it with the help of I/O devices.



Unit-V

Topic 2: Disk Scheduling

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.

Various Disk Scheduling Algorithms

FCFS Disk Scheduling Algorithm

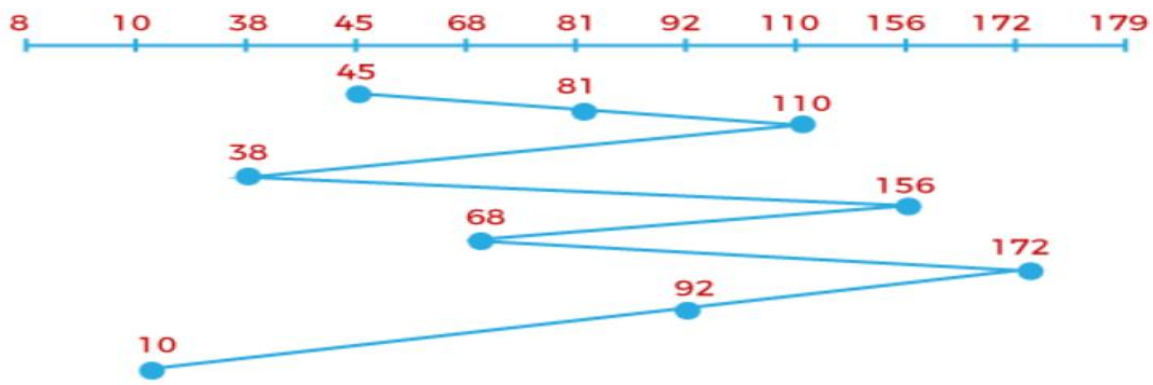
FCFS stands for **First-Come-First-Serve**. It is a very easy algorithm among the all-disk scheduling algorithms. In this scheduling algorithm, the process which requests the processor first receives the processor allocation first. It is managed with a FIFO queue.

Example:

Let's take a disk with **180** tracks **(0-179)** and the disk queue having input/output requests in the following order: **81, 110, 38, 156, 68, 172, 92, 10**. The initial head position of the Read/Write head is **45**. Find the total number of track movements of the Read/Write head using the FCFS algorithm.

The initial head point is **45**,

Solution:



FCFS Disk Scheduling Algorithm

Total head movements= (81-45) + (110-81) + (110-38) + (156-38) + (156-68) + (172-68) + (172-92) + (92-10)

Advantages:

Implementation is easy.

No chance of starvation.

Disadvantages:

'Seek time' increases.

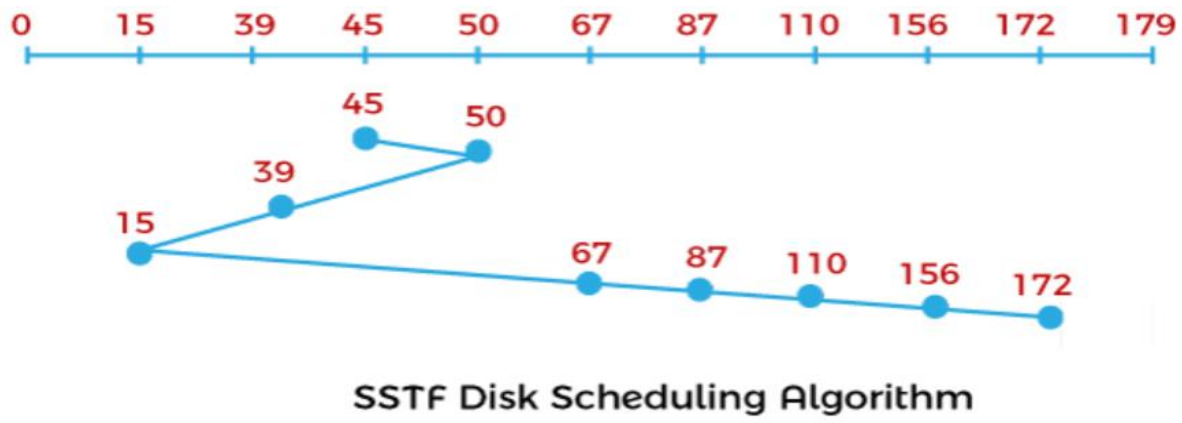
Not so efficient.

