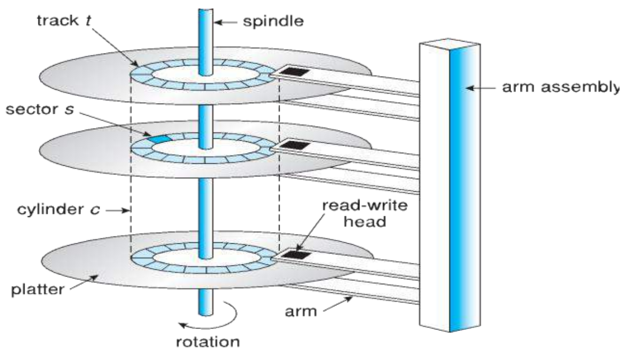
Unit 5 - Name

Mass storage structure – Overview of Mass storage structure – Magnetic Disk

- Overview of Mass Storage Structure
  - Hard disk
  - Magnetic tape
  - Storage Array
  - Storage Area Networks
  - Network Attached Storage
- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface -- That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI, SAS, Firewire**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array



Hard Disks

- Platters range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency

Platters range from .85” to 14” (historically)
Commonly 3.5”, 2.5”, and 1.8”
Range from 30GB to 3TB per drive
Performance
Transfer Rate – theoretical – 6 Gb/sec
Effective Transfer Rate – real – 1Gb/sec
Seek time from 3ms to 12ms – 9ms common for desktop drives
Average seek time measured or calculated based on 1/3 of tracks
Latency based on spindle speed
1 / (RPM / 60) = 60 / RPM
Average latency = ½ latency

Terms to be considered for calculating Hard Disk Performance

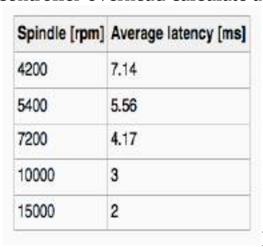
1. **Seek time-** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.

2. **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

3. **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.

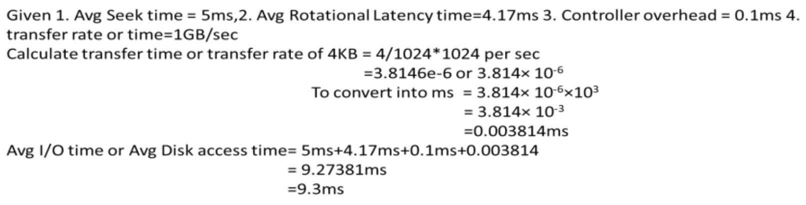
4. **Disk Access Time:** Disk Access = Seek time + Rotational latency + Transfer time

- To transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead calculate average I/O time or Disk access time of 4KB block.



Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

Disk Access time = Seek time + Rotational latency + Transfer time



Given 1. Avg Seek time = 5ms,2. Avg Rotational Latency time=4.17ms 3. Controller overhead = 0.1ms 4. transfer rate or time=1GB/sec

Calculate transfer time or transfer rate of 4KB = 4/1024*1024 per sec

=3.8146e-6 or 3.814×10^{-6}

To convert into ms = $3.814\times 10^{-6}\times10^{3}$

= 3.814×10^{-3}

=0.003814ms

Avg I/O time or Avg Disk access time= 5ms+4.17ms+0.1ms+0.003814

= 9.27381ms

=9.3ms

Solid-State Disks

- Nonvolatile memory used like a hard drive

  Many technology variations

  Can be more reliable than HDD

  More expensive per MB

  Maybe have shorter life span

  Less capacity

  But much faster

  Busses can be too slow -> connect directly to PCI for example No moving parts, so no seek time or rotational latency

Magnetic Tape

- Was early secondary-storage medium
  - Evolved from open spools to cartridges

- Relatively permanent and holds large quantities of data
- Access time slow
- Random access ~1000 times slower than disk
- Mainly used for backup, storage of infrequently-used data, transfer medium between systems
- Kept in spool and wound or rewound past read-write head
- Once data under head, transfer rates comparable to disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000

Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **logical blocks** on physical media
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

Disk Attachment

- Host-attached storage accessed through I/O ports talking to I/O busses

- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
    - Each target can have up to 8 **logical units** (disks attached to device controller)
- FC is high-speed serial architecture
    - Can be switched fabric with 24-bit address space – the basis of **storage area networks (SAN**s**)** in which many hosts attach to many storage units
- I/O directed to bus ID, device ID, logical unit (LUN)

STORAGE MANAGEMENT

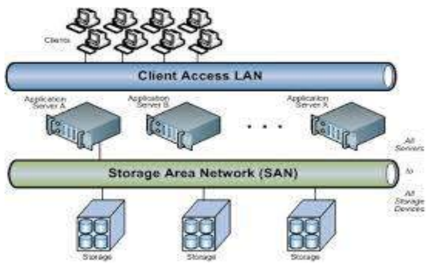
Understanding the Basics in storage management

Storage Array

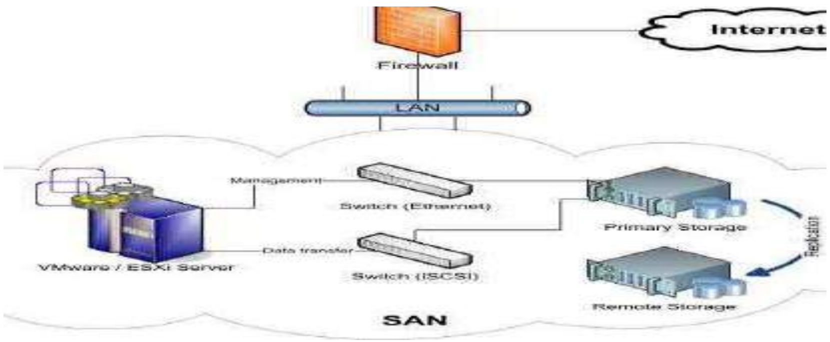
- Can just attach disks, or arrays of disks
- Storage Array has controller(s), provides features to attached host(s)
    - Ports to connect hosts to array
    - Memory, controlling software (sometimes NVRAM, etc)
    - A few to thousands of disks
    - RAID, hot spares, hot swap (discussed later)
    - Shared storage -> more efficiency
    - Features found in some file systems
        - Snaphots, clones, thin provisioning, replication, deduplication, etc

Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple storage arrays - flexible



- SAN is one or more storage arrays
  - Connected to one or more Fibre Channel switches
- Hosts also attach to the switches
- Storage made available via **LUN Masking** from specific arrays to specific servers
- Easy to add or remove storage, add new host and allocate it storage
  - Over low-latency Fibre Channel fabric

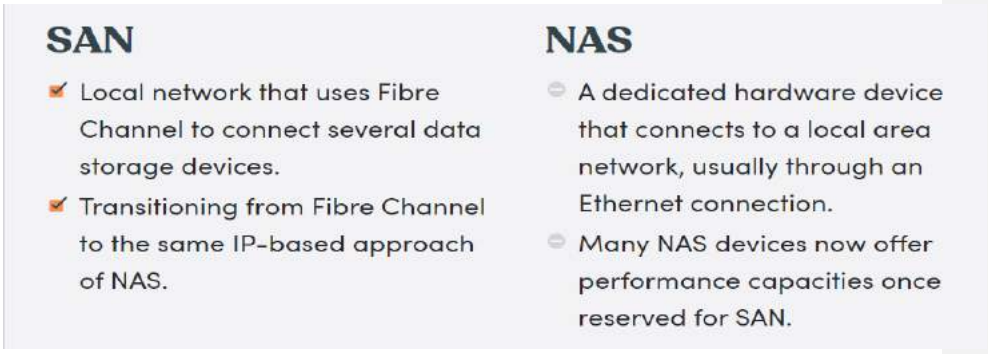


Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
  - Remotely attaching to file systems
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network

