**Module-6**

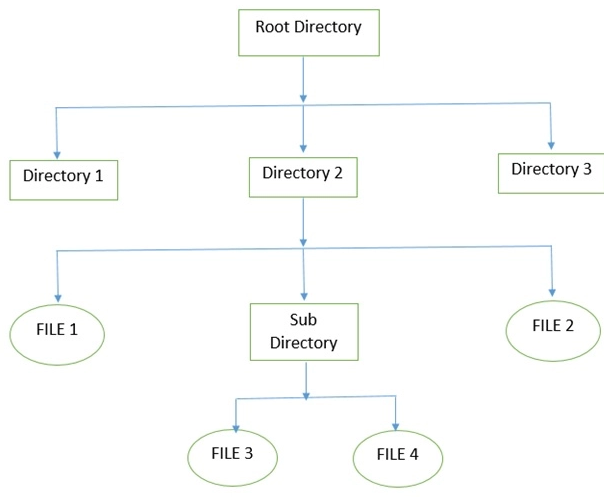
**File Management in Operating System**

**File management is one of the basic and important features of operating system. Operating system is used to manage files of computer system**. All the files with different extensions are managed by operating system.

A file is collection of specific information stored in the memory of computer system. **File management is defined as the process of manipulating files in computer system, it management includes the process of creating, modifying and deleting the files**.

The following are some of the **tasks performed by file management of operating system of any computer system**:

1. It helps to create new files in computer system and placing them at the specific locations.
2. It helps in easily and quickly locating these files in computer system.
3. It makes the process of sharing of the files among different users very easy and user friendly.
4. It helps to stores the files in separate folders known as directories. These directories help users to search file quickly or to manage the files according to their types or uses.
5. It helps the user to modify the data of files or to modify the name of the file in the directories.



The above figure shows the **general hierarchy of the storage in an operating system**. In this figure the root directory is present at the highest level in the hierarchical structure. It includes all the subdirectories in which the files are stored. Subdirectory is a directory present inside another directory in the file storage system. The directory base storage system ensures better organization of files in the memory of the computer system.

**The file management of function in operating system (OS) is based on the following concepts:**

1. **File Attributes**  
   It specifies the characteristics of the files such as type, date of last modification, size, location on disk etc. file attributes help the user to understand the value and location of files. File attributes is one most important feature. It is uses to describe all the information regarding particular file.
2. **File Operations**  
   It specifies the task that can be performed on a file such as opening and closing of file.
3. **File Access permission**  
   It specifies the access permissions related to a file such as read and write.
4. **File Systems**  
   It specifies the logical method of file storage in a computer system. Some of the commonly used files systems include FAT and NTFS.

**FILE DIRECTORIES:**  
Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.  
  
**Information contained in a device directory are:**

* Name
* Type
* Address
* Current length
* Maximum length
* Date last accessed
* Date last updated
* Owner id
* Protection information

**Operation performed on directory are:**

* Search for a file
* Create a file
* Delete a file
* List a directory
* Rename a file
* Traverse the file system

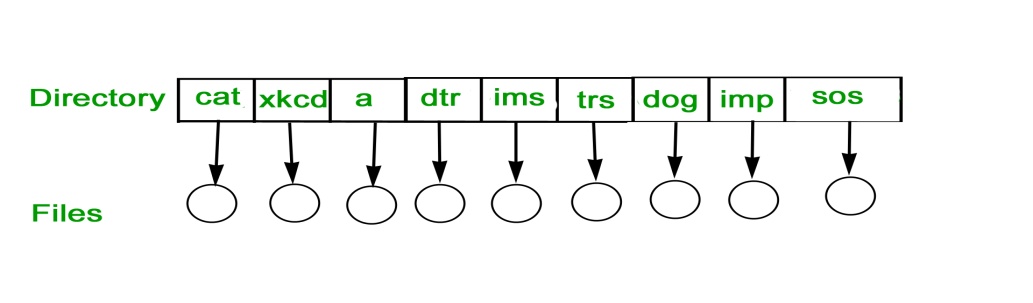
**Advantages of maintaining directories are:**

* **Efficiency:** A file can be located more quickly.
* **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
* **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

