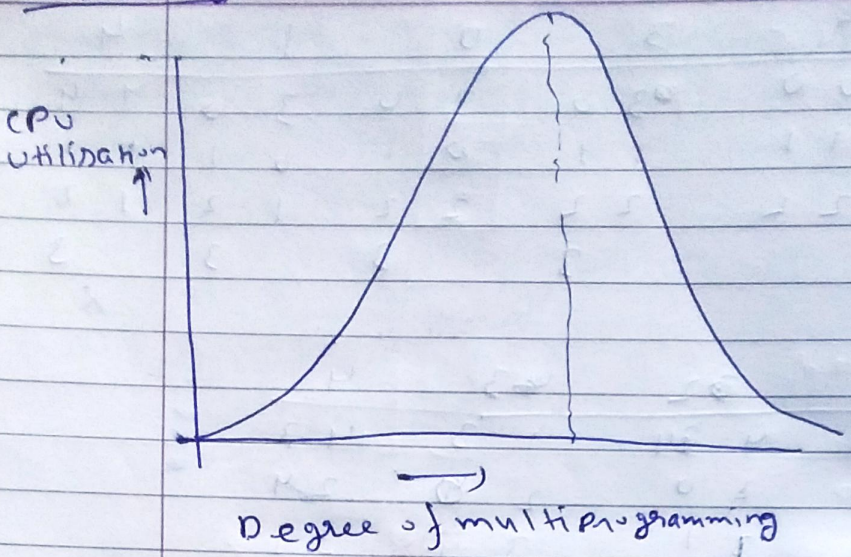
Thrashing



Degree of multiprogramming increases means that no. of processes in mm increases. If more processes are present in mm, CPU utilization will increase. Since, mm size is limited, so, to increase the no. of process in mm, we will have to keep less no. of pages per process in mm. So, as degree of multiprogram increases, no. of pages per process present in mm decreases. So, in this case page fault will increase, therefore, CPU utilisation will decrease. In this case, Scheduler will think that CPU utilisation has decreased bcz of lower degree of multiprogramming. As a result, it will increase to increase the degree of multiprogramming, more processes will be pushed to mm, hence, more page faults. This and CPU utilisation will become very low. This situation is called thrashing.

Disk Scheduling

Page No.
Date

Seek time :- time required to move the R/W head on the desired track

Rotational latency :- the time required by the R/W head to move to the desired section in the after reaching the desired track.

Access time = Seek time + rotational latency + transfer time

Transfer time = $\frac{\text{Data to be transfer}}{\text{transfer rate}}$

transfer rate or Data rate
= no. of heads in no. of surfaces * Capacity of one track + no. of rotations in one second.

Rotation time = time taken for one full rotation.
Average rotational latency is considered as half of the rotation time.
Actual rotational latency in best case can be 0 and in worst case can be the time to take one full rotn, i.e. rotation time.

Controller overhead :- time taken by the controller to manage the transfer of data. However, it is not an important parameter.
Queue time :- waiting time of the request in the buffer due to speed mismatch of CPU and hard disk. not an important parameter

Disk access time = Seek time + rotational latency + transfer time.

Page N
Date

Disk Scheduling algos.

FCFS (First come First Serve)

Consider the following seqence of tracks in which reqs are generated.

98, 183, 41, 122, 14, 124, 65, 67

Currently R/W head is at 53.

98 means that

Here, 98 means that a request is arrived to access data which is stored in some particular section of the track 98. So, R/W head will go to 98th track. After reaching at track 98, the required section of the track 98 will reach below the R/W head through rotation of the platter. Similarly will be done for other request.