SSTF Disk Scheduling Algorithm

SSTF stands for **Shortest Seek Time First**, and it serves the request that is closest to the current position of the head. The direction of the head pointer is quite important in this algorithm. When a tie happens between requests, the head will serve the request in its current direction. In comparison to the FCFS, the SSTF algorithm is very efficient in terms of the total seek time.

Example:

Let's take an example to understand the SSTF Disk Scheduling Algorithm. Let's take a disk with **180** tracks **(0-179)** and the disk queue having input/output requests in the following order: **87, 110, 50, 172, 67, 156, 39, 15**. The initial head position of the Read/Write head is 45 and will move in the left-hand side direction. Find the total number of track movements of the Read/Write head using the SSTF algorithm.



SSTF Disk Scheduling Algorithm

Initial head point is **45**,

Total head movements= (50-45) + (50-39) + (39-15) + (67-15) + (87-67) + (110-87) + (156-110) + (172-156)

Advantages:

1. In this algorithm, disk response time is less.
2. More efficient than FCFS.

Disadvantages:

1. Less speed of algorithm execution.
2. Starvation can be seen.

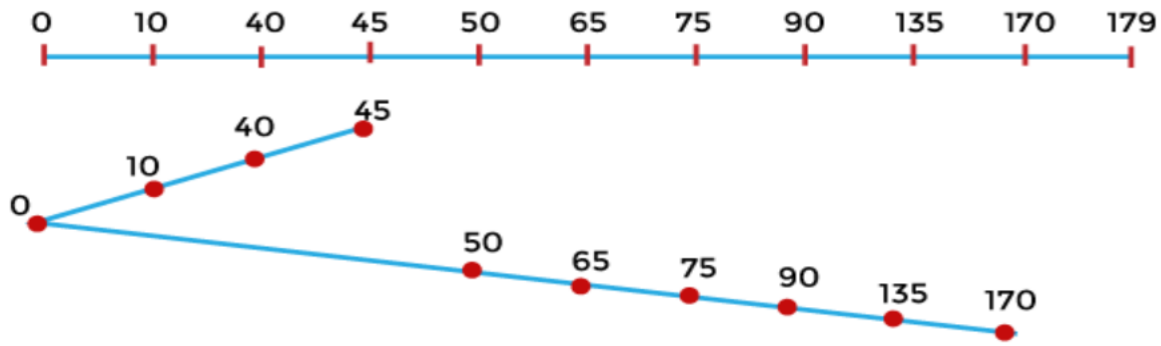
SCAN Disk Scheduling Algorithm

It is also known as the **Elevator algorithm**. In this algorithm, the head may move in both directions, i.e., the disk arm begins to move from one end of the disk to the other end and servicing all requests until it reaches the other end of the disk. After reaching the other end, the head position direction is changed and further continues servicing the requests till the end of the disk.

Example:

Let's take a disk with **180** tracks **(0-179)** and the disk queue having input/output requests in the following order: **75, 90, 40, 135, 50, 170, 65, 10**. The initial head position of the Read/Write head is **45** and will move on the left-hand side. Find the total number of track movements of the Read/Write head using the SCAN algorithm.

Solution:



SCAN Disk Scheduling Algorithm

Total head movements,

Initial head point is 45,

= (45-40) + (40-10) + (10-0) + (50-0) + (65-50) + (75-65) + (90-75) + (135-90) + (170-135)

= 5 + 30 +10 +50 +15 + 10 +15 + 45 + 35= 215

Advantages

1. In the SCAN disk scheduling algorithm, low variance happens in the waiting time and response time.
2. The starvation is avoided in this disk scheduling algorithm.

Disadvantages

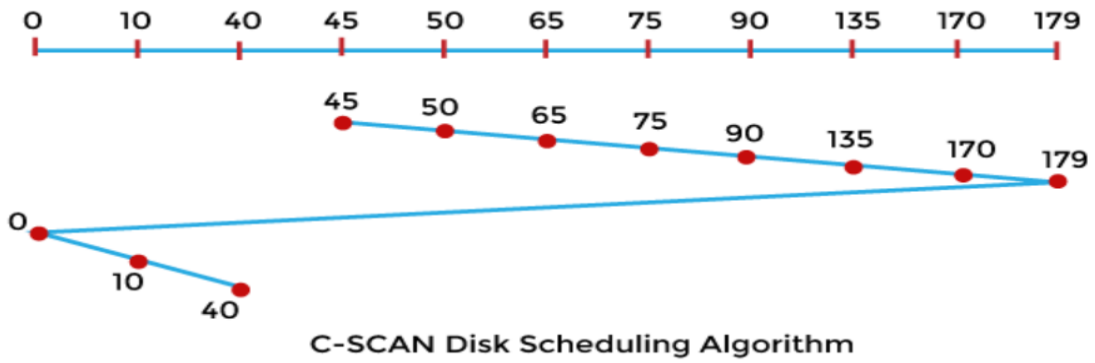
1. If no requests remain to be serviced, the head moves till the end of the disk.