- iSCSI protocol uses IP network to carry the SCSI protocol
  - Remotely attaching to devices (blocks)

Difference between SAN Vs NAS



SAN	NAS
✓ Local network that uses Fibre Channel to connect several data storage devices.	A dedicated hardware device that connects to a local area network, usually through an Ethernet connection.
✓ Transitioning from Fibre Channel to the same IP-based approach of NAS.	Many NAS devices now offer performance capacities once reserved for SAN.

Disk Scheduling

Introduction to Disk Scheduling

Why Disk Scheduling is Important

Disk Scheduling Algorithms

- FCFS
- SSTF
- SCAN
- C-SCAN
- LOOK
- C-LOOK

- - Disk Management

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer
- There are many sources of disk I/O request

  - OS
  - System processes
  - Users processes

- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.

- Two or more request may be far from each other so can result in greater disk arm movement.

- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.
- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying "depth")

- Several algorithms exist to schedule the servicing of disk I/O requests

- The analysis is true for one or many platters

Disk Scheduling Algorithms

Disk Scheduling Algorithms

FCFS

SSTF

SCAN

C-SCAN

LOOK

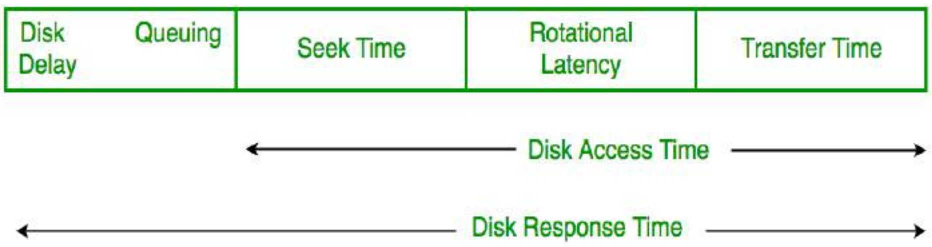
C-LOOK

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **<u>Seek Time</u>**:Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **<u>Rotational Latency:</u>** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **<u>Transfer Time:</u>** 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.
- **<u>Disk Access Time:</u>** Disk Access Time is:

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


        Transfer Time
```

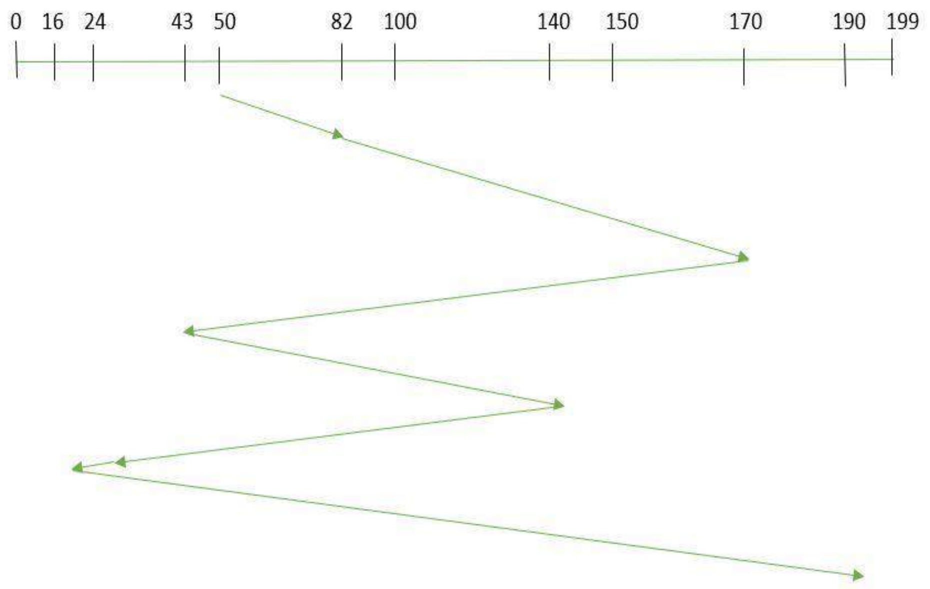


- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

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 seek time:

=(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

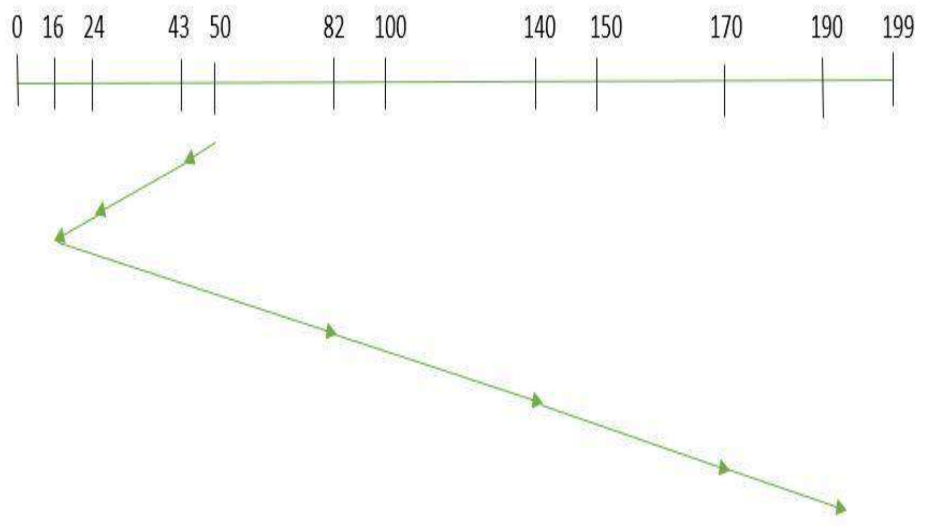
- May not provide the best possible service

2. **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:

Suppose the order of request is- (82,170,43,140,24,16,190)

And current position of Read/Write head is : 50



So, total seek time:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-40)+(190-170)

=208

Advantages:

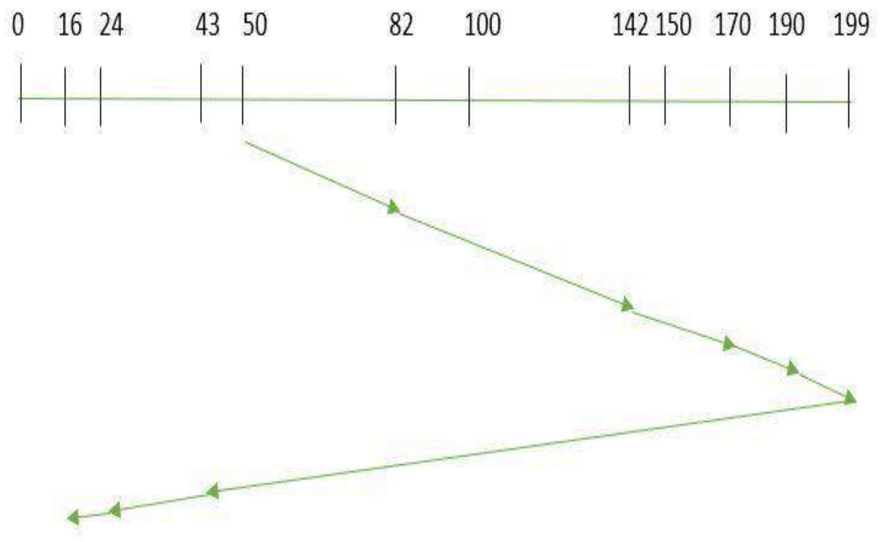
- Average Response Time decreases
- Throughput increases

Disadvantages:

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

3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as **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:
   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 seek time is calculated as:

=(199-50)+(199-16)

=332

Advantages:

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

Disadvantages:

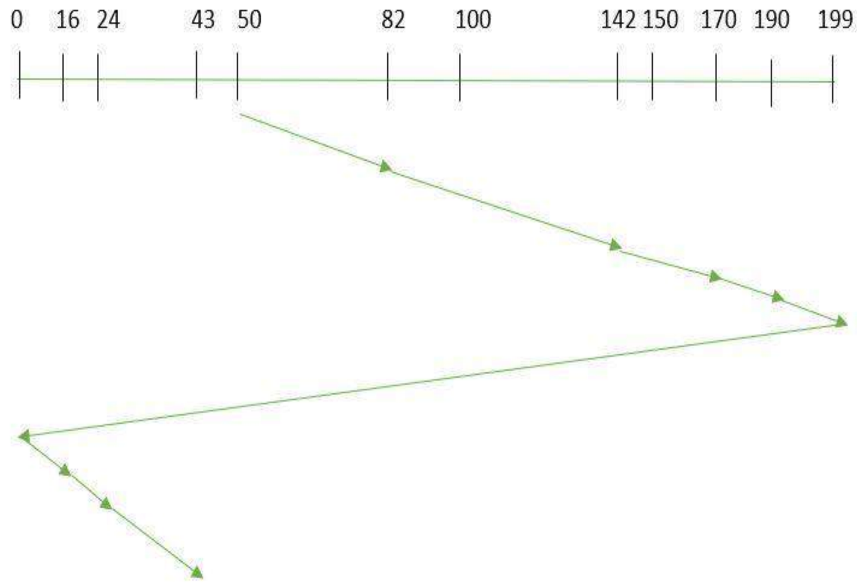
- Long waiting time for requests for locations just visited by disk arm

4. **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**".



Seek time is calculated as:

=(199-50)+(199-0)+(43-0)

=391

Advantages:

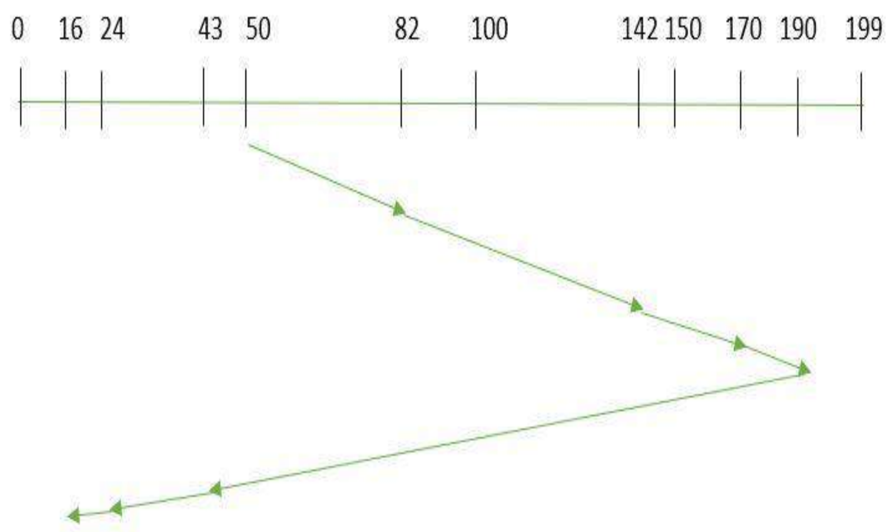
- Provides more uniform wait time compared to SCAN

5. **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:

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 seek time is calculated as:

=(190-50)+(190-16)

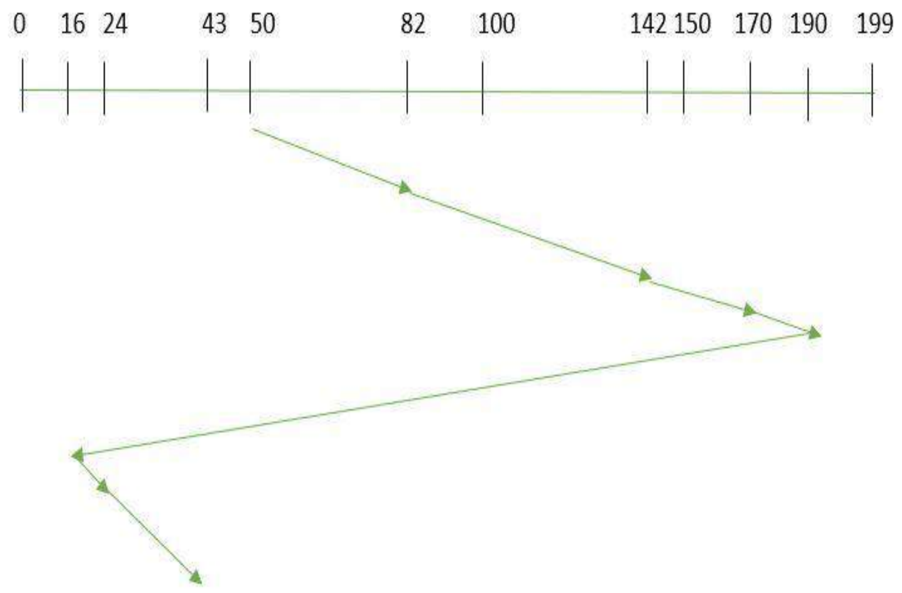
=314

6. **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.

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 seek time is calculated as:

=(190-50)+(190-16)+(43-16)

=341

7. **RSS**– It stands for random scheduling and just like its name it is nature. It is used in situations where scheduling involves random attributes such as random processing time, random due dates, random weights, and stochastic machine breakdowns this algorithm sits perfect. Which is why it is usually

used for and analysis and simulation.

8. **LIFO**– In LIFO (Last In, First Out) algorithm, newest jobs are serviced before the existing ones i.e. in order of requests that get serviced the job that is newest or last entered is serviced first and then the rest in the same order.

   Advantages

   - Maximizes locality and resource utilization

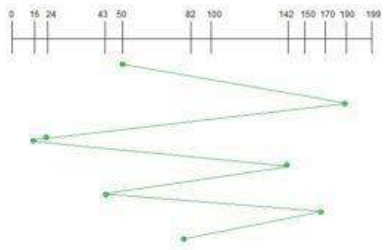
9. **Disadvantages**