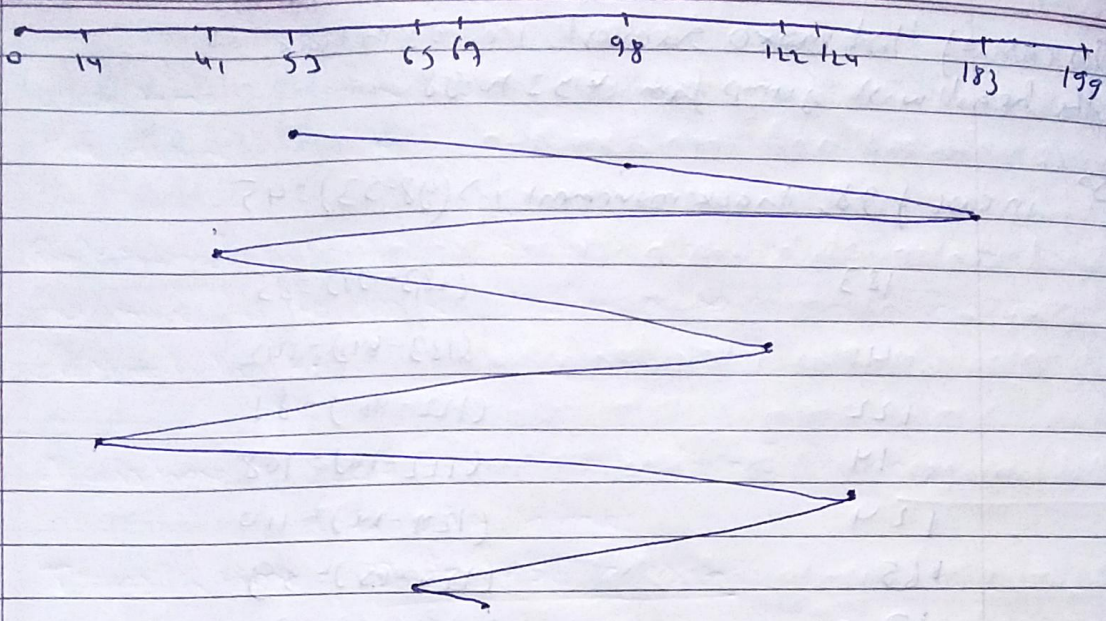
Here, Seek time is involved in reaching the R/W head to the desired track. So, if the disk controller has received multiple disk access request then in order to minimise the av. seek time, an algorithm is followed on reaching to these track. These algos are called disk scheduling algo.

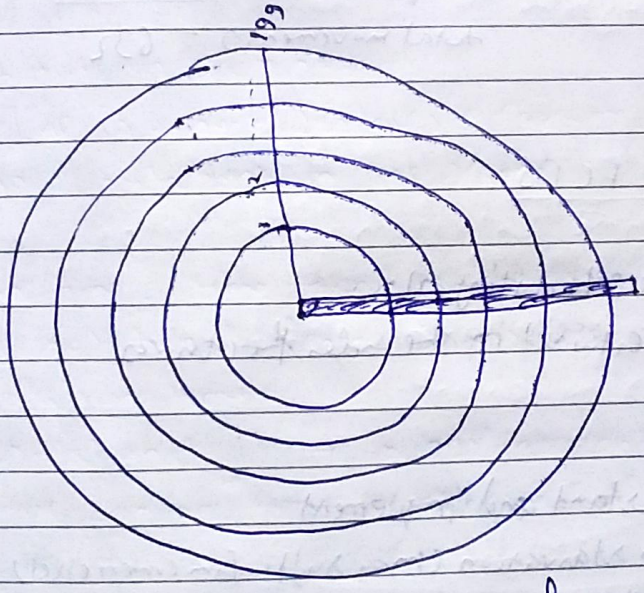
The 1st disk algo we will discuss is First come First Serve (FCFS) algo.

In this algo, the R/W head will jump to the tracks in the same order in which the requests are arrived. In case of above sequence, in FCFS algo, the R/W head will jump to 98th track then 183, then 41th track and so on.

98, 183, 41, 122, 14, 124, 65, 67

Page No.
Date





In the given sequence of requests we have to calculate total no. of track movements done by the R/W head. In place of track movements, we can cylinder movements is also considered. Both are technically same. By calculating the track movements or cylinder movements, total seek time can be calculated if seek time per cylinder or track is given.

98 183 41 122 14 124 65 67

Page No.
Date

In case of 1st request, i.e. 98th track, the head will jump from 53 to 98

So, in case of 98, track movement is (98-53) = 45

183	(183-98) = 85
41	(183-41) = 142
122	(122-41) = 81
14	(122-14) = 108
124	(124-14) = 110
65	(124-65) = 59
67	(67-65) = 2
	total movements = 632

Advantages of FCFS

→ Simplest disk scheduling algo.
→ Request are entertained in the order they arrives

Advantage

→ easy to understand and implement
→ no starvation (may suffer from convoy effect)
→ can be used with less load

Disadvantage

→ Require more seek time
→ more waiting time and response time
→ Inefficient.