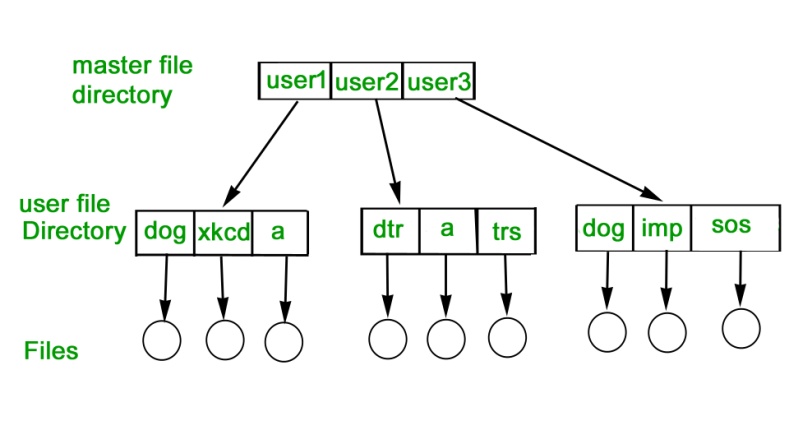
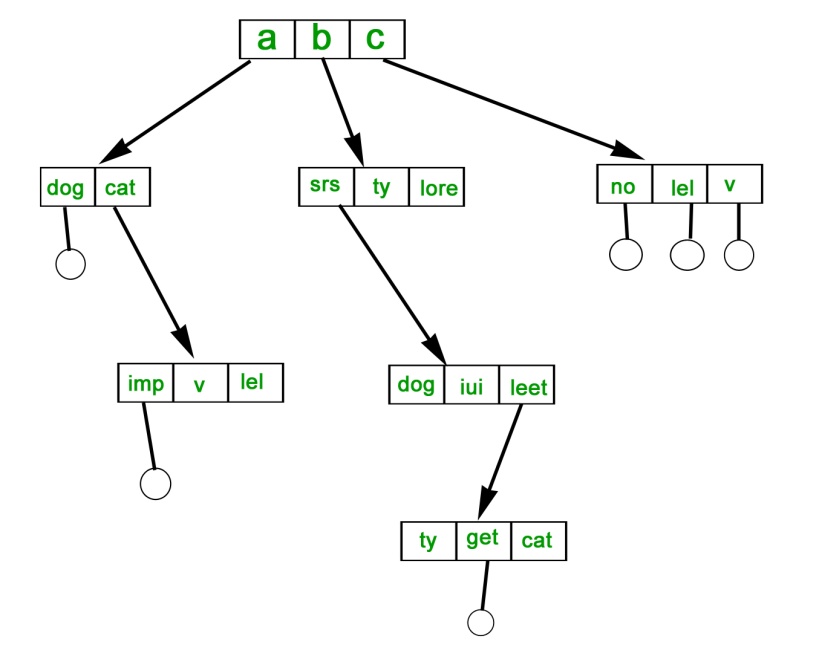
**TYPES OF DIRECTORIES:**

**SINGLE-LEVEL DIRECTORY**  
In this a single directory is maintained for all the users.

* **Naming problem:** Users cannot have same name for two files.
* **Grouping problem:** Users cannot group files according to their need.

  
  
**TWO-LEVEL DIRECTORY**  
In this separate directories for each user is maintained.

* Path name:Due to two levels there is a path name for every file to locate that file.
* Now,we can have same file name for different user.
* Searching is efficient in this method.

  
  
**TREE-STRUCTURED DIRECTORY :**  
Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.

**Structures of Directory in Operating System**

A **directory** is a container that is used to contain folders and file. It organizes files and folders into a hierarchical manner.

There are several logical structures of a directory, these are given below.

111.jpg

1. **Single-level directory –**  
   Single level directory is simplest directory structure.In it all files are contained in same directory which make it easy to support and understand.

A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have the unique name . if two users call their dataset test, then the unique name rule violated.

222.jpg

**Advantages:**

* + Since it is a single directory, so its implementation is very easy.
  + If the files are smaller in size, searching will become faster.
  + The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

**Disadvantages:**

* + There may chance of name collision because two files can not have the same name.
  + Searching will become time taking if the directory is large.
  + In this can not group the same type of files together.