C-SCAN Disk Scheduling

It is also known as the **Circular Elevator algorithm or Circular SCAN**. It is an improved version of the SCAN disk scheduling algorithm. In this algorithm, the head works for requests in a single direction, i.e., it scans all the way to the end of a direction and then jumps to another end and services the requests in the same direction.

Example:

Let's take a disk with **180** tracks **(0-179)** and the disk queue having input/output requests in the following order: **75, 90, 40, 135, 50, 170, 65, 10**. The initial head position of the Read/Write head is **45** and will move on the right-hand side. Find the total number of track movements of the Read/Write head using the C-SCAN algorithm.



C-SCAN Disk Scheduling Algorithm

Total head movements,

The initial head point is 45,

= (50-45) + (65-50) + (75-65) + (90-75) + (135-90) + (170-135) + (179-170) + (179-0) + (10-0) + (40-10)

Advantages

1. It gives a uniform waiting time.
2. It gives a better response time.
3. The head travels from one end to the other disks end and serves all requests along the way.
4. The C-SCAN algorithm is the improved version of the SCAN scheduling algorithm.

Disadvantages

1. If no requests remain to be serviced, the head will travel to the end of the disk.
2. It generates more search movements than the SCAN algorithm.

LOOK Disk Scheduling

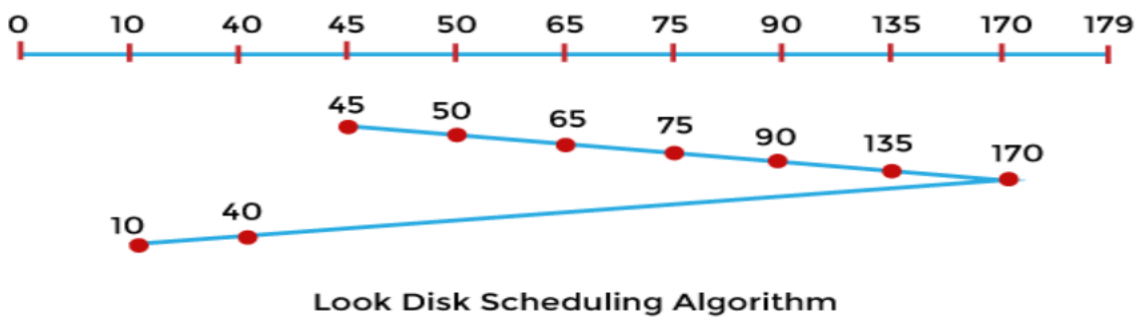
It is the more advanced version of the SCAN disk scheduling algorithm. In this algorithm, the head begins at one end of the disk and works its way to the other end, and serving all requests along the way. When the head reaches the end of one end's last request, it changes direction and returns to the first request, servicing all requests in between. Unlike SCAN, instead of going to the last track, this head goes to the last request and then changes direction.

Example:

Let's take an example to understand the LOOK Disk Scheduling Algorithm. Let's take a disk with **180** tracks **(0-179)** and the disk queue having input/output requests in the following order: **75, 90, 40, 135, 50, 170, 65, 10**. The initial head position of the Read/Write head is 45 and

would move on the right-hand side. Find the total number of track movements of the Read/Write head using the LOOK disk scheduling algorithm.

Solution:



Look Disk Scheduling Algorithm

Total head movements,

The initial head point is 45,

= (50-45) + (65-50) + (75-65) + (90-75) + (135-90) + (170-135) + (170-40) + (40-10)

= 5 + 15 + 10 + 15 + 45 + 35 + 130 + 30= 285

Advantages

1. It provides better performance in comparison to the SCAN algorithm.
2. The LOOK scheduling algorithm avoids starvation.
3. The head will not move to the end of the disk if no more requests are fulfilled.
4. Waiting time and response time have a low variance.