98 183 41 122 14 124 65 67

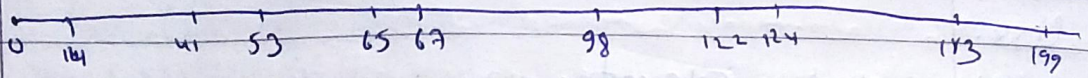
Page No.
Date

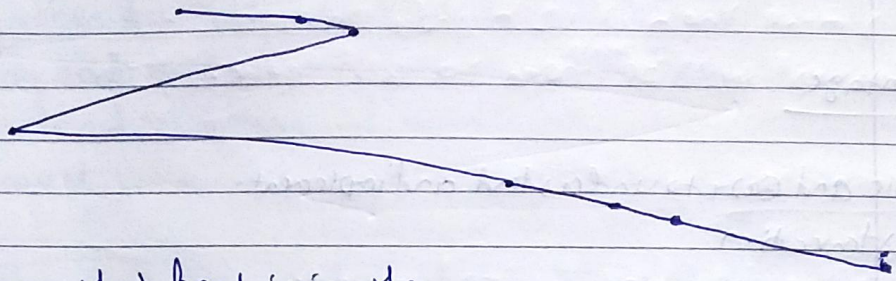
Shortest Seek time First (SSTF)

Serve the request which is closest to the current position of the R/w head.

→ tie is broken in the direction of head movement.

~~A tie~~ is i.e. if head in at 50 and reqests are for 40 and 60.





Assume that head is in the direction of 199.

total movements = (65-53) + (67-65) + (67-41) + (41-14)
+ (98-14) + (122-98) + (124-122) + (183-124)
= 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59
= 236

Advantage

→ very efficient in seek moves
→ less av. response and waiting time
→ increase throughput.
→ good for low load but in high load ~~some reaquests may starve~~ farther reaquests may starve

Disadvantage

→ overhead to find out closest request
→ Reaquest which are far from head will starve
→ high variance in response and waiting time i.e. some reaquests will have less waiting time whereas some reaquest will have high waiting time.

98 183 41 122 14 124 65 67
head - 53 ↑

SCAN

Page No.
Date

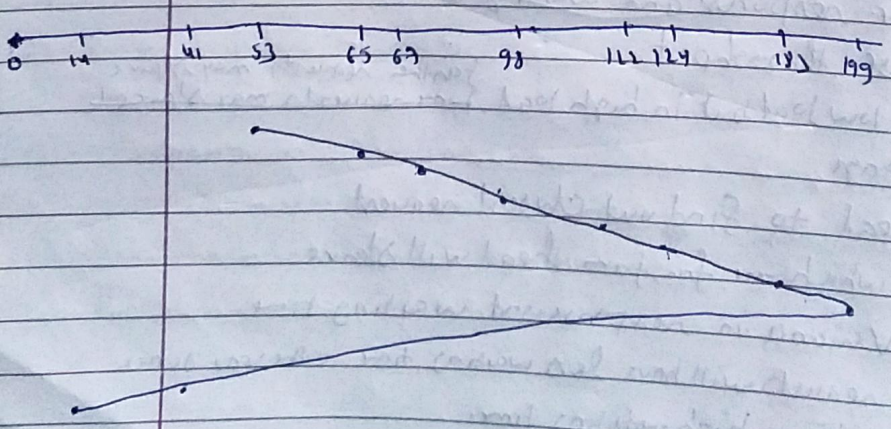
Head start at one end of the disk and moves towards the other end, servicing requests in between and reach other end and then direction of the head is reversed and process continues.
Head continously scans back and forth across the disk.
→ Also called as Elevator algo.

Advantage

→ Simple and easy to understand and implement.
→ no starvation
→ low av. waiting time

Disadvantage

→ long waiting time for locations just visited by head.
→ unnecessary move till the end of the disk even if there is no request.