1. **Two-level directory –**  
   As we have seen, a single level directory often leads to confusion of files names among different users. the solution to this problem is to create a separate directory for each user.

In the two-level directory structure, each user has there own *user files directory (UFD)*. The UFDs has similar structures, but each lists only the files of a single user. system’s *master file directory (MFD)* is searches whenever a new user id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.

333.jpg

**Advantages:**

* + We can give full path like /User-name/directory-name/.
  + Different users can have same directory as well as file name.
  + Searching of files become more easy due to path name and user-grouping.

**Disadvantages:**

* + A user is not allowed to share files with other users.
  + Still it not very scalable, two files of the same type cannot be grouped together in the same user.

1. **Tree-structured directory –**  
   Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.  
   This generalization allows the user to create there own subdirectories and to organize on their files accordingly.

444.jpg

A tree structure is the most common directory structure. The tree has a root directory, and every file in the system have a unique path.

**Advantages:**

* + Very generalize, since full path name can be given.
  + Very scalable, the probability of name collision is less.
  + Searching becomes very easy, we can use both absolute path as well as relative.

**Disadvantages:**

* + Every file does not fit into the hierarchical model, files may be saved into multiple directories.
  + We can not share files.
  + It is inefficient, because accessing a file may go under multiple directories.

1. **Acyclic graph directory –**  
   An acyclic graph is a graph with no cycle and allows to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.

It is used in the situation like when two programmers are working on a joint project and they need to access files. The associated files are stored in a subdirectory, separating them from other projects and files of other programmers, since they are working on a joint project so they want the subdirectories to be into their own directories. The common subdirectories should be shared. So here we use Acyclic directories.

It is the point to note that shared file is not the same as copy file . If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.

555.jpg

**Advantages:**

* + We can share files.
  + Searching is easy due to different-different paths.

**Disadvantages:**

* + We share the files via linking, in case of deleting it may create the problem,
  + If the link is softlink then after deleting the file we left with a dangling pointer.
  + In case of hardlink, to delete a file we have to delete all the reference associated with it.

1. **General graph directory structure –**  
   In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.  
   The main problem with this kind of directory structure is to calculate total size or space that has been taken by the files and directories.

666.jpg

**Advantages:**

* + It allows cycles.
  + It is more flexible than other directories structure.

**Disadvantages:**

* + It is more costly than others.
  + It needs garbage collection.

## File structure

A File Structure needs to be predefined format in such a way that an operating system understands . It has an exclusively defined structure, which is based on its type.

Three types of files structure in OS:

* A text file: It is a series of characters that is organized in lines.
* An object file: It is a series of bytes that is organized into blocks.
* A source file: It is a series of functions and processes.

## File Access Methods

File access is a process that determines the way that files are accessed and read into memory. Generally, a single access method is always supported by operating systems. Though there are some operating system which also supports multiple access methods.

Three file access methods are:

* Sequential access
* Direct random access
* Index sequential access

### Sequential Access

In this type of file access method, records are accessed in a certain pre-defined sequence. In the sequential access method, information stored in the file is also processed one by one. Most compilers access files using this access method.

* Records are stored and accessed in a particular order sorted using a key field.
* Retrieval requires searching sequentially through the entire file record by record to the end.
* Because the record in a file are sorted in a particular order, better file searching methods like the binary search technique can be used to reduce the time used for searching a file .
* Since the records are sorted, it is possible to know in which half of the file a particular record being searched is located, Hence this method repeatedly divides the set of records in the file into two halves and searches only the half on which the records is found.
* For example, of the file has records with key fields 20, 30, 40, 50, 60 and the computer is searching for a record with key field 50, it starts at 40 upwards in its search, ignoring the first half of the set.

**Advantages of sequential file organization**

* The sorting makes it easy to access records.
* The binary chop technique can be used to reduce record search time by as much as half the time taken.

**Disadvantages of sequential file organization**

* The sorting does not remove the need to access other records as the search looks for particular records.
* Sequential records cannot support modern technologies that require fast access to stored records.
* The requirement that all records be of the same size is sometimes difficult to enforce.