There **Disadvantages**

1. is an overhead of finding the final requests.

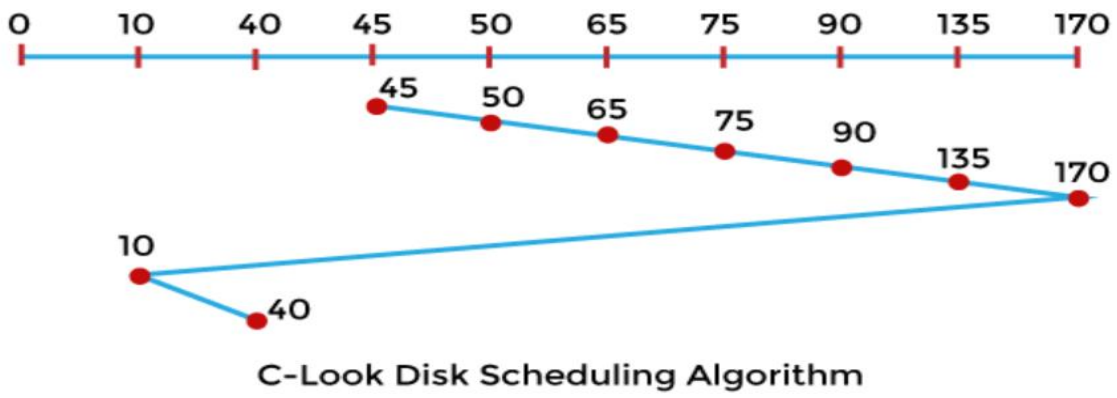
C-LOOK Disk Scheduling Algorithm

It is a combination of the LOOK and SCAN disk scheduling algorithms. In this disk scheduling algorithm, the head begins from the initial request to the last request in the other direction and serves all requests in between. The head jumps in the other direction after finishing the last request at one end and proceeds towards the remaining requests, completing them in the same direction as previously. Unlike LOOK, it only responds to requests in one direction.

Example:

Let's take an example to understand the **C-LOOK** Disk Scheduling Algorithm. Let's take a disk with **180** tracks **(0-179)** and the disk queue having input/output requests in the following order: **75, 90, 40, 135, 50, 170, 65, 10**. The initial head position of the Read/Write head is 45 and

would move on the right-hand side. Find the total number of track movements of the Read/Write head using the C-LOOK disk scheduling algorithm.



C-Look Disk Scheduling Algorithm

Total head movements,

The initial head point is 45,

= (50-45) + (65-50) + (75-65) + (90-75) + (135-90) + (170-135) + (170-10) + (40-10)

Advantages

1. It provides better performance compared to the LOOK disk scheduling algorithm.
2. The starvation is avoided in the C-LOOK disk scheduling algorithm.
3. If no requests are to be served, the head doesn't have to go all the way to the end of the disk in the C-LOOK disk scheduling algorithm.
4. In C-LOOK, there is minimal waiting time for cylinders that are only visited by the head.
5. Waiting time and response time have a low variance.

Disadvantages

1. The overhead of finding the end requests is present in C-LOOK.

Disk Management

The operating system is responsible for several other aspects of disk management.

Disk formatting 2. Booting from disk 3. Bad-block recovery.

Disk Formatting

Disk formatting is of two types.

Physical formatting or low level formatting. b) Logical formatting.

Physical formatting :

- Disk must be formatted before storing a data.

- Disk must be divided into sectors that the disk controller can read/write.
- Low level formatting fills the disk with a special data structure for each sector.
- Data structure consists of three fields: header, data area and trailer.
- Header and trailer contain information used by the disk controller.
- Sector number and Error Correcting Codes (ECC) contained in the header and trailer.
- For writing data to the sector - ECC is updated.
- For reading data from the sector - ECC is recalculated.
- If there is mismatch in stored and calculated value then data area is corrupted.

Logical formatting:

- After disk is partitioned, logical formatting used.
- Operating system stores the initial file system data structures onto the disk.