98 183 41 122 14 124 65 67

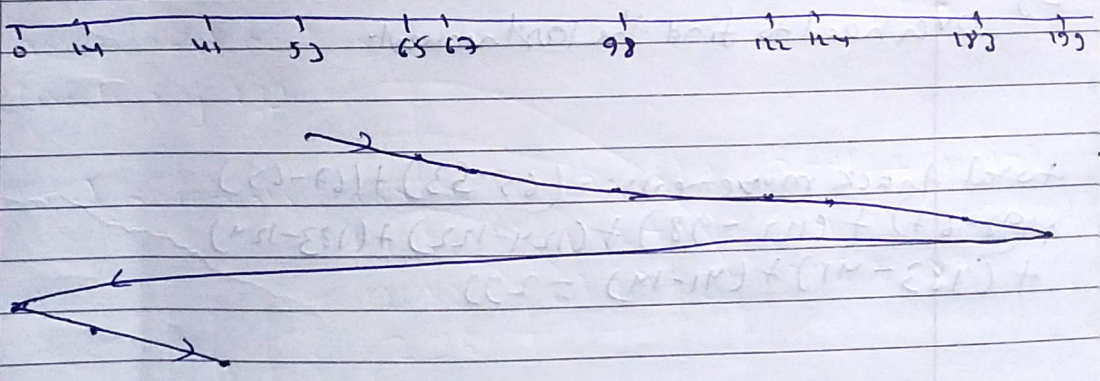
RW head 53 ↑

Page No.
Date

total movements :- (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + (199-41) + (41-14) = 331

C-Scan (Circular-Scan)

Head starts at one end of the disk and moves towards the other end, servicing reauests in between and reach other end and then direc^{n} of the head is reversed and head reaches 1st end without Satisfying any request.



Advantage :- → Provides uniform waiting time
→ better response time

Disadvantage :- → more ~~seek~~ track movements in compare to Simple Scan

total movement :- (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + (199-0) + (14-0) + (41-14) = 386

Page No.
Date

Look Algorithm

It is same as scan algo but instead of going till last track we go till last request and then change direc".

Advantage :-

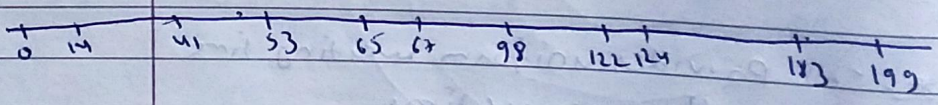
→ Better performance compared to SCAN

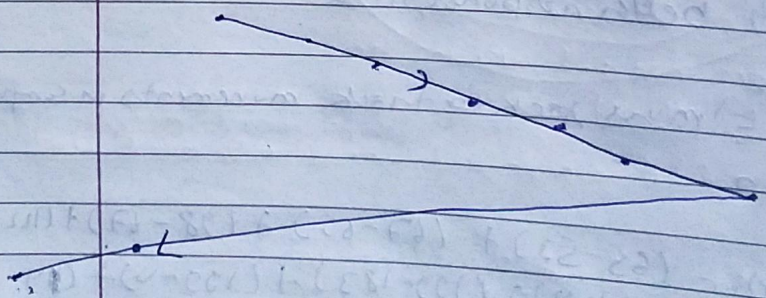
→ Should be used in case of less load bcoz if load is more then it is highly probable that last request will be very near to last track.

Disadvantage

→ overhead to find the last request

total track movements :- (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-41) + (41-14) = 299





98 183 41 122 14 124 65 67

Page No.
Date

C-Look (Circular Look)