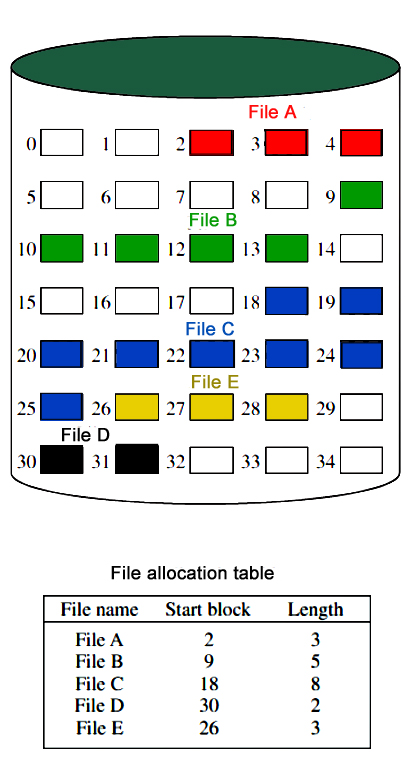
### Random Access

The random access method is also called direct random access. This method allow accessing the record directly. Each record has its own address on which can be directly accessed for reading and writing.

### Sequential Access

This type of accessing method is based on simple sequential access. In this access method, an index is built for every file, with a direct pointer to different memory blocks. In this method, the Index is searched sequentially, and its pointer can access the file directly. Multiple levels of indexing can be used to offer greater efficiency in access. It also reduces the time needed to access a single record.

**FILE ALLOCATION METHODS**

**1. Continuous Allocation:** A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block. For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.  
  
  
**Disadvantage**

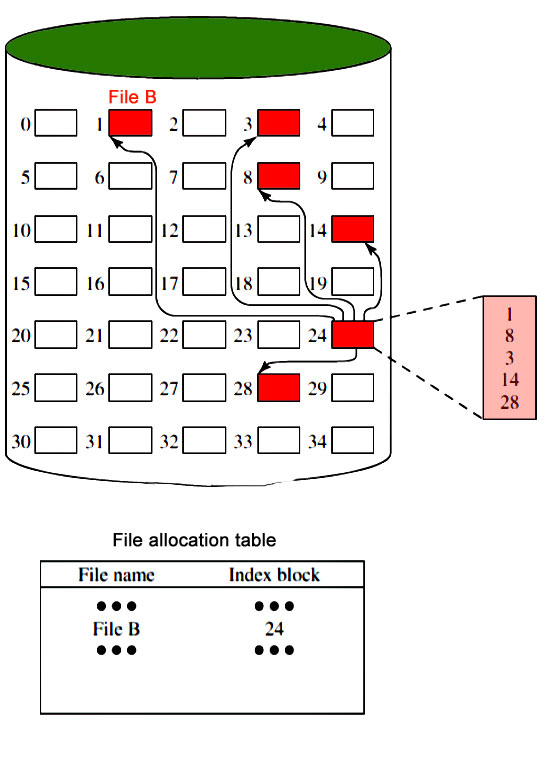
* External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. Compaction algorithm will be necessary to free up additional space on disk.
* Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

**2. Linked Allocation(Non-contiguous allocation) :** Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. Increase in file size is always possible if free disk block is available. There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

**Disadvantage:**

* Internal fragmentation exists in last disk block of file.
* There is an overhead of maintaining the pointer in every disk block.
* If the pointer of any disk block is lost, the file will be truncated.
* It supports only the sequencial access of files.

**3. Indexed Allocation:**  
It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file: The index has one entry for each block allocated to the file. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality.

This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.  


**Disk Free Space Management**

Just as the space that is allocated to files must be managed ,so the space that is not currently allocated to any file must be managed. To perform any of the file allocation techniques,it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table.The following are the approaches used for free space management.

1. **Bit Tables** : This method uses a vector containing one bit for each block on the disk. Each entry for a 0 corresponds to a free block and each 1 corresponds to a block in use.  
   For example: 00011010111100110001

In this vector every bit correspond to a particular block and 0 implies that, that particular block is free and 1 implies that the block is already occupied. A bit table has the advantage that it is relatively easy to find one or a contiguous group of free blocks. Thus, a bit table works well with any of the file allocation methods. Another advantage is that it is as small as possible.