   - Can seem a little unfair to other requests and if new requests keep coming in, it cause starvation to the old and existing ones.

10. **Example**

    Suppose the order of request is- (82,170,43,142,24,16,190)

    And current position of Read/Write head is : 50



11. **N-STEP SCAN** – It is also known as N-STEP LOOK algorithm. In this a buffer is created for N requests. All requests belonging to a buffer will be serviced in one go. Also once the buffer is full no new requests are kept in this buffer and are sent to another one. Now, when these N requests are serviced, the time comes for another top N requests and this way all get requests get a guaranteed service

    Advantages

    - It eliminates starvation of requests completely

12. **FSCAN**– This algorithm uses two sub-queues. During the scan all requests in the first queue are serviced and the new incoming requests are added to the second queue. All new requests are kept on halt until the existing requests in the first queue are serviced.

    Advantages

    - FSCAN along with N-Step-SCAN prevents "arm stickiness" (phenomena in I/O scheduling where the scheduling algorithm continues to service requests at or near the current sector and thus prevents any seeking)

Each algorithm is unique in its own way. Overall Performance depends on the number and type of requests.

Note: Average Rotational latency is generally taken as 1/2(Rotational latency).

Exercise

1) Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135 and 145. If Shortest-Seek Time First (SSTF) is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing ____________ number of requests. (GATE CS 2014

(A) 1

(B) 2

(C) 3

(D) 4

See this for solution.

2) Consider an operating system capable of loading and executing a single sequential user process at a time. The disk head scheduling algorithm used is First Come First Served (FCFS). If FCFS is replaced by Shortest Seek Time First (SSTF), claimed by the vendor to give 50% better benchmark results, what is the expected improvement in the I/O performance of user programs? (GATE CS 2004)

(A) 50%

(B) 40%

(C) 25%

(D) 0%

See this for solution.

3) Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given: 45, 20, 90, 10, 50, 60, 80, 25, 70. Assume that the initial position of the R/W head is on track 50. The additional distance that will be traversed by the R/W head when the Shortest Seek Time First (SSTF) algorithm is used compared to the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is _________ tracks

(A) 8

(B) 9

(C) 10

(D) 11

See this for solution.