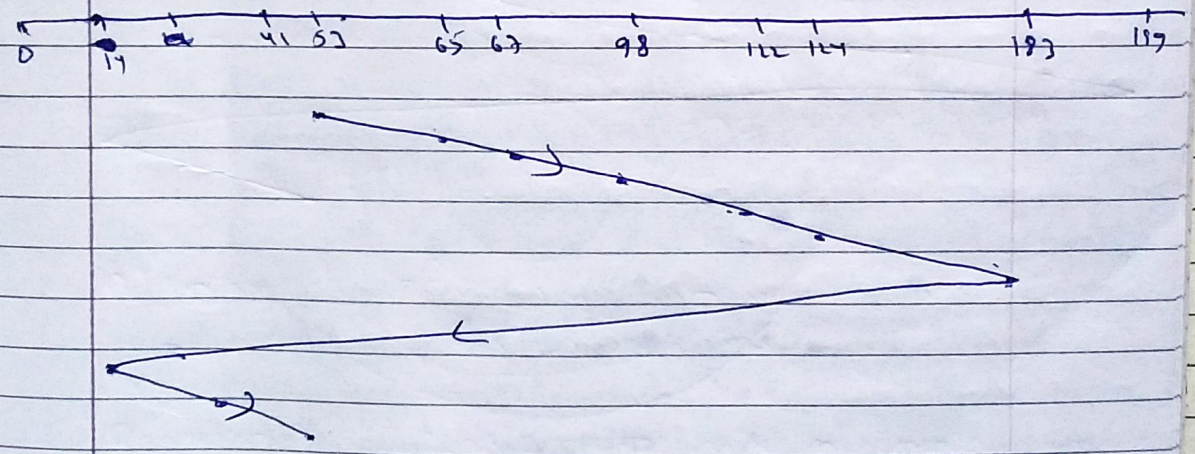
- It takes advantage of both C-SCAN and Look algo.
- → will satisfy request only in one direction
- → will go till last request and return but not till last track.

Advantage

- → more uniform waiting time compared to Look.
- → more efficient (less moves) compared to C-SCAN.

Disadvantage

- → more overhead in calculations
- → should not be used in case of more load (as C-SCAN)



total movements :- (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (183-14) + (41-14) = 326

FCFS - 632
SSTF - 236
SCAN - 331
C-SCAN - 386
LOOK - 299
C-LOOK - 326

Page No.
Date

=> Consider a disk with following specifications :- 16 surfaces, 128 tracks per surface, 256 sectors per track and 512 bytes per sector.

(a) what is the capacity of the disk?

Ans:- $16 \times 128 \times 256 \times 512$ bytes $= 2^{28}$ bytes $= 256$ MB

(b) If the format overhead is 32 bytes per sector then what is the formatted disk space?

Ans:-

formatting overhead = total no. of sectors * overhead per sector

$= 16 \times 128 \times 256 \times 32$

$= 2^{24}$ bytes $= 16$ MB

∴ formatted disk space = total disk − formatted overhead

$= 256 - 16 = 240$ MB

(c) If disk rotational speed is 3000 RPM (revolution per minute) then find the ~~cpu~~ access time if seek time = 11.5 ms

Ans:- Here, access time means that time to access one sector.

rotational latency = $\frac{\text{time in one rotation}}{2} = \frac{60 \times 1000}{3000 \times 2} = 10$ ms

transfer rate or datarate = $16 \times 256 \times 512 * \frac{3000}{60}$

$= 2^4 \times 2^8 \times 2^9 \times 50 = 50 \times 2^{21}$

transfer time = $\frac{512}{50 \times 2^{21}} \times 1000$ ms $= \frac{20}{4096}$ ms

∴ access time for 1 sector = $11.5 + 10 + \frac{20}{4096}$ ms

Page No.
Date

File System

- File system logically divides a file into several block and maps these blocks to sectors in the hard disk so that the file can be stored in the hard disk. The file system keeps the information that which block of a file is kept in which section so that it can be easily fetched when user wants to access the file.

- Operation on files :-
  (a) Creating (b) reading (c) writing (d) deleting
  (e) ~~truncating~~ (e) truncating (f) repositioning

In deleting a file, both content of file and its attributes (meta data) are deleted but in truncation, only content is deleted but attributes of file are not deleted

File attributes :-
(1) name (2) extension - .c, .cpp, .mp4
(3) Identifier - given by OS to identify the file. like roll no.
(4) location (5) size
(6) date and time :- modified date, created date
(7) protection / permission
(8) encryption, compression

Reposition is actually repositioning of the file pointer at any position of the file.

Allocation methods

As we know that file system divids the data of a file in multiple logical block. The method used to allocate these logical blocks to the sectors (physical blocks) of the hard disk is called allocation methods.

Allocation method are of two type:

(i) Contiguous allocation

(ii) non-contiguous allocation.