1. **Free Block List** : In this method, each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.

# Secondary Storage and Disk Scheduling Algorithms

Secondary storage devices are those devices whose memory is non volatile, meaning, the stored data will be intact even if the system is turned off. Here are a few things worth noting about secondary storage.

* Secondary storage is also called auxiliary storage.
* Secondary storage is less expensive when compared to primary memory like RAMs.
* The speed of the secondary storage is also lesser than that of primary storage.
* Hence, the data which is less frequently accessed is kept in the secondary storage.
* A few examples are magnetic disks, magnetic tapes, removable thumb drives etc.

# Magnetic Disk Structure in Operating System

In computers the secondary storage typically is formed of stacked up magnetic disks on the top of one another.

One single unit of such disk is called **platter**.

A single unit of secondary storage device like HDD may have **100-200** such platters stacked up on one another.

Each platter is divided into circular shaped **tracks.**

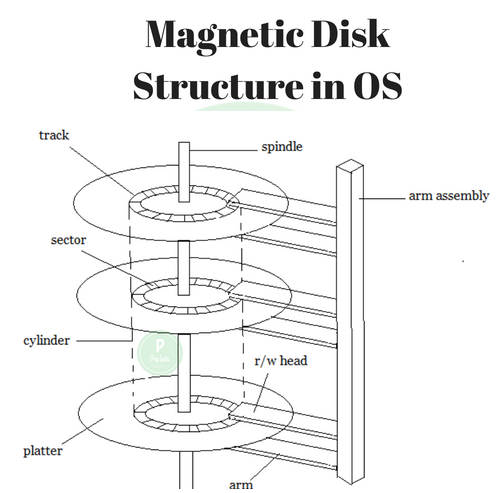
**Note** – The length of the tracks near the centre is less than the length of the tracks farther from the centre.**(Asked in AMCAT, HirePro)**

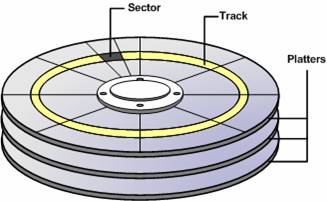
Each track is further divided into **sectors.**

Spindle revolves the platters and is controlled by r/w unit of OS. Some advanced spindles have capability to only revolve a particular disk and keep others intact.

Arm Assembly is there which keeps a pointy r/w head on each disk to read of write on a particular disk.

A world cylinder may also be used at times to refer disk stack.





The speed of the disk is measured as two parts:

* **Transfer rate:** This is the rate at which the data moves from disk to the computer.
* **Random access time:** It is the sum of the seek time and rotational latency.

**Seek time** is the time taken by the arm to move to the required track. **Rotational latency** is defined as the time taken by the arm to reach the required sector in the track.

**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 is:

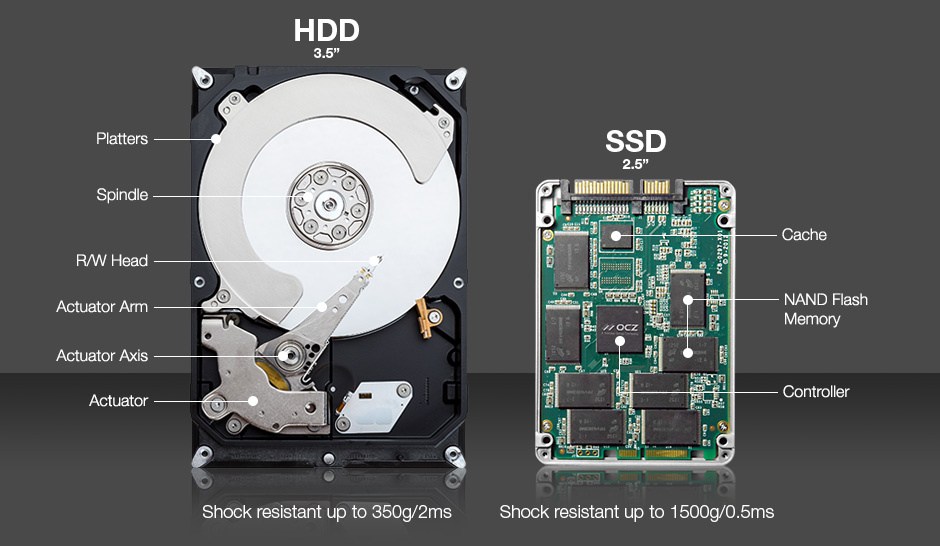
Seek Time + Rotational Latency + Transfer Time