4) Consider a typical disk that rotates at 15000 rotations per minute (RPM) and has a transfer rate of 50 × 10^6 bytes/sec. If the average seek time of the disk is twice the average rotational delay and the controller's transfer time is 10 times the disk transfer time, the average time (in milliseconds) to read or write a 512 byte sector of the disk is

____

File concept, File access methods

File System Interface

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

- **Files** are the most important mechanism for storing data permanently on mass-storage devices. *Permanently* means that the data is not lost when the machine is switched off. Files can contain:

- data in a format that can be interpreted by programs, but not easily by humans (*binary files*);

- alphanumeric characters, codified in a standard way (e.g., using ASCII or Unicode), and directly readable by a human user (*text files*). Text files are normally organized in a sequence of lines, each containing a sequence of characters and ending with a special character (usually the *newline character*). Consider, for example, a Java program stored in a file on the hard-disk. In this unit we will deal only with text files.

- Each file is characterized by a *name* and a *directory* in which the file is placed (one may consider the whole path that allows one to find the file on the hard-disk as part of the name of the file).

File Access Methods

1. Sequential Access Method: – Sequential Access Method is one of the simplest file access methods. Most of the OS (operating system) uses a sequential access method to access the file. In this method, word by word, the operating system read the file, and we have a pointer that points the file's base address. If we need to read the file's first word, then there is a pointer that offers the word which we want to read and increment the value of the word by 1, and till the end of the file, this process will continue.

- The Modern world system offers the facility of index access and direct access to file. But the sequential access method is one of the most used methods because more files like text files, audio files, and the video files require to be sequentially accessed.

1. Sequential Access Method is reasonable for tape.

Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 21 pt
Formatted: Font: 21 pt
Formatted: Font: 21 pt
Formatted: Font: 21 pt
Formatted: Font: 21 pt

2. In the sequential access method, if we use read command, then the pointer is moved ahead by 1.

3. If the write command is used, then the memory will be allocated, and at the end of the file, a pointer will be moved.