Boot Blocks

- The process of starting or restarting a computer system or any other computing device is called booting. Blockwhich contains all the data and instructions required for starting the booting process of the system is referred to as boot block. It is also known as boot sector, as it is a sector in the memory device which contains all the instructions required to boot the system.
- Boot block generally resides in the first sector of the data storage device like hard disk and is designed in a standard format so that the BIOS (Basic Input Output System) can understand and execute it.

Components of a Boot Block

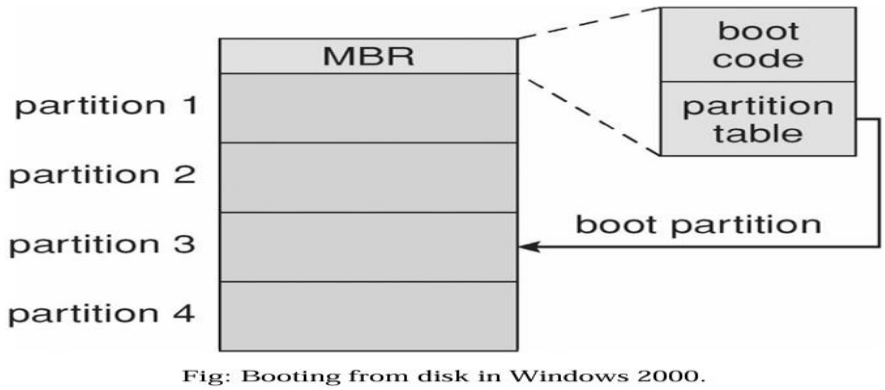


Fig: Booting from disk in Windows 2000.

The following are the important components of the boot block −

- **Master Boot Record (MBR)** − The master boot record (MBR) is a first part of a storage device that contains the boot block, boot code, partition table, and other required data

and instructions. In a computer system, the master boot record is an essential part that helps to understand how the storage device is organized and which partition is required to boot.

- **Bootloader** − Bootloader is a computer program which is responsible for starting the system and loading the operating system into the main memory. It executes all the essential steps required to initiate the booting process.
- **Boot Code** − Boot code, also called bootstrap code, is another important component of boot block. Bootstrap code consists of all the essential instructions written in low-level language like machine language or assembly language. Bootstrap code performs some important functions, including configuration of system components, initialization of hardware parts, loading of operating system into the main memory, etc.
- **Partition Table** − Partition table is another important component of the boot block. It is basically a data table that contains information about the different partition of the data storage device. It helps in identifying the active partition on the disk from which the operating system is to be loaded into the main memory.

Bad Blocks

- A storage region or sector of a data storage device like hard disk drive, flash drive, optical disk, etc. that cannot be used for storage and retrieval of data due to permanent damage is referred to as a bad block. Sometimes, a **bad block** is also known as a **bad sector.**

Depending on the disk and controller in use, these blocks are handled in a variety of ways;

Method 1: "Handled manually: If blocks go bad during normal operation, a special program must be run manually to search for the bad blocks and to lock them . Data that resided on the bad blocks usually are lost.

Method 2: "sector sparing or forwarding" The controller maintains a list of bad blocks on the disk. Then the controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

Method 3: "sector slipping" For an example, suppose that logical block 17 becomes defective, and the first available spare follows sector 202. Then, sector slipping would remap all the sectors from 17 to 202, moving them all down one spot. That is, sector 202 would be copied into the spare, then sector 201 into 202, and then 200 into 201, and so on, until sector 18 is copied into sector 19. Slipping the sectors in this way frees up the space of sector 18, so sector 17 can be mapped to it.

Unit V

Topic 3: Protection

Protection is especially important in a multiuser environment when multiple users use computer resources such as CPU, memory, etc. It is the operating system's responsibility to offer a mechanism that protects each process from other processes.

Goals of Protection

The primary goals of protection in operating systems are to ensure the confidentiality, integrity, and availability of system resources and data. Here are the key goals in detail:

- **Confidentiality**: Ensure that only authorized users and processes can access sensitive information.
- **Integrity**: Ensure that data is not altered or tampered with by unauthorized users or processes.
- **Availability**: Ensure that system resources (CPU, memory, disk space, etc.) are available for authorized users and processes when needed.
- **Controlled Access**: Allow the specification of detailed access controls to system resources.
- **Isolation**: Ensure that processes run in isolated environments to prevent them from interfering with each other.