Disk Response time is the average of all the tasks starting from disk i/o request till the time when first data output from disks is sent to the system (This is given wrong on GeeksforGeeks)

Even though the disk is arranged as sectors and tracks physically, the data is logically arranged and addressed as an array of blocks of fixed size. The size of a block can be **512** or **1024** bytes. Each logical block is mapped with a sector on the disk, sequentially. In this way, each sector in the disk will have a logical address.



### HDD Structure



### HDD Vs SSD

SSD’s are A solid-state drive (SSD) is a solid-**state storage device**that uses integrated circuit assemblies as memory to store data persistently.

They don’t have disks, thus at times also known as shock resistant storage system as they don’t break is hard disk falls.(Generally, as everything breaks if you know how to throw :D)

## Disk Scheduling Algorithm in OS

As we all know a hard disk(typically found in a computer) is a collection of disks placed on the top of one another. A typical hard disk may have 100-200 such disks stacked.

Now to have the best seek time or reduce seek time of an hard disk, we have different types of algorithms. Most common are mentioned below.

There are more advanced disk scheduling algorithm which are trade secrets of companies like – HP, Sandisks, etc. You have to work in those companies ;), to know them.



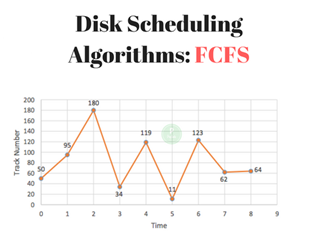
**First Come First Serve**

Now, this algorithm is fairly simple. In the order the request would come, in the same order visit the address on the disk.

**Example –**

Track Range from 0 to 199 and head initially is rested on 50

**95, 180, 34, 119, 11, 123, 62, 64**



First the head from 50 goes to 95 then**→**95**→**180**→**34**→**119**→**11**→**123**→**62**→**64

Total Head Movement Computation: (THM)

(95 – 50) + (180 – 95) + (180-34) + (119-34) + (119-11) + (123-11) + (123-62) + (64-62)

**Trick**

Now rather than doing (95 – 50) + (180 – 95) since direction doesnt change we can do (180 – 50).

The above step will save a lot of your time.

= (180 – 50) + (180-34) + (119-34) + (119-11) + (123-11) + (123-62) + (64-62) = 130 + 146 + 85 + 108 + 112 + 61 + 2 (THM) = 644 tracks

Assuming a seek rate of 5 milliseconds is given, we compute for the seek time using the formula:

Seek Time = THM \* Seek rate = 644 \* 5 ms

Seek Time = 3,220 ms

**Shortest Seek Time First (SSTF) –**

To save seek time, it would be better to visit the closest address first right.

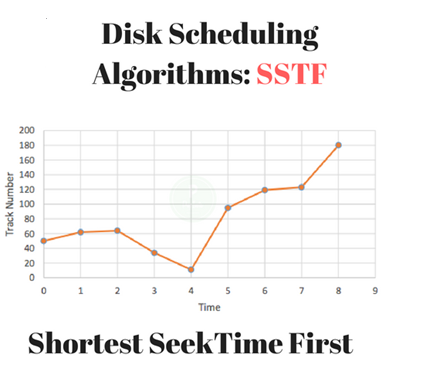
Image if we start from 60 and 199 and 58 are FCFS order so first it will go from 60 to 199 and then from 199 to 58.

In this case total movement is – (199-60) + (199 – 58) = 280

However, if we visit closest from current r/w head, which is 60 then first we will visit 58 and then 199.

Total movement – (60-58) + (199 – 58) = 143.

Which is considerably lower than 280.