Direct Access Method: – Direct Access Method is also called the Relative access method. In the Database System, we mostly use the Direct Access Method. In most of the situations, there is a need for information in the filtered form from the database. And in that case, the speed of sequential access may be low and not efficient.

- Let us assume that each storage block stores 4 records, and we already know that the block which we require is stored in the 10th block. In this situation, the sequential access method is not suitable. Since if we use this, then to access the record which is needed, this method will traverse all the blocks.

- So, in this situation, the Direct access method gives a more satisfying result, else the OS (operating system) needs to perform a few complex jobs like defining the block number, which is required. It is mostly implemented in the application of the database.

Index Access Method: – Index Access Method is another essential method of file, accessing. In this method, for a file an index is created, and the index is just like an index which is at the back of the book. The index includes the pointer to the different blocks. If we want to find a record in the file, then first, we search in the index, and after that, with the help of the pointer, we can directly access the file.

- In the Index Access method, we can search fast in the large database, and also easy. But in this, the method need some additional space to store the value of the index in the memory.

File Attributes

- Name: It is the only information stored in a human-readable form.
- Identifier: Every file is identified by a unique tag number within a file system known as an identifier.
- Location: Points to file location on device.
- Type: This attribute is required for systems that support various types of files.
- Size. Attribute used to display the current file size.
- Protection. This attribute assigns and controls the access rights of reading, writing, and executing

Formatted: Font: 21 pt

Formatted: Font: 21 pt

Formatted: Indent: Left: 0.35 cm, Hanging: 0.49 cm, Space Before: 18 pt, Line spacing: Multiple 0.75 li, Outline numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Right + Aligned at: 0.3 cm + Indent at: 0.76 cm

Formatted: Font: (Default) Century Gothic, 1 pt, Font color: Black

Formatted: Font: 7 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

Formatted: Font: 11 pt

the file.

- Time, date and security: It is used for protection, security, and also used for monitoring