Principles of Protection

The principles of protection in operating systems are foundational guidelines that help design and implement security mechanisms to safeguard system resources and data. Here are the key principles:

Least Privilege: Each user or process should have the minimum level of access rights necessary to perform their tasks. Assign minimal permissions and escalate privileges only when required, reducing the risk of unauthorized access or damage.

Separation of Duties: Divide responsibilities among multiple users or processes to prevent any single entity from having excessive control. Ensure that critical tasks require collaboration or approval from multiple parties, reducing the risk of insider threats and errors.

Economy of Mechanism: Security mechanisms should be as simple and straightforward as possible.Simplify design and implementation to reduce complexity, making it easier to understand, verify, and manage.

Complete Mediation: Every access to a resource must be checked for proper authorization. Ensure that access control checks are performed every time a resource is accessed, preventing bypassing of security checks.

Separation of Privilege: system should not grant permission based on a single condition.: Require multiple conditions to be met before granting access, such as two-factor authentication or multi-party approval processes.

Least Common Mechanism: Minimize the sharing of mechanisms among users to prevent indirect attacks. Reduce shared resources and interfaces, isolating users and processes as much as possible.

Domian of Protection

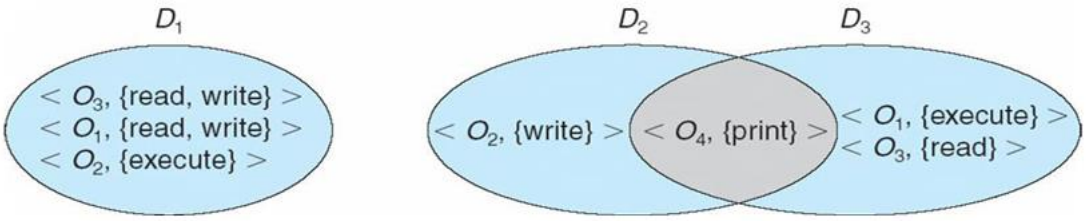
- A computer system is a collection of processes and objects. Objects are both hardware objects (such as the CPU, memory segments, printers, disks, and tape drives) and software objects (such as files, programs, and semaphores). Each object (resource) has a unique name that differentiates it from all other objects in the system.
- The operations that are possible may depend on the object. For example, a CPU can only be executed on. Memory segments can be read and written, whereas a CD-ROM or DVD-ROM can only be read. Tape drives can be read, written, and rewound. Data files can be created, opened, read, written, closed, and deleted; program files can be read, written, executed, and deleted.
- A process should be allowed to access only those resources for which it has authorization and currently requires to complete process . This requirement is known as need to know principle.

Domain Structure

- A domain is a set of objects and types of access to these objects. Each domain is an ordered pair of . <objects set,Rights set>

• Example, if domain D has the access right <File f,{read.write}>, then all process executing in domain D can both read and write file F, and cannot perform any other operation on that object.

• Domains do not need to be disjoint; they may share access rights. For example, in below figure, we have three domains: D1 D2, and D3. The access right < O4, (print}> is shared by D2 and D3,it implies that a process executing in either of these two domains can print object O4.



• A domain can be realized in different ways, it can be a user, process or a procedure. ie. each user as a domain, each process as a domain or each procedure as a domain.

Unit-V

Topic 4: Access Matrix

The Access Matrix is a security model for a computer system's protection state. It is described as a matrix. An access matrix is used to specify the permissions of each process running in the domain for each object

Structure of Matrix:

Rows: Rows define the domain processes that perform some operation on each object. The domain is usually a user or process.

Column: columns are the objects on which domain processes perform operations. Objects can be files, devices, or any system resources.

Cells: cells represent the access right granted to domain processes to operate on objects. Examples are Read, write, delete, execute, etc.

Let us understand the structure of the access matrix with an example of a system command where different user roles have additional access rights to perform some operations on the system command.

Domain: Domain represents the users with different roles such as manager, employee, and staff.

Object: The object represents the system command.

Access rights: access rights are the rights granted to different users for performing some operation on commands that are read, executed, and have no access.

Every matrix cell reflects a set of access rights granted to domain processes, i.e., each entry **(i, j)** describes the set of operations that a domain **Di** process may invoke on object **Oj.**

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

- In the above diagram, there are four domains and four objects—three files (F1, F2, F3) and one printer. A process executing in domain D1 can read files F1 and F3. A process executing in domain D4 has the same privileges as one executing in domain D1; but in addition, it can also write onto files F1 and F3.