Linked list allocation — Indexed allocation

Aim of the allocation methods are

(i) efficient disk utilitation

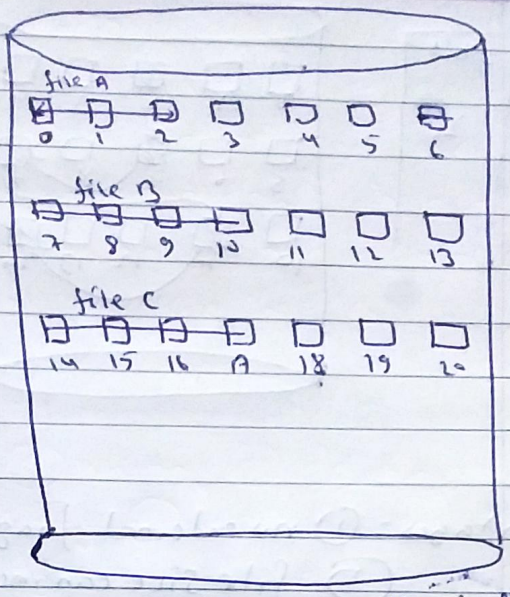
(ii) accessing the data should be faster.

Contiguous allocation of a file:- In this methods, the logical blocks are allocated contiguous sectors in the hard disk.

Let there are three files A, B, C which of size 3, 5, and 4 blocks.

Since, in contiguous allocation, all blocks of a file are given contiguous sectors or physical block, so, in this case, we just need to know the beginning sector no. of a file.

file	Start	Length
A	0	3
B	6	5
C	14	4



contiguous allocation in disk

Here squares are sectors in hard disk.

Advantages of contiguous allocation:

(i) easy to implement:- but no need of maintaining pointers or index.

(ii) excellent read performance:- Since data is stored in contiguous sectors, so, moving to different tracks will be minimized in case of accessing data of a file.

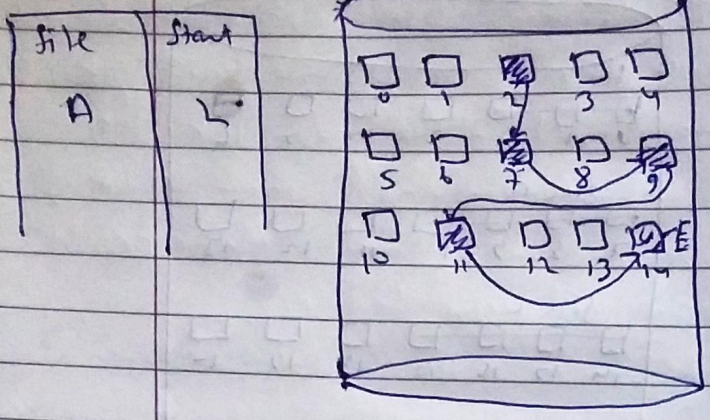
Disadvantages

(i) Disk will become fragmented, so disk space will not be efficiently used.

(ii) difficult to grow file.

On the above allocation, if file A is edited and it needs 5 more blocks then it is not possible to allocate 5 more blocks there but from block 6, file B's allocation starts. On this case, file A will be forcibly allocated.

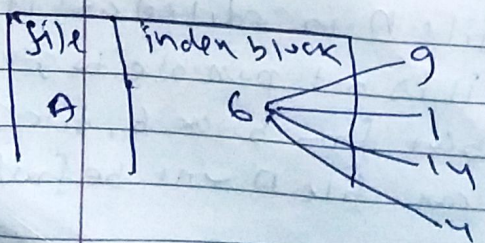
Linked list Allocation



Advantages: (i) no external fragmentation
(ii) file size can grow

Disadvantage:-
(i) large seek time
(ii) random access / direct access difficult
(iii) overhead of pointers.

Indexed Allocation
In this allocation, each file maintains an index in which allocation of its block is stored with block numbers are given. This index is stored in a different block. For example, let file A has 4 blocks which are stored in sector no. 9, 1, 14, and 4. Its index is stored in block 6.



Page No.
Date

Advantages:

(i) Support to direct access
(ii) no external fragmentation

Disadvantages:

(i) pointer overhead (used in indexing)
(ii) multilevel index (in case of large file)