File Operations

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

2.Write

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

3.Read

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

4.Re-position

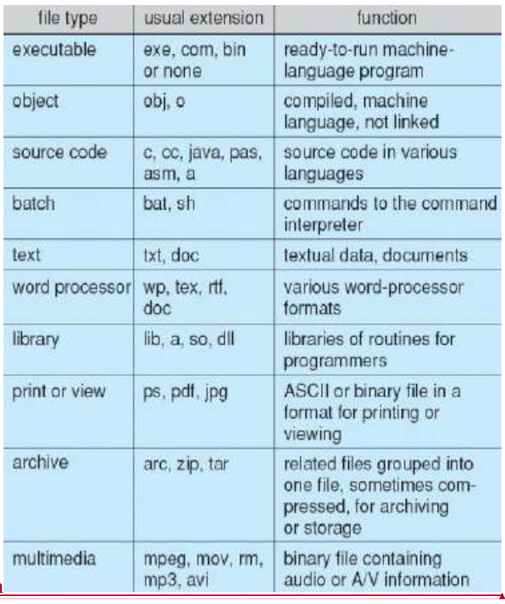
- Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as seeking.

5.Delete

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

6.Truncate

Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt
Formatted: Font: 11 pt



file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Types – Names,Extension

File Structure

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

PROCESS SYNCHRONIZATION

Semaphore

Semaphore is used to implement process synchronization. This is to protect critical regions shared among multiple processes. Include the following header files for shared memory , . System V Semaphore System Calls

To create a semaphore array,

int semget(key_t key, int nsems, int semflg)

key -> semaphore id nsems -> no. of semaphores in the semaphore array semflg -> IPC_CREATE|0664 : to create a new semaphore

IPC_EXCL|IPC_CREAT|0664 : to create new semaphore and the call fails if the semaphore already exists

To perform operations on the semaphore sets viz., allocating resources, waiting for the resources or freeing the resources,

int semop(int semid, struct sembuf *semops, size_t nsemops) semidsa-> semaphore id returned by semget() nsemops -> the number of operations in that array semops -> The pointer to an array of operations to be performed on the semaphore set.

The structure is as follows struct sembuf { unsigned short sem_num; /* Semaphore set num */ short sem_op; /* Semaphore operation */ short sem_flg; /*Operation flags, IPC_NOWAIT, SEM_UNDO */ };

Element, sem_op, in the above structure, indicates the operation that needs to be performed −

● If sem_op is –ve, allocate or obtain resources. Blocks the calling process until enough resources have been freed by other processes, so that this process can allocate.

● If sem_op is zero, the calling process waits or sleeps until semaphore value reaches 0.

● If sem_op is +ve, release resources. To perform control operation on semaphore,

int semctl(int semid, int semnum, int cmd,…);

semid -> identifier of the semaphore returned by semget() semnum -> semaphore number cmd -> the command to perform on the semaphore.

Ex. GETVAL, SETVAL semun -> value depends on the cmd. For few cases, this is not applicable.

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))

Formatted: Font: (Default) Arial, 11 pt, Font color: Custom Color(RGB(4,97,123))