- When a user creates a new object Oj, the column Oj is added to the access matrix with the appropriate initialization entries, as dictated by the creator.
- The process executing in one domain and be switched to another domain. When we switch a process from one domain to another, we are executing an operation (switch) on an object (thedomain).

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

- Domain switching from domain D_i to domain D_j is allowed if and only if the access right switch access(i,j). Thus, in the given figure, a process executing in domain D_2 can switch to domain D_3 or to domain D_4. A process in domain D_4 can switch to D_1, and one in domain D_1 can switch to domain D_2.

Allowing controlled change in the contents of the access-matrix entries requires three additional operations: copy, owner, and control.

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

The ability to copy an access right from one domain (or row) of the access matrix to another is denoted by an asterisk (*) appended to the access right. The copy right allows the copying ofthe access right only within the column for which the right is defined. In the below figure, a process executing in domain D_2 can copy the read operation into any entry associated with file F_2. Hence, the access matrix of figure (a) can be modified to the access matrix shown in figure (b).

This scheme has two variants:

1. A right is copied from access(i,j) to access(k,j); it is then removed from access(i,j). This action is a transfer of a right, rather than a copy.
2. Propagation of the copy right- limited copy. Here, when the right R* is copied from access(i,j) to access(k,j), only the right R (not R*) is created. A process executing in domain D_k cannot further copy the right R.

We also need a mechanism to allow addition of new rights and removal of some rights. The owner right controls these operations. If access(i,j) includes the owner right, then a process executing in domain Di, can add and remove any right in any entry in column j.

For example, in below figure (a), domain D1 is the owner of F1, and thus can add and delete any valid right in column F1. Similarly, domain D2 is the owner of F2 and F3 and thus can add and remove any valid right within these two columns. Thus, the access matrix of figure(a) can be modified to the access matrix shown in figure(b) as follows.

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

A mechanism is also needed to change the entries in a row. If access(i,j) includes the control right, then a process executing in domain D_i, can remove any access right from row j. For example, in figure, we include the control right in access(D_4). Then, a process executing in domain D_4 can modify domain D_3.

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Implementation Of Access Matrix

Different methods of implementing the access matrix (which is sparse)

- Global Table
- Access Lists for Objects
- Capability Lists for Domains
- Lock-Key Mechanism

1. Global Table

- This is the simplest implementation of access matrix.
- A set of ordered triples <domain, object, rights-set> is maintained in a file. Whenever anoperation M is executed on an object O_j, within domain D_i, the table is searched for a triple $<D_i, O_j, R_k>$. If this triple is found, the operation is allowed to continue; otherwise,an exception (or error) condition is raised.

Drawbacks -

The table is usually large and thus cannot be kept in main memory. Additional I/O is needed

2. Access Lists for Objects

- Each column in the access matrix can be implemented as an access list for one object. The empty entries are discarded. The resulting list for each object consists of ordered pairs <domain, rights-set>.
- It defines all domains access right for that object. When an operation M is executed on object O_j in D_i, search the access list for object O_j, look for an entry $<D_i, R_K>$ with $M \epsilon R_k$. If the entry is found, we allow the operation; if it is not, we check the default set. IfM is in the default set, we allow the access. Otherwise, access is denied, and anexception condition occurs. For efficiency, we may check the default set first and then search the access list.

3. Capability Lists for Domains

- A capability list for a domain is a list of objects together with the operations allowed on those objects. An object is often represented by its name or address, called a capability.
- To execute operation M on object O_j, the process executes the operation M, specifying the capability for object O_j as a parameter. Simple possession of the capability means that access is allowed.

Capability Format

Object Descriptor	Rights of The Subject Read , Write , Execute

Capabilities are distinguished from other data in one of two ways:

1. Each object has a tag to denote its type either as a capability or as accessible data.
2. The address space associated with a program can be split into two parts. One part is

accessible to the program and contains the program's normal data and instructions. The other part, containing the capability list, is accessible only by the operating system.

4. A Lock-Key Mechanism

- The lock-key scheme is a compromise between access lists and capability lists.
- Each object has a list of unique bit patterns, called locks. Each domain has a list ofunique bit patterns, called keys.
- A process executing in a domain can access an object only if that domain has a key thatmatches one of the locks of the object.