(THM) = (64-50) + (64-11) + (180-11)

= 14 + 53 + 169

(THM) = 236 tracks

Seek Time = THM \* Seek rate = 236 \* 5ms

Seek Time = 1,180 ms

In this algorithm, request is serviced according to the next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34.

The process would continue up to the last track request. There are a total of 236 tracks and a seek time of 1,180 ms, which seems to be a better service compared with FCFS which there is a chance that starvation would take place.

The reason for this is if there were lots of requests closed to each other, the other requests will never be handled since the distance will always be greater.

**SCAN**

Now, even in SSTF to and fro movement is there a lot.

Wouldn’t it be better if we just moved in one direction and once all requests in that direction are complete changed the direction and moved to 2nd direction.

Note – No other website tells you this in Scan the first request or direction decision is based on the nearest request. Example –

In Case below, from 50 it goes to 34 as its closer.

In Scan it will scan till the whole limit for our case its 0 and 199 to both will be scanned.



(THM) = (50-0) + (180-0) = 50 + 180 (THM) = 230

Seek Time = THM \* Seek rate = 230 \* 5ms

Seek Time = 1,150 ms

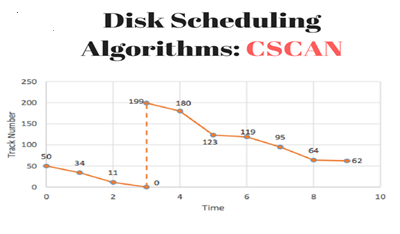
**C-SCAN**

The idea comes from a disk, this is a modified version of scan, in which we changed the direction once we reached the end address for that scan.

Like in previous example, once we reached 199 we changed the direction.

In case of disks, which are circular 199 and 0 will be side by side. Correct?

If you still don’t understand – imagine a circle on the disk, now divide that circle in 200 equal parts starting from 0 till 199. Now imagine !!!  its easy really.



This algorithm is a modified version of the SCAN algorithm. C-SCAN sweeps the disk from end-to-end, but as soon it reaches one of the end tracks it then moves to the other end track without servicing any requesting location.

As soon as it reaches the other end track it then starts servicing and grants requests headed to its direction.

This algorithm improves the unfair situation of the end tracks against the middle tracks. Using the same sets of example in FCFS the solution are as follows:

Note – In CSCAN a new variable alpha(α) is there which is basically adjustment time to shift from end to start(in this case from 0 to 199).

In ideal cases alpha(α) is 0. But, in real time cases alpha is some mili seconds.

Assume α = 20ms

(THM) = (50-0) + (199-62) + α = 50 + 137 + 20 (THM) = 207 tracks

Seek Time = THM \* Seek rate = 187 \* 5ms

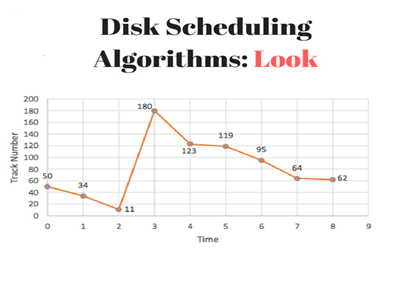
Seek Time = 935 ms

**Look**

Look is exactly like SCAN, but rather than going end to end like going to 0 and 199 both depending on the direction of the movement.

It looks for the last service required in either direction, hence called look.

In our case 11 and 180 are the last services.



(THM) = (50-11) + (180-11) = 39 + 169

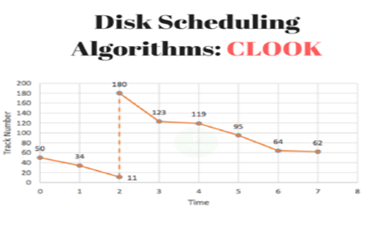
(THM) = 208 tracks

Seek Time = THM \* Seek rate = 208 \* 5ms

Seek Time = 1,040 ms

**C-Look**

C look again is modified version of looks. It is knows the last services in the either direction and also has alpha value(α) as, shift adjustment value.



Take alpha value as 20ms again.

(THM) = (50-11) + (180-62) + α = 39 + 118 + 20 (THM) = 177 tracks

Seek Time = THM \* Seek rate = 157 \* 5ms

Seek Time = 785